UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

LEONARDO FELIPE MENDES

Análise comparativa de desenvolvedores e analistas de qualidade para atuação em correção e manutenção de softwares

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Engenharia da Computação

Orientador: Prof[a]. Dr[a]. Renata Galante

Porto Alegre
2023

AGRADECIMENTOS

Gostaria de agradecer a minha esposa, Lisiane Alves Vieira, a qual me acompanha desde 2018 me dando força, afeto, carinho, amor e muita parceria ao longo da nossa trajetória como casal.

Agradeço aos meus pais, Grace Anne Sampaio e Fernando Riveira Mendes, os quais não mediram esforços para fazer esse sonho de finalizar a universidade acontecer, dando-me artifícios para conseguir me manter e levantar vôo em minha vida. Também quero agradecer meus irmãos, Mikael Sampaio Martins e Rafaela Tanise Mendes, que me apoiaram em minha escolha nesse curso.

Em especial minha mãe e meu irmão, gostaria de dizer que depois de todos os percalços da vida que já passamos, podemos dizer com afinco: "vencemos".

Queria agradecer aos meus avós paternos, Salete Correa Mendes e José Luis Mendes, os quais não poderão ter o prazer de me ver colar grau nessa vida, mas que com certeza teriam orgulho enorme de ver o neto alcançar seu sonho de menino.

Agradeço aos diversos colegas da universidade, em especial aqueles que iniciaram comigo no segundo semestre de 2014, sendo agora um grupo de amigos que se apoiam em qualquer ocasião.

Agradeço a professora Renata Galante, que acreditou na minha proposta, orientou-me com sabedoria, tranquilidade, dando-me os conselhos corretos, além de tornar um trabalho de conclusão de curso uma experiência menos assustadora possível.

Muito obrigado.

RESUMO

Sistemas de grandes corporações, como instituições financeiras, possuem diferentes níveis de complexidade e, consequentemente, impactos para seus usuários. Assim, tendo em vista o cenário de tratamento de erros, é fundamental que os sistemas desenvolvidos contenham em suas construções as definições de entregáveis com transparência e objetividade, sob pena de ocorrerem falhas ou comportamentos inadequados durante a experiência para os usuários. Portanto, a análise que precisa ser feita pela equipe de desenvolvimento para a correção definitiva desses erros precisa ser muito precisa. O objetivo central deste trabalho é analisar o comportamento dos profissionais envolvidos para compreender e atuar neste cenário. Foi realizada uma pesquisa por meio de formulários do tipo "Survey" para grupos de profissionais de uma instituição financeira. Com base nesses dados, foi realizada uma análise dos principais elementos de atuação e preocupações dos profissionais diante de cenários de correção e manutenção de erros em comparação com o desempenho na construção e evolução dos sistemas. Os resultados apontaram para uma convergência de maior tração nas atividades de evolução, justificado pela diretriz da instituição em proporcionar maior fluidez para essas atividades.

Palavras-chave: Manutenção de software. métodos ágeis. incidentes. evolução de software. governança de TI. instituições financeiras.

Comparative analysis of developers and quality analysts for performance in software correction and maintenance

ABSTRACT

Systems of large corporations, such as financial institutions, have different levels of complexity and, consequently, impacts on their users. Thus, in view of the error handling scenario, it is essential that the developed systems contain in their constructions the definitions of deliverables with transparency and objectivity, otherwise failures or inappropriate behavior may occur during the experience for users. Therefore, the analysis that needs to be carried out by the development team for the definitive correction of these errors needs to be very precise. The central objective of this work is to analyze the behavior of the professionals involved in order to understand and act in this scenario. A survey was carried out using forms of the "Survey" type for groups of professionals from a financial institution. Based on these data, an analysis was carried out of the main elements of actions and concerns of professionals in the face of error correction and maintenance scenarios in comparison with performance in the construction and evolution of systems. The results pointed to a convergence of greater traction in evolution activities, justified by the institution's guideline to provide greater fluidity for these activities.

Keywords: software maintenance, agile methods, incidents, software evolution, IT governance, financial institution.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

IF	Instituição Financeira
TI	Tecnologia da Informação
ITIL	*Information Technology Infrastructure Library*
COBIT	*Control Objectives for Information and related Technology*

SUMÁRIO

1 INTRODUÇÃO

Sistemas de grandes corporações, principalmente de instituições financeiras (IF), por muitas vezes possuem um alto grau de complexidade, grande número de facetas de serviços e utilidades para seus usuários. Listando infinidades de funcionalidades vitais, como transferências instantâneas para quaisquer outras IF, conferência de saldo e extratos de sua conta-corrente, investimentos *on-line*, contratação de empréstimos, financiamentos e muitos outros serviços. Essa funcionalidades podem estar presentes em mais diferentes níveis de canais disponíveis, desde autoatendimento como o seu *internet banking* pelo computador ou pelo aplicativo *mobile* de seu *smartphone* ou então presencialmente em uma unidade de atendimento da IF. Portanto, essas facetas de serviços necessitam de uma grande infraestrutura tecnológica para sua disponibilização para os clientes.

Toda essa estrutura demanda, por si só, uma grande rede de suporte em casos de dúvidas no transacional ou erros quaisquer durante as jornadas propostas pelas experiências iniciais do sistema a nível operacional. Para isso, é necessário uma vasta equipe de diversos profissionais responsáveis desde o primeiro contato com o cliente até o *feedback* final que sua demanda foi concluída. Segundo a Federação Brasileira de Bancos (2022) (FEBRABAN), em 2020 67% das transações bancárias foram realizadas pelos canais digitais (*internet banking* e *smartphones*), o que reforça a demanda por suporte técnico em caso de falhas ou eventos extraordinários que interrompam o funcionamento desses sistemas. Essas falhas em sistemas e eventos que interrompam o funcionamento são representados pelo conceito de incidentes e problemas, que englobam o universo citado em governança de TI, os quais citam os *frameworks* do COBIT e ITIL, que envolve atuação necessária de profissionais alocados para essa função. Porém, em algumas IF, esses mesmos profissionais podem estar alocados em um time de desenvolvimento de um grande projeto, o que engloba os conceitos de gerenciamento de projetos (PMBOK) e possíveis metodologias ágeis (*Scrum* ou Kanban) a serem utilizadas pelas equipes.

Esses conceitos citados anteriormente englobam os ambientes com os quais esses profissionais de TI, público alvo desta pesquisa, estão alocados em instituições financeiras. Nesse sentido, o objetivo deste trabalho é analisar e comparar, através de uma pesquisa tipo "survey", a rotina dos profissionais de TI envolvidos para atendimento de demandas oriundas do suporte ao cliente, muitas vezes englobando possíveis manutenções no sistema produdo, em comparação com a atuação do mesmo profissional em relação a atuação em evolução de sistemas ou elaboração de novas *features* do sistema. Sendo

possível, após análise dos resultados, compreender e analisar, a partir do cotidiano desses profissionais, como se relacionam os conceitos da literatura na aplicação dentro do ciclo de desenvolvimento para lidar com situações de manutenção de *software* versus evolução de *software*. A pesquisa, por si só, explora de maneira teórica e técnica, de acordo com o contexto e cotidiano de uma IF, trazendo possíveis inferências que podem trazer resultados na análise de perfil de cada profissional e até mesmo entender onde que podem estar sendo alocado os esforços a nível de evolução ou manutenção do sistema produto.

Há diversas abordagens de outros trabalhos que trazem reflexo de uma possível realidade independente para uma equipe de desenvolvimento lidando com gerência de projetos de *software* tendo ênfase em manutenção ou evolução, isto é, cada projeto com uma finalidade diferente, utilizando ou não das metodologias ágeis presentes e amplamente utilizadas no mercado, sendo o mais comum haver uma estrutura de time ágil *Scrum*, com membros fixos de sete a oito profissionais. No contexto do trabalho, tanto as atividades de manutenção e evolução do sistemas ocorrem de maneira concomitante, trazendo maior riqueza para análise, uma vez que é possível entender muito melhor o que autores como Lehman e Ramil (2000), Chapin et al. (2001) e Stammel et al. (2011) contribuem para o que é um conceito de mudanças do estado original do software, que tem como motivação na IF estudada em ambas vertentes na perspectiva dos profissionais alocados.

O restante do trabalho está organizado da seguinte forma: O Capítulo 2 descreve a fundamentação teórica que está dividida em Governança de TI, COBIT, ITIL, Gerenciamento de projetos, Metodologias ágeis, Scrum, Kanban, Manutenção de *Software* e Evolução de *Software*, que serão utilizadas para contextualização do universo, os quais estão submetidos os profissionais de TI de IF, que serão estudados e compreendidos através da pesquisa. O Capítulo 3 lista os trabalhos relacionados, os quais foram compreendidos e citados ao longo da elaboração desse trabalho. O Capítulo 4 contém a proposta e metodologia que será utilizada para esse trabalho, exemplificando aqui o tipo de pesquisa a ser aplicada bem como sua estrutura e como tecnicamente será aplicado nas perguntas do questionário. O Capítulo 5 contém a análise dos resultados obtidos, após coleta de dados dos respondentes. E, por fim, o Capítulo 6 cita as conclusões dos resultados obtidos nesse trabalho.

2 CONCEITOS E TECNOLOGIAS UTILIZADAS

Esse capítulo apresenta os principais conceitos que fundamentam a estrutura e desenvolvimento desse trabalho. O objetivo central é contextualizar todo o universo de conceitos abordados para que seja possível entender o contexto que será aprofundado nesse trabalho.

2.1 Governança de TI

Segundo Oliveira e Malagolli (2016) e Albertin (1999), a tecnologia de informação é uma poderosa ferramenta empresarial. Dessa forma, é comum que hoje impreterivelmente toda e qualquer instituição financeira ou bancos devem possuir uma estrutura de tecnologia para lidar com o oferecimento de seus serviços, inferindo assim, que junto com toda essa carga de tecnologia, sejam necessário o papel da governança de TI.

Segundo Barbosa e Lima (2011), governança de TI é um conjunto de práticas, padrões e relacionamentos estruturados, que são permeados muitas vezes por altas escalações de gestão e diretorias, tendo como principal objetivo garantir que os processos relacionados a TI tenham mais garantia de controle, assertividade no quesito segurança, ampliação de desempenho das áreas que possuem as diretrizes impostas, alinhar negócios junto com as tecnologias e também as áreas de TI, além de reduzir custos para organização como um todo.

Também é relatado no artigo que governança de TI começa a desempenhar três elementos chaves essenciais para a governança corporativa: valor, risco e controle. O trabalho terá como foco a dedicação ao valor, visto que um dos grandes pilares das entregas de sistemas são as que agreguem valor final ao usuário, muitas vezes sendo evolução de produtos financeiros que serão utilizados através de sistemas ERP (*Enterprise Resource Planning*) bancários.

2.2 COBIT

Um dos modelos utilizados para implementação da governança de TI é o COBIT (*Control Objectives for Information and related Technology*) (BARBOSA; LIMA, 2011) é um modelo e uma ferramenta de suporte que permite as gerências de TI de conseguirem

visualizar todos os controles com sentido crítico, de forma que possua capacidade de enxergar as debilidades dos processos de forma operacional ou que possuam algum tipo de risco de negócios. Por sua vez, após a percepção desses elementos fazer o correto reporte as áreas interessadas, tornando a empresa capaz de prosseguir com os padrões de controles de TI para a empresa como um todo.

É perceptível e factível inferir que grandes corporações como instituições financeiras possuem processos e metodologias utilizadas através do COBIT para governança de TI, uma vez que, principalmente no quesito geração de valor ao seu cliente final, é necessário possuir um processo claro de mapeamento de todos os controles e exposição dos processos e entregas de forma crítica, que possam demonstrar não só o que é entregue no sentido de produto financeiro e negócio, mas sim como foi o processo de entrega.

COBIT possui segundo Barbosa e Lima (2011), 4 domínios com 34 processos sendo os domínios:

1. Planejar e organizar - proposição de diretrizes para planejamentos de todos os serviços entregavéis, construção das diretrizes que estejam em alinhamento com a corporação, gerenciar os projetos, riscos e custos (BARBOSA; LIMA, 2011).
2. Aquisição e implementação - focada em soluções para a entrega de serviços, principalmente tendo como o foco o seu produto interno, por exemplo, a elaboração de um sistema de software que será utilizado em unidade de atendimento para atendimento aos clientes.
3. Entregar e suportar - garantia de após a implementação a entrega ocorra para os clientes finais, além de propor elementos para lidar com problemas e incidentes que possam surgir em função da operacionalização do uso desse entregável.
4. Monitoramento - supervisão dos processos de forma que esteja sempre o mais próximo alinhado aos processos impostos pela diretriz da corporação (BARBOSA; LIMA, 2011).

Dos processos, daremos foco aos que estão na esfera do domínio de Entrega e Suportar (BARBOSA; LIMA, 2011):

1. Acordo de níveis de Serviço (SLA);
2. Serviços de Terceiros;
3. Gestão de capacidade (performance dos elementos de *hardware* e *software*;
4. Gestão de continuidade de serviços;
5. Gerencia Incidentes e Problemas;

6. Gerencia dados; e
7. Gerencia infraestrutura e operações.

Dos processos citados anteriormente, os que geram demanda para correção e manutenção de sistemas muitas vezes são originadas por incidentes e problemas que são detectados, analisado e corrigidos pelas equipes de TI da IF. Normalmente, ocorre através de atendimento a clientes de forma direta em um suporte ao cliente (SAC) ou então por meio de um suporte ao colaborador. Note que por ser demanda que relacionam diretamente com a produção da instituição e também com a satisfação do cliente interno, que é o próprio colaborador que auxília no atendimento, ou externo, que é o cliente final de fato, o processo precisa acontecer em um espaço de tempo dentro do acordo de nível de serviço (SLA *Service Level Agreement*) e para isso precisará necessariamente da intervenção das equipes de TI para solução desses incidentes e problemas com maior agilidade possível.

2.3 ITIL

ITIL (*Information Technology Infrastructure Library*) segundo Barbosa e Lima (2011) foi desenvolvido em meados de 1980 pelo governo britânico tendo como característica possuir uma estrutura bastante útil em empresas no quesito de gerenciamento de serviços de TI. Durante a década de 90 foi quando ela foi consolidada como um padrão de gerenciamento de serviços.

Nesse *framework*, segundo Adams (2009), é constada uma série de melhores práticas para lidar com a gama de serviços como o gerenciamento de *help desk*, incidentes, mudanças, capacidade, problemas, segurança e etc. Também segundo Barbosa e Lima (2011), o ITIL rastreia os problemas nessas áreas de serviços de TI citadas anteriormente através do suporte a sistemas havendo contato com o cliente interno ou externo. Portanto, podemos inferir que os serviços de suporte do ITIL podem servir de auxílio para a formação da estrutura de atendimento quando o mesmo tiver a motivação de ser relacionado a TI, visto que é notável que esses serviços de suporte tem como finalidade o alinhamento aos objetivos de negócios. No contexto das IF não é diferente, uma vez que a demanda por atendimento pode resultar muitas vezes em itens que podem receber a classificação de incidente que poderá evoluir a um problema.

Dessa forma, os focos principais do ITIL podem ser listados segundo Barbosa e Lima (2011) e Adams (2009) como:

- Gestão de operações e infraestrutura de TI;
- Fornecimento de qualidade de serviço dos clientes de TI; e
- Relacionar os custos dos serviços de tecnologia e como estes trazem valor estratégico ao negócio.

Os processos do ITIL podem ser divididos, segundo Adams (2009) como:

- Gerenciamento de infraestrutura de tecnologia de comunicações e informações (TCI);
- Gerenciamento de serviços; e
- Gerenciamento de aplicações.

Por sua vez, segundo Barbosa e Lima (2011) cada um desses processos possuem suas próprias subdivisões, entrando em Gerenciamento de infraestrutura e TCI por exemplo:

- Projeto e planejamento;
- Implantação;
- Operação; e
- Suporte técnico.

Essas subdivisões citadas anteriormente, no contexto das IF, simbolizam uma das possíveis etapas de desenvolvimento de um sistema ou nova *feature* a ser implantada ou evoluída no sistema ou produto financeiro, ou possuir motivação de correção, que por inferência podem surgir a partir do próprio suporte técnico. Note que essas etapas são importantes no contexto dessa pesquisa, uma vez que manutenção e suporte técnico a esses sistemas possuem motivações diferentes em relação as evoluções de sistema de software. Nesse contexto, evolução entende-se por qualquer nova *feature* no sistema de software em relação ao que já existe. Por outro lado, as motivações de correção partem da necessidade de manter o sistema estável, entrando no mérito de estabilidade e sustentação que (ADAMS, 2009) tem como premissa, e essas necessidades, normalmente, são oriundas de incidentes ou problemas, relacionados aos sistemas ou produtos, originados a partir de uma central de atendimento ainda no contexto de uma IF.

Nos processos de Gerenciamento de serviços, há a classificação dos serviços como processos de entregas e suporte, sendo os de entregas:

- Gerenciamento de capacidade;
- Gerenciamento de disponibilidade;

- Gerenciamento de finanças;
- Gerenciamento de níveis de serviço; e
- Gerenciamento de continuidade dos serviços de TI.

E os de suporte:

- *Service desk*;
- Gerenciamento de incidentes;
- Gerenciamento de problemas;
- Gerenciamento de configuração;
- Gerenciamento de mudanças; e
- Gerenciamento de versões.

Dentro das IF, podemos destacar o gerenciamento de níveis de serviço, os quais incluem o acordo de nível de serviço, também conhecido pela sigla SLA (*Service Level Agreement*) o qual estabelece e assegura um acordo para uma prestação de um bom nível de serviço entre o provedor e o usuário. Nesse caso, o provedor seria a própria IF e os usuários podem ser tanto os internos (colaboradores da própria IF) como os externos (clientes finais). Um desses acordos, o qual é bastante comum, é o tempo acordado para atendimento de um incidente por exemplo.

Ainda nesse contexto, podemos destacar os gerenciamentos de incidentes e problemas, que são justamente os que mais demandam atenção das equipes de TI no quesito sustentação da operação e manutenção de software. Internamente, cada IF pode dividir e classificar esses incidentes e problemas da forma como preferir sendo comum a prática disso justamente para modularizar a atuação de suas equipes de TI. Um possível exemplo é separar os incidentes e problemas por produto financeiro, como cartão de crédito, cheque-especial, empréstimos e etc, ou então por canal digital, como *internet banking*, *mobile* e etc, sendo essa uma escolha de melhor organização da própria IF.

2.4 Gerenciamento de Projetos

Um projeto, segundo Moraes (2012) é a entrega de um empreedimento com início, meio e fim bem definidos, que utilizam recursos disponíveis e liderado e composto por pessoas que tem um comum objetivo de atingir uma meta de tempo, de custo e de qualidade definida por um corpo estratégico de empresas. Nesse contexto, é importante

salientar que empresas que não planejam bem seus projetos podem estar fadadas ao fracasso (ANDRADE; TAIT, 2012). Por outro lado, segundo Andrade e Tait (2012) gerência de projetos é aplicação de conhecimentos, habilidade e ferramentas e técnicas nas atividades de projeto, com o objetivo de cumprir as necessidades definididas pelos *stakeholders*.

Dessa forma, podemos elencar que são necessárias boas práticas de gerência de projetos, sendo o PMBOK (*Project Management Body of Knowledge*), que foi publicada pela PMI (*Project Management Institute*) empresa norte-americana que desde 1969 vem atualizando esse guia de boas práticas.

O guia, segundo Moraes (2012) e Cruz (2013) é composto por processos que podem ser classificados em duas categorias:

- Processos de gerenciamento de projetos; e
- Processos orientados ao produto.

Processos de gerenciamento de projeto, segundo (PMBOK, 2012), possuem a característica de descrever, organizar e finalizar o trabalho do projeto. Esses processos podem aplicavéis em quaisquer tipos de projetos, desde o de uma construção de um edifício até um de engenharia de software (CRUZ, 2013). Já os processos orientados ao produto se caracterizam por criar ou especificar um produto do projeto, e isso pode variar de acordo com o projeto e área de aplicação de cada projeto.

Esses dois grandes grupos de processos integram-se durante toda existência do projeto, e o PMBOK possui 44 processos de gerenciamento de projetos que são organizados em cinco grupos (CRUZ, 2013):

- Processos de iniciação;
- Processos de planejamento;
- Processos de Execução;
- Processos de Controle; e
- Processos de Encerramento.

Dessa forma, internamente em uma IF, a importância de se haver uma gestão de projetos organizadas e transparentes, de forma que atendam os *stakeholders* são primordiais. Usualmente, as IF utilizam os projetos baseados em processos de orientados a produtos, uma vez que grande parte dos esforços em projetos são para evolução de seus próprios produtos. Os afincos realizados para projetos de criação de novos sistemas em grande parte são o tempo todo complementado pelas novas *features* que os produtos vão ou adicionando ou aperfeiçoando. Ainda nesse contexto, é importante salientar que con-

forme PMBOK (2012), há pessoas envolvidas que precisam ser lideradas e estarem dentro de seus escopos, nesse sentido, é comum ver em IF ou grandes bancos, times que possuem focos em produtos, que podem possuir um número fixo ou variável de pessoas partindo de um projeto a ser gerenciado.

2.5 Metodologias Ágeis

Atualmente, grande parte das empresas são carentes de inovação, principalmente no quesito tecnológico, uma vez que cada vez mais se faz necessário o seu negócio estar preparado para as evoluções que os seus clientes esperam. No entanto, essas carências não foram sentidas apenas no século atual, e sim foi uma grande debilidade de boa parte das empresas que queriam buscar destaque frente a seus modelos de negócios. Nesse sentido, em 2001, 17 especialistas em processos de desenvolvimento de *software* se reuniram e estudaram medidas que aumentariam os sucessos dos projetos. Porém, para isso, quatro princípios são essenciais. Para divulgação desses princípios, foi publicado o que conhecemos como Agile Manifesto (2001), que destacam:

- Indivíduos e interações em vez de processos e ferramentas;
- Software executável em vez de documentação;
- Colaboração do Cliente ao invés de negociação de contrato; e
- Rápidas respostas a mudanças ao invés de seguir planos (KOSCIANSKI; SOARES, 2007).

Dessa forma, destacaremos duas das abordagens de metologia de desenvolvimento ágil utilizada em larga escala em empresas em seus times de desenvolvimento: o Scrum e o Kanban. Especificamente, no público alvo dessa pesquisa, os times podem estar utilizando uma delas, ou ambas combinadas, sendo esta bastante comum essa prática pelas equipes da IF em questão.

2.6 *Scrum*

O Scrum, segundo Schwaber e Sutherland (2010), é um framework de desenvolvimento ágil de software que possui uma abordagem interativa e incremental, que fora entendida a partir de abordagens empíricas que afirmam que o conhecimento é adquirido

através da tomada de decisões e experiência no que é conhecido.

Sua principal finalidade é tornar o processo de desenvolvimento de software mais flexível dentro de um ambiente que pode possuir um grande nível de mudanças, fazendo com que os membros dessa equipe de desenvolvimento possam estar preparados para esses eventos de forma flexível. Isso se faz necessário quando há diversos *stakeholders* contribuindo de forma frequente no projeto, que é bastante comum dentro de uma IF. Diante desse cenário, o Scrum é composto por papéis, eventos, artefatos e regras segundo (SCHWABER; SUTHERLAND, 2010), que auxiliam no processo de autoadministração da equipe.

Os papéis de um time Scrum, são (SCHWABER; SUTHERLAND, 2010):

- *Product owner*: também conhecido como dono do produto, possui a característica de ser o representante de produto e negócios dentro dessa equipe. Possui a responsabilidade de ser o dono do *backlog*, o qual constitui uma lista de tarefas a ser priorizada de acordo com a necessidade dos *stakeholders* ou clientes. O mesmo também deve manter a transparência nessa listagem de forma que todo o time tenha a percepção do que está sendo realizado, priorizado e entregue de acordo com o valor que isso gere para o produto ou negócio;
- Time de desenvolvimento: são profissionais que utilizam-se do *backlog* de produto e os incrementam em sistema ou evoluções do produto ao final de cada ciclo, também chamado de *sprint*. Nesse papel também estão contidos os analistas de qualidade, analistas de sistemas e também analistas de experiência de usuários, também conhecidos como *UX*; e
- Scrum master: é um profissional que é responsável por garantir que o *framework* seja compreendido e utilizado de forma coerente pelo time. Possui uma abordagem facilitadora, por muitas vezes, e avalia se o time está seguindo a teoria proposta pelos criadores do *framework*. Auxília o time Scrum a qualquer tempo de forma que faça com o time atenda às necessidades do projeto ou evolução do produto como um todo.

Além de papéis, definidos anteriormente, também existem os chamados eventos Scrum, sendo eles segundo Schwaber e Sutherland (2010):

- *Sprint*: um dos principais conceitos empregados no *framework*. Caracteriza-se por ser um evento de duração de uma a quatro semanas, sendo que ao final desse evento, deve-se haver um entregável, isto é um incremento do produto. Para auxílio nesse

evento, há a criação de uma definição que, por si só, guiará o processo de produção, estando fortemente alinhado ao objetivo do produto final. Dentro de cada *sprint* há outros eventos que acontecem dentro desse período, sendo que eles serão citados nos próximos itens;

- *Sprint planning*: conhecida por reunião de planejamento, é um evento o qual todo o time scrum deve estar presente, que serve exclusivamente para elaborar um planejamento dos itens a serem trabalhados em uma *sprint*, possuindo duração máxima de oito horas em um ciclo de quatro semanas. Nesse evento, também são definidos os objetivos que serão realizados ao longo da *sprint*;
- *Daily scrum*: é uma reunião diária do Scrum, tendo duração máxima quinze minutos diários. Normalmente, é executada no início das jornadas de trabalho, porém ficando a critério de cada equipe o melhor momento para realização. Nesse evento são relatados, por todos os profissionais, o que foi entregue, quais as barreiras encontradas desde a última reunião diária e o que será feito até a próxima diária. Essas reuniões diárias tem como consequência uma melhor comunicação da própria equipe, fazendo com que não hajam tantas reuniões paralelas, além de ser um elemento que auxília o dono de produto e o *Scrum master* a inspecionar o time (SCHWABER; SUTHERLAND, 2010);
- *Sprint review*: é um evento de revisão da *sprint*, que é realizada apenas ao final de cada *sprint*, que serve para apresentação e inspeção final do produto gerado. Normalmente participam os todo o time Scrum além de pessoas convidadas pelo dono de produto; e
- *Sprint retrospective*: é um evento retrospectiva do time Scrum, sendo usualmente a última reunião do ciclo inteiro, o qual todos os membros tem a oportunidade de avaliarem a si próprios e identificarem melhorias para serem seguidas nas próximas *sprints*. Normalmente dura cerca de uma hora.

Os times Scrum, contidos nessa pesquisa, estão todos seguindo o *framework* Scrum, sendo que cada IF pode decidir qual projeto ou produto que o time terá como responsabilidade. Nota-se que, durante a elaboração de um ciclo, são planejados entregas que estejam alinhados com a evolução do produto e listados pelo dono do produto. Nesse sentido, podemos inferir que os conceitos empregados por essa metodologia ágil podem não estar abrigando a possibilidade de eventos extraordinários que possam causar mal funcionamento do produto, sendo um desafio para todos os membros do time Scrum,

estarem preparados para atuar em itens não planejados que podem entrar de forma abrupta em uma *sprint*. Note que esses itens podem acabar causando o atraso na entrega que foi planejada e concordada com os membros do time na reunião de planejamento, o que pode ser um problema, em caso do item extraordinário durar muito tempo. As consequências da não possibilidade de cumprir com o objetivo de uma *sprint* podem causar a necessidade de se levar mais tempo para cumprimento do acordado com todos os *stakeholders*.

2.7 *Kanban*

A metodologia ágil Kanban surgiu com iniciativa da empresa japonesa Toyota, inclusive sendo denominada como TPS *Toyota Production System*, que tinha como intuíto controlar a produção de automóveis (SHINGO, 1996). Kanban significa visão, pela tradução livre da lingua japonesa para a lingua portuguesa.

Para o contexto de gerenciamento de projetos de software, segundo Poppendieck e Poppendieck (2003), o uso dessa metodologia teve aumento expressivo em 2007, quando teve publicados os trabalhos de *"Lean software development* e *Agile*, que apresentaram resultados a respeito do uso desse método para equipes que desenvolvem software. Desde então em IF são utilizados conceitos do Kanban para fortalecer o gerenciamento de projetos e fluxos de trabalho (*workflow*) (SILVA; SANTOS; NETO, 2012).

Segundo Kniberg (2016), a metodologia possui três prescrições:

- Visualizar o fluxo atual de trabalho;
- Limitar o fluxo de trabalho; e
- Acompanhar e gerenciar o fluxo de trabalho.

Percebe-se que diferente da metologia ágil *Scrum*, o Kanban possui menos prescrições, o que pode fazer com que haja um engajamento maior das equipes que utilizarem, já que ela é muito mais adaptativa do que as demais. Essa flexibilidade, de certa forma, contribui para a maior aceitação das equipes de desenvolvimento de software.

A primeira prescrição, segundo Kniberg (2016), indica que é necessário que o fluxo de trabalho (*workflow*) que será visualizado deve se aproximar o máximo possível da realidade de contexto de atividades do time de desenvolvimento. Importante salientar essa questão, visto que muitas das adotantes dos métodos ágeis acabam se apegando muito mais as cerimonias do que de fato aos benefícios e processos que os *frameworks* oferecem, sendo esse um ponto visível dentro da IF no contexto desse trabalho.

No segundo item da prescrição, é apresentado o conceito de WIP (*Work in Progress*), que representa uma limitação no fluxo de trabalho, fazendo de forma explícita quantos itens devem estar em cada uma das fases do processo (ANDERSON, 2010). Note que esse conceito, segundo Silva, Santos e Neto (2012), acaba se tornando uma propriedade principal da metodologia como um todo, pois permite transparecer que um item só poderá avançar de fase quando a próxima fase tiver um número menor de itens que o limite descrito pelo WIP (KNIBERG, 2016). Um exemplo disso é uma atividade de desenvolvimento estar na fase denominada pronta para testes, essa mesma atividade só poderá avançar a etapa de testes caso a fase em questão não esteja com o número máximo de atividades nessa etapa. Apesar de ser sugerido um WIP para cada uma das etapas, não há uma fórmula padronizada para isso, significando que cada time poderá ter seu próprio limite.

No terceiro item da prescrição é apresentado a definição que auxilia a medir e controlar um fluxo de trabalho. Segundo Silva, Santos e Neto (2012), é nesse ponto que os times de desenvolvimento de software realizam adaptações na metodologia, já que no contexto do desenvolvimento de software por um time específico, o mesmo pode possuir diferentes problemas aparentes, e junto com eles o time mesmo poderá ter impeto para controlar e solucionar esses problemas.

Segundo Anderson (2010) e Silva, Santos e Neto (2012) e também com suas próprias prescrições, é possível verificar que esse método ágil propõe um fluxo contínuo, isto é sem um fim definido para cada ciclo como no *Scrum*. O Kanban possui uma característica de controlar o ingresso de itens a serem explorados, e o *throughput* pode ser visualizado a partir da WIP definida nas fases, sendo esse conceito definido como *lead time*, que simboliza a contagem de tempo de um item desde sua entrada até sua saída. Na prática, no contexto da IF, esse indicador é bastante explorado, uma vez que pode ser um forte indício que time de desenvolvimento em questão está com um ritmo de entregas elevado, regular ou baixo, de acordo com o número puro de itens entregues divididos pelo *leadtime*.

Ainda nesse contexto, a partir desses indicadores, poderá ser possível também evidenciar a existência de fases que estejam com gargalos, ou seja com itens que estejam levando muito mais tempo que o usual para serem finalizados, sendo esse um ponto interessante para análise da equipe de desenvolvimento, já que podem ser necessários mais profissionais para serem responsáveis por essa atividade ou fases. Vale ressaltar que o cenário contrário também é possível.

Outra característica importante do Kanban é sua possibilidade de combinação de ferramentas de metodos diferentes, enfatizando sua flexibilidade e adaptabilidade com o fluxo de trabalho. Isso deve-se ao forte conceito baseado na ideia *Lean*, segundo Poppendieck e Poppendieck (2003).

2.8 Manutenção de Software

O conceito de manutenção de *software* é definido pela atividade de alteração de um *software* já entregue anteriormente. Essa alteração pode ser uma correção de falhas, erros, incidentes e problemas, implementação de melhorias a nível de performance e também adaptações (TRIPP et al., 1998), sendo essa uma definição bastante generalista, uma vez que manutenção e evolução de *softwares* podem estar dentro de uma mesma interpretação. Isso deve-se ao fato, que no seu cerne central, apenas a mudança original do estado do software sendo modificado já caracterizaria em uma manutenção segundo a IEEE.

Apesar desse conceito, há um grande nível de diferença, sendo que, segundo Pfleeger e Atlee (2010), há três diferentes categorizações segundo sua complexidade para manutenção de *software*. A primeira categoria refere-se aos *softwares* que foram desenvolvidos em definições tenazes e consolidadas, os quais possuem resultados muito bem conhecidos. Um simples *software* capaz de realizar operações matemáticas simples entre quaisquer números são um exemplo dessa categoria, sendo que esse tipo possui a ênfase que dificilmente terão necessidade de quaisquer manutenções ou evoluções.

A segunda categoria refere-se aos projetos de softwares que foram designados e definidos para resolverem questões mais próximas ao cotidiano de pessoas, onde há destaque para soluções que sejam mais idealizadas na teoria no momento de buscar resultados. Como exemplo vale citar um simples jogo de xadrez, onde não é possível calcular todos os movimentos possíveis das peças envolvidas no tabuleiro em um curto espaço de tempo. Nesse contexto já é possível inferir que as definições desse tipo de categoria de software, segundo Pflegger, podem implicar em possíveis manutenções, já que pode haver mudança na abstração inicial durante a construção.

A terceira e última categoria, segundo Pfleeger e Atlee (2010), leva em consideração a mudança no ambiente que será utilizado o contexto do software desenvolvido. Isso deve-se a base de criação desse software, o qual levou em consideração modelos com processos abstratos envolvidos como parametrização desse sistema. A medida que esses parâmetros tenham mudanças, será necessário rever todo o modelo considerado. Um

exemplo prático desse contexto é o comportamento das ações em um mercado na bolsa de valores, que constantemente sofre variações tanto de valores quanto de disponibilidade. Os fatores para isso ocorrer são diversos, sendo, principalmente, o cenário econômico atual de cada país e sua respectiva bolsa de valores. Portanto é possível inferir que a medida que essas abstrações são mais compreendidas, a necessidade de manutenção torna-se uma consequência inevitável.

Nota-se que segundo Ogheneovo (2013), os sistemas de softwares, independente de suas categorias, podem tornar-se tão complexos que naturalmente quando um software é alterado, mantido, evoluído e atualizado de forma constante podem gerar altos custos. Os custos dessas manutenções ou evoluções normalmente são bastante elevados em tempo de recursos financeiros. Havendo o fator de custos elevados e complexidades elevadas dos softwares, infere-se que seja interessante haver um possível investimento e planejamento em manutenção de software, já que o mesmo pode causar dificuldades futuras, sendo que junto com todas as implicações de seu crescimento se fará mais necessário que a robustez e confiabilidade se mantenham estáveis. Estima-se que existam mais de um bilhão de linhas de código em produção no mundo todo (OGHENEOVO, 2013), sendo que desse total, 80% é desestruturado e não documentado de forma adequada. O correto emprego e planejamento de manutenções desses softwares podem aliviar e contribuir para abaixar essa volumetria citada acima.

De acordo com Ogheneovo (2013), as características que influenciam em futuras etapas de manutenção e evolução de *software* são:

- Tamanho: fator bastante comum de acordo com o número de novas funcionalidades que vão se acumulando no sistema, dessa forma manter estável todas as funcionalidades é, sem dúvida, um grande desafio. Esse crescimento por consequência aumenta também sua complexidade, porém o autor refere-se exclusivamente a projetos de sistemas que possuam centenas de milhares de linhas de código, o que infere-se que em algum momento a manutenção se tornará inevitável, o que pode também ser inferido é que sistemas com tamanhos maiores serão mais difíceis e complicados do que sistemas menores em número de linhas de código;
- Idade: um *software* naturalmente deve transformar-se ao longo do tempo, contudo é preciso ressaltar que isso depende do quanto é investido em esforço para essa manutenção, visto que assim que um *software* aumenta em sua idade, o mesmo automaticamente deprecia, portanto para que seja possível fazer o sistema permanecer útil e confiável, há, indubitavelmente, a necessidade de mantê-lo;

- Velocidade de desenvolvimento: manutenção de *software* demanda um espaço de tempo para ser realizado. Isso é bastante visível porque nem sempre o primeiro profissional desenvolvedor será o mesmo que irá fazer a sua manutenção. Portanto, antes do profissional iniciar esse tipo de atividade, o mesmo precisará estudar as características do software, podendo isso ser feito por leitura de documentação, caso haja, ou fazer o estudo por conta própria de leitura do código. Por conseguinte levará algum tempo empregado bem como um esforço bastante elevado;
- Suportar novas tecnologias: para que os sistemas evoluam e possam suportar o emprego de novas tecnologias é preciso necessariamente realizar algum tipo de manutenção. Esse tipo de necessidade ocorre devido ao fato de ser preciso garantir que esse sistema atende aos requisitos dos usuários desse sistema, sendo inclusive ser possível que o *software* seja redesenhado e reprojetado; e
- Estrutura: a estrutura do sistema afeta necessariamente sua manutenção se o software não estiver com uma estrutura bem consolidada.

Portanto o software deve evoluir ao longo do tempo, isso significa que todo o produto de software continua a evoluir após a implantação do mesmo através dos esforços de manutenção (OGHENEOVO, 2013). Segundo ISO (1999) e Tripp et al. (1998) classificam manutenção de software:

- Manutenção corretiva: esse tipo de manutenção acontece sempre com ênfase para correção de erros e correção de *bugs*;
- Manutenção adaptativa: essa categoria envolve modificações ao sistema quando é necessário alterações a questões do ambiente, no sentido de infraestrutura por exemplo. Um exemplo prático é quando precisa efetuar uma manutenção de software em função de uma atualização do banco de dados utilizado pela aplicação, sistema operacional ou até mesmo a versão do compilador da linguagem utilizada. Esses tipos de manutenção normalmente são mantidas pelas próprias produtoras dessas ferramentas que auxiliam no desenvolvimento do seu próprio sistema;
- Manutenção de *performance*: esse tipo de manutenção engloba efetuar mudanças de *software* com o objetivo de alcançar resultados mais performáticos. Um exemplo prático é uma manutenção com base em melhora de desempenho com um tempo de resposta menor que antes de uma funcionalidade existente; e
- Manutenção preventiva: para manutenção preventiva, o *software* enfrenta modificações com o objetivo de detectar e corrigir possíveis falhas latentes antes mesmo

delas serem percebidas pelos usuários, uma vez que esse tipo de manutenção necessariamente ocorre pós entrega desse sistema.

Percebe-se, portanto que há distintas formas de categorização segundo suas complexidades, diferentes formas de motivações e influências e além disso tipos de manutenção de software. O foco desse trabalho será as manutenções corretivas, uma vez que no contexto da IF estudada acontecem incidentes e problemas que necessitam em sua grande maioria a necessidade desse tipo de manutenção para solução desses casos.

Ainda nesse sentido, os incidentes normalmente recebem manutenção corretiva em tempo de correção paliativa, isto é, contornar um comportamento inadequado de forma isolada para o incidente em si. A manutenção corretiva de fato do sistema normalmente acontecem com a resolução do que é chamado de problemas, o qual resolve o comportamento de forma fixa, sem apresentar novas ocorrências de anomalia. Os times de produto de desenvolvimento de software da IF normalmente utiliza bastante dessa categoria de manutenção, embora hajam outras necessidades como otimizações em tempo de performance, as quais normalmente já são interpretadas como evolução de software propriamente dita.

2.9 Evolução de Software

Evolução de *software* é o processo pelo qual os sistemas mudam de forma, adaptam-se as exigências de mercado e herdam características de programas preexistentes (OGHENEOVO, 2013). Já os autores Chapin et al. (2001), definem evolução de software como qualquer atividade de programação que possua como resultado a geração de uma nova versão do *software* a partir de uma versão anterior desse mesmo *software*. Pela visão de Chapin et al. (2001), a definição de evolução de *software* como aplicação das atividades de manutenção de *software* e processos que tenham como resultado uma nova versão do sistema com uma funcionalidade ou quaisquer propriedades alteradas no ponto de evista de experiência do cliente comparado com a versão anterior.

Logo, podemos concluir que evolução de *software* é um processo de mudança de software (OGHENEOVO, 2013). Esse processo de mudança, segundo Stammel et al. (2011), refere-se ao momento que o sistema é entregue até o momento da total desativação, também chamado de aposentadoria por Stammel, o qual necessariamente o software se transformou em um diferente e, usualmente, mais complexo que o anterior. Nota-se

que a evolução de *software* faz parte do ciclo de vida do sistema, podendo ser motivada e controlada a partir de decisões humanas ou até mesmo a partir de um julgamento do profissional desenvolvedor.

Com todos os conceitos apresentados, infere-se que a evolução de software é originado apenas se o desenvolvimento inicial e entrega foram bem sucedidas. No contexto de negócios, segundo, Ogheneovo (2013), o *software* estar sendo evoluído significa que o sistema possui alto grau de sucesso no mercado, a receita proveniente do *software* é alta também, assim como as demandas de usuários por mudanças também. Todos esses fatores citados contribuem para que a atmosfera do desenvolvimento seja vibrante e positiva na organização, o que consequentemente espera retornar o investimento em um produto melhor.

De acordo com Stammel et al. (2011) foram identificadas diversas razões para evolução de software:

- Novos requisitos para o sistema;
- Mudanças de ambiente; e
- Evolução do *stack* de tecnologia, principalmente no caso de um forte acoplamento do software, que pode resultar em uma coevolução de certas partes do sistema em si.

Dessa forma, segundo Stammel et al. (2011), ter capacidade de evoluir um *software* de forma ágil e confiável, isto é preservando a integridade da arquitetura de uma aplicação, é um desafio para quaisquer organizações, sendo, no contexto de IF, um desafio ainda maior, já que grande parte das evoluções de software são consequências de funcionalidades que nascem com uma necessidade maior de confiabilidade. Por consequência disso é nítido que uma evolução de *software* poderá se tornar bastante caro, no sentido de desgastes de *software*, também chamado de erosão de *software* por Niessink e Vliet (2000). Essa erosão é evidenciada, principalmente, pela volumosa necessidade de mudanças em sistemas legados, podendo significar uma perda de qualidade da estrutura de um sistema de software.

Essa grande quantidade de necessidade de mudanças, as quais levam a uma evolução de software são frutos de muito estudo e pesquisa pelos autores Lehman e Ramil (2000) e Lehman e Belady (1985). A partir do resultado desses estudos, foram inicialmente elaboradas as leis de Lehman para evolução de software. As primeiras cinco leis foram citadas no *paper* entitulado *Programs, Life Cycles, and Laws of software Evolu-*

tion" (LEHMAN; RAMIL, 2000). Após mais estudos, três leis adicionais foram propostas (LEHMAN; BELADY, 1985). Dessa forma os estudos que iniciaram com três leis no ano de 1980 desdobraram-se em oito leis de evolução de *software* por Lehman, Belady e seus colegas. Essas leis estão descritas abaixo:

- Evolução contínua: A manutenção do sistema é um processo inevitável segundo essa lei. Isso justifica pelo fato que a partir da mudança do ambiente do sistemas, automaticamente novos requisitos surgem e por consequência o sistema deve ser alterado. Dessa forma, qualquer *software* utilizado se mudará ou se tornará mais usual nesse novo ambiente. Logo o *software* deve sofrer mudanças contínuas ou, progressivamente, tornará-se menos útil;
- Aumento de complexidade: como qualquer sistema passa por mudança, a tendência é que acabem ficando mais complexos, o que pode causar uma desestruturação cada vez maior. Isso será inevitável, a menos que algum trabalho seja realizado para amenizar essa situação;
- Auto-regulação de evolução de software: essa lei especula que a evolução é um processo auto-regulado, com isso fica a cargo de cada organização estabelecer normas e verificações que sejam capazes de lidar com as dificuldades do processo, podendo contribuir para que as respostas dessas dificuldades sejam facilitadas;
- Estabilidade organizacional: essa lei estabelece que ao longo do ciclo de vida de um *software*, as taxas de desenvolvimento sejam constantes ou se aproximem muito de pouca variação. É sugerido que todos esses processos sejam independentes de mudanças de profissionais e movimentações, para que não haja dificuldades em efetuar as evoluções necessárias no sistema em si;
- Conservação de familiaridade: a quinta lei de evolução refere-se ao conceito de conservação da familiaridade das mudanças necessárias a todos envolvidos, principalmente nos objetivos das alterações. A volumetria de mudanças necessárias é proporcional a dificuldade em manter todos envolvidos engajados e cientes dessas alterações;
- Crescimento contínuo: a sexta lei refere-se ao crescimento contínuo do software. A medida que o software evolui, acerca de novas funcionalidades, há a necessidade de manter a satisfação do usuário que utilize esse software;
- Declínio de qualidade: a sétima lei apresenta a característica de qualidade decrescente do software a medida que ele evoluí. Normalmente esse fato acontece em

função do ambiente imprevisível o qual o software está contido. Esse ambiente gera infinitas possibilidades de mudanças que certamente irão divergir da ideia inicial do *software*, e isso irá expor a suscetibilidade do software a fatores externos; e

- Sistema de *feedback*: a oitava e última lei refere-se ao sistema de retorno, que sugere a existência de um sistema constante de *feedbacks* dos usuários, causando sugestões de evoluções futuras. Dessa é possível inferir que um sistema de software possui um ciclo de impressões positivas e negativas dos usuários, sendo importante manter essas impressões próximos das próximas evoluções a serem realizadas.

No cenário da IF e do público alvo dessa pesquisa, haverá o foco na primeira, segunda, quarta e oitava lei de evolução de software, já que são constantes as necessidades de melhorias, para uma evolução contínua, de criação de novas funcionalidades, aumentando naturalmente a complexidade do sistema, de estabilidade, já que é importante manter o usuário com impressões positivas sobre as funcionalidades atuais e futuras, de *feedbacks*, pois é nítido que atualmente sejam cada vez mais necessários haver retornos da usabilidade do sistema pelos usuários, sendo esse um atributo necessário para tomada de decisões acerca das próximas funcionalidades a serem implementadas.

Ainda nesse sentido, para fins de direcionamento de conceitos, a pesquisa desse trabalho terá como conceito de evolução de software toda e qualquer implementação de novas funcionalidades ou então melhorias acerca das funcionalidades atuais.

3 TRABALHOS RELACIONADOS

Neste Capítulo, são apresentados os principais trabalhos relacionados. Os trabalhos estão divididos em três seções, onde a primeira destaca os conceitos de governança de TI, através do ITIL, COBIT, a segunda através uso do Scrum com PMBOK para gerenciamento de projetos de *software* e a terceira os benefícios do uso do Kanban para gerenciamento de projetos de manutenção de *software*.

3.1 Aplicação da governança de TI

Dentro da IF, ambiente o qual os profissionais estão alocados, é muito consolidado a figura da governança de TI, dessa forma a imensa maioria dos trabalhos relacionados e mencionados relatam como se caracterizam os conceitos de governança de TI e quais são os principais *frameworks* utilizados, nesse contexto são citados os conceitos do COBIT e ITIL através dos artigos dos autores Mangalaraj, Singh e Taneja (2014), Barbosa e Lima (2011) e também o livro de autoria de Adams (2009).

Esses conceitos são utilizados para melhor compreensão dos cenários que relatamos nessa pesquisa perante suporte técnico a sistemas, representados pela atuação em incidentes e problemas, sendo esses um dos grandes ofensores e causadores de manutenção do *software* do sistema produto. O artigo FEBRABAN (Federação Brasileira de Bancos, 2022) reforça que cada vez mais são requisitadas disponibilidades de autoserviços pelos grandes bancos, fazendo com que a robustez de infraestrutura sejam bastante elevadas para suas áreas de tecnologia. Por se tratar de serviços críticos, o papel da governança de TI, principalmente na gestão de continuidade de serviços (BARBOSA; LIMA, 2011) e (ADAMS, 2009) é bastante explorado, uma vez que hoje em dia falhas ou indisponibilidade para quaisquer funcionalidades acarretam na falta de confiança do cliente final.

Também podemos citar o quanto a tecnologia de informação influenciou na evolução dos sistemas bancários brasileiros, sendo esse elemento de estudo do artigo (OLIVEIRA; MALAGOLLI, 2016), e o comércio eletrônico estudado pelo artigo dos autores (ALBERTIN, 1999), sendo que esses fatores apresentados representem algumas das principais fontes causadores de necessidade e como explica o crescimento dos conceitos de governança de TI, o qual está evoluíndo de maneira tão rápida nos últimos anos dentro do mercado como um todo, principalmente dentro de IF.

3.2 PMBOK e Scrum com projetos de software

Também são mencionados os trabalhos em gerenciamento de projetos, fundamentando os conceitos e as vantagens do modelo apresentado pela PMI (PMBOK, 2012), e relacionando diretamente com projetos de *software*, principalmente na gerência de projetos orientados a produto, evidenciado pelo artigo de autoria do Moraes (2012) e livro do autor Cruz Cruz (2013). Apesar dos conceitos do PMBOK possuírem um alto grau de uso no mercado como um todo, é evidenciado que ele possui debilidades, as quais são evidenciadas pelo livro de Fabio Cruz 2013, que destaca o uso das metodologias ágeis para auxiliar, principalmente, na gerência de projetos de software uma forma de agregar mais valor com seus entregáveis ao longo do projeto, uma vez que o modelo tradicional de gerenciamento de projetos idealiza um produto entregável apenas no final do projeto. Essa debilidade destacada anteriormente junto com a carência de inovação tecnológica e necessidade de rápida de resposta aos avanços de recursos tecnológicos disponíveis, contribuem para que as empresas do mercado, especialmente as IF, tenham suas demandas por adaptações de seus serviços de forma mais rápida possível, já que é necessário cada vez mais estar preparado para as evoluções que os clientes esperam. Nesse sentido, é relacionado o Manifesto Ágil, (Agile Manifesto, 2001), que apresentou as metodologias ágeis, as quais introduzem novas formas de haver maior índice de sucesso nos projetos. Dentro do público da IF estudada, não é diferente, visto que são utilizados o *Scrum* e o *Kanban* de forma isolados ou combinados.

Os artigos dos autores Schwaber e Sutherland (2010) apresentam os conceitos e aplicabilidade do *Scrum*, e também sua principal característica, isto é, tornar o processo de desenvolvimento de software mais flexível, principalmente em ambientes que são necessários um grande e elevado número de mudanças. São apresentados todos eventos *Scrum*, havendo destaque para o ciclo (*sprint*), o qual tem como conceito haver um entregável ao final desse tempo transcorrido, fazendo ligação direta com a debilidade citada anteriormente e já havendo atitude de resposta rápida as necessidades dos clientes que demandam mudanças.

Na IF o emprego dessa metodologia ágil junto com os conceitos apresentados do PMBOK são aplicados com bastante afinco e qualidade, já que o cenário apresentado pela teoria aproxima-se da realidade prática, principalmente na necessidade de mudanças, as quais desdobram-se em requerimentos de novas funcionalidades ou melhorias em seus sistemas existentes.

3.3 Kanban e manutenção de software

No artigo dos autores Silva, Santos e Neto (2012) são apresentados os conceitos do uso da metologia ágil Kanban com um projeto que tem como principalidade a manutenção de software. O artigo faz uma comparação imediata da metodologia ágil *Scrum*, Kanban e XP (eXtreme Programming). O motivo dessa comparação é justamente evidenciar que o Kanban parece ser muito mais adaptativo do que as outras metodologias.

São apresentados o contexto inicial de uma empresa de tecnologia, a qual tem uma problemática central em uma equipe: o constante número de itens extraordinários que entram no ciclo de desenvolvimento sem ter sido planejado. Destacam os sentimentos dos profissionais envolvidos, principalmente com a frustração em não conseguir, apesar do esforço empregado, entregar o que fora combinado na reunião de planejamento. Dessa forma é demonstrado o emprego da metodologia Kanban, tendo como resultado imediato a não existência de um ciclo pré-existente, como no *Scrum*, já havendo uma sensação de acolhimento maior pela equipe em função da flexibilidade demonstrada. Também são apontadas sugestões de classificação dos itens que estão na coluna "a fazer"de modo que sejam mais visível a prioridade no quadro de trabalho, além de apontar uma estimativa do esforço em uma análise embrionária do contexto e necessidade de solução desse item em si, sendo possível classificar entre tamanho P, M e G, chamada de *T-Shirtsizing* (KNIBERG, 2016):

- Tamanho P: uma atividade que levaria menos de um dia para realização;
- Tamanho M: como uma atividade que levaria mais de um dia e menos de uma semana para finalização; e
- Tamanho G: atividade com duração estimada de igual ou superior a uma semana.

Após a implantação desse cenário foi feito o acompanhamento pelos autores da primeira semana a sexta semana para fins de análise dos resultados, sendo destacado o decréscimo linear dos dias de *leadtime* dos itens até mesmo o número de atividades entregues pela equipe. A tabela 3.1 abaixo demonstra o resultado:

A partir desse resultado apresentado, é possível verificar que houve uma melhoria significativa no número total de itens entregues por iteração, assim como a diminuição do *leadtime*, embora houvesse acrescido o número de um profissional desenvolvedor nesse meio tempo.

O cerne apresentado pelos autores (SILVA; SANTOS; NETO, 2012) é evidenciar

Tabela 3.1 – Tabela com iterações ocorridas com o Kanban

	Iteração 1	Iteração 6
Quantidade de desenvolvedores na equipe	4	5
Quantidade média de tarefas entregues (por desenvolvedor)	6	8
Quantidade média de tarefas entregues (equipe)	24	40
WIP (Work in Progress)	10	10
Leadtime (dias)	15	6

Fonte: (SILVA; SANTOS; NETO, 2012)

que algumas metodologias ágeis para gerência de projetos de *software* podem ser mais adequadas que as outras, dependendo do contexto, sendo possível inferir que o Kanban tenha muito mais afinidade com projetos de manutenção de *software* que o *Scrum* por exemplo.

3.4 Considerações finais

Neste capítulo, foram apresentados os trabalhos relacionados e foi explorado o que esses trabalhos discutem e como se relacionam com o trabalho construído. Com esses conceitos apresentados anteriormente, foi possível buscar relações palpáveis do que é buscado apresentar por esse trabalho, uma vez que, no cenário da IF, são utilizadas ambas metodologias ágeis, combinadas ou não combinadas, para gerência de projetos de *software* com manutenções e evoluções do sistema de forma concomitante, sendo um desafio lidar com diversas priorizações e tipos de itens surgindo ao longo do tempo.

4 METODOLOGIA E PROPOSTA

Neste capítulo são apresentadas a proposta e metodologia aplicadas para evolução do trabalho, que está dividida em duas seções. A primeira explica a metodologia, através do tipo de pesquisa empregada com suas características. A segunda apresenta o conteúdo da pesquisa, que se divide em dois blocos. O primeiro é mais teórico e o segundo mais técnico, utilizando questões mais enviezadas com o cotidiano do profissional público alvo da pesquisa, buscando explorar como os conceitos teóricos se aplicam no dia a dia, já para a parte técnica explorando, a partir das questões, como são classificados os contextos apresentados para que, a partir das respostas possa ser possível analisar e relacionar com o objetivo da pesquisa.

4.1 Metodologia

Para desenvolvimento desse trabalho, é realizada uma pesquisa com abordagem quantitativa (FONSECA, 2002), de modo a captar um grande número de respostas de forma que consiga-se quantificar os resultados, sendo possível obter o entendimento da realidade do ambiente. A pesquisa tem objetivo de ser exploratória, isto é, tem como objetivo se aproximar e tornar familiar o problema exposto, de forma que o mesmo fique mais explícito (SANTOS, 2011). Quanto aos procedimentos da pesquisa, é utilizado o método do tipo *Survey* (GERHARDT; SILVEIRA, 2009), que engloba na criação de um questionário para um público alvo. Esse tipo de procedimento busca, além da obtenção de dados, fazer com que tenhamos mais informações, ações ou opiniões sobre um grupo de pessoas, sendo apropriada quando (GERHARDT; SILVEIRA, 2009) :

- O foco do interesse é saber sobre "o que está acontecendo"ou "como e por que isso está acontecendo";
- Não é possível controlar variáveis dependentes e independentes;
- O ambiente natural é a melhor situação para estudo do ambiente; e
- O objeto da pesquisa ocorre no presente ou no passado recente.

No contexto desse trabalho, o primeiro item aproxima-se com a realidade do ambiente que esse trabalho explorará, já que apesar da gama teórica estar bastante consolidada, pode ser que através da pesquisa tenhamos resultados de modo que entendamos de fato como será refletido a realidade prática através dos profissionais que serão respondentes

da pesquisa.

Quanto ao número de momentos de coletas de dados, esse trabalho tem apenas um único momento para coleta de dados, aproximando do conceito de corte-transversal (GERHARDT; SILVEIRA, 2009). Essa escolha aproxima-se da característica e objetivo da pesquisa, uma vez que é pretendido descrever e analisar o estado de uma ou mais variáveis em apenas um único momento.

No quesito amostragem, a pesquisa tem um público alvo específico, nesse caso profissionais de TI em papéis de desenvolvedores de *software* e analistas de qualidade, também conhecido como testadores de *software* ou trazendo para um conceito mais atual de mercado como engenheiro de qualidade (*quality engineer*). Vale ressaltar que a pesquisa está direcionada aos profissionais alocados em uma IF apenas, sendo possível de analisar apenas o ambiente dessa IF em questão.

As alternativas de respostas dessa pesquisa estão sendo representadas na escala Likert. A escala Likert, segundo Bermudes et al. (2016) foi criada pelo educador e psicólogo Rensis Likert em 1932, quando recebeu seu Ph.D. em psicologia pela Universidade de Columbia. No seu trabalho foram apresentados um levantamento usando uma escala de um a cinco pontos, propando um meio, através da pesquisa, de medir atitudes, mostrando já na época ser um método eficaz de coletar informações em comparação aos métodos que concorrem com essa escala.

Segundo (AGUIAR; CORREIA; CAMPOS, 2011) escalas Likert são uma das escalas com maior ênfase em autorrelato, possuindo perguntas que tem como suas alternativas de respostas opções, normalmente cinco, como:

- Concordo muito;
- Concordo pouco;
- Neutro ou indiferente;
- Discordo pouco; e
- Discordo muito.

Essas possibilidades de respostas faz com que seja possível verificar o grau de intensidade do respondente, fazendo com que seja possível analisar de uma forma mais assertiva o conteúdo das respostas. Apesar desse padrão, é comum as alternativas se adaptarem as perguntas realizadas, podendo, portanto, haver maior flexibilidade na forma como é pretendido buscar alternativas para as perguntas realizadas. A mesma flexibilidade aplica-se a quantidade de alternativas, é comum haver uma infinidade de trabalhos

com número de alternativas entre um e sete alternatvivas, porém com cinco sendo a mais utilizada pela ampla maioria.

Ainda segundo (AGUIAR; CORREIA; CAMPOS, 2011), visando a análise dos resultados, é necessário atribuir valores para cada uma das alternativas, começando em zero para o item neutro e aumentando em 1 para cada item acima ou abaixo, para que seja possível em seguida ser obtido a média dos valores totais avaliados, bem como desvios padrões e etc.

4.2 Característica da Pesquisa

Nessa seção é apresentada a característica da pesquisa realizada, ela se subdivide em duas subseções, a primeira com a parte teórica, a qual busca explorar no questionário os conceitos abordados no 2 em relação ao cotidiano do público alvo. A segunda apresenta a parte técnica, a qual terá perguntas a partir de um contexto previamente apresentado junto com algumas funcionalidades implementadas, provocando, na prática, a percepção do profissional em relação aos conceitos teóricos explorados em relação aos contextos apresentados.

4.2.1 Parte teórica

A primeira seção do bloco de perguntas é baseado nos conceitos empregados no capítulo 2, relacionando-se diretamente com o dia a dia do profissional na IF alvo.

O público alvo dessa pesquisa está localizado em um time de desenvolvimento que possui em sua grande maioria a estrutura de time ágil Scrum, isto é, com um número fixo de membros, cada um com suas respectivas responsabilidades de acordo com seu escopo, e alocada para objetivos ligados diretamente a desenvolvimento de uma família de produto financeiro, por exemplo, empréstimo, atuando especificamente com crédito pessoal. No interior de cada time, há diferentes nichos de atuação em tempo de desenvolvimento, havendo a figura de um PO (*product owner*) ou PM (*product manager)*, os quais são responsáveis pela condução dos itens a serem priorizados pela sua equipe, um membro responsável por cuidar especificamente em manter a metodologia ágil empregada com qualidade, o qual pode ser referenciado no mercado como um profissional chamado de "agilista"ou *"scrum master"*, e membros diretamente ligados ao processo de desenvol-

vimento, podendo ser desenvolvedores, analistas de sistemas e analistas de qualidade.

Normalmente, durante reuniões de planejamento, são apresentados os itens a serem analisados e trabalhados pela equipe, sendo que esses podem ser desde novas funcionalidades até mesmo itens de manutenção das funcionalidades ligadas ao produto que a equipe está alocada. Itens que surgem ao longo do ciclo são considerados extraordinários e possuem natureza diversa, desde um incidente, que causou uma interrupção ou erro em um dos sistemas, até mesmo requerimentos de melhorias ou novas funcionalidades em função das necessidades diversas que podem surgir no ambiente.

Diante de todo esse escopo, o questionário apresenta perguntas que buscam atrelar essa rotina vivida pelos profissionais, buscando entender, por exemplo, onde que estão sendo colocados os esforços, seja em manutenção de *software* ou evolução de *software*, ou então buscar verificar se estão sendo seguidos os preceitos das metodologias ágeis e até mesmo entender a visibilidade das entregas desses itens sejam eles planejados ou não planejados (extraordinários).

4.2.2 Parte técnica

Para a segunda seção do questionário foi necessário a implementação de algumas funcionalidades modelos utilizadas na IF estudada. Em função da não autorização pela IF estudada do uso do código fonte das funcionalidades dos sistemas produtos para uso direto no questionário, foi necessário a implementação das mesmas funcionalidades selecionadas na forma de pseudocódigo, com isso foi preciso reescrever as funcionalidades do códigos de modo que fosse possível realizar as perguntas servindo de exemplos práticos com ênfase nos tipos de itens, sejam de manutenção ou evolução de *software*.

Para isso foi utilizada a linguagem de programação *Java*, utilizando da ferramenta de edição de código *Visual Studio Code* (VS-Code). A escolha dessa linguagem é justificada pela facilidade de implementação de funcionalidades que funcionam como *back-end*, isto é, que representam as *features* em sua forma mais robusta sendo uma funcionalidade que têm todo seu processamento ocorrendo sem que o usuário saiba da lógica sendo aplicada.

Para que houvesse material técnico para perguntas nos questionários foram realizadas a codificação de quatro funcionalidades ligadas diretamente a produto de crédito da IF:

1. Operacionalização de propostas a partir da geração dos eventos: atualmente o sistema de construção de propostas de crédito possui uma arquitetura de solução orientada a eventos, isto é, a medida que uma proposta, representada por um evento, é criada e enviada para uma fila, de modo que, periodicamente, é realizado a leitura desses eventos pelos consumidores, sendo muitas vezes representados pelos motores de cálculo de concessão de crédito, o qual pode retornar um valor máximo de crédito disponível para o cliente da proposta enviada originalmente;
2. Implementação de métodos de liquidação da dívida de um produto específico: no contexto de sistema de crédito, após um período longo de inadimplência, é necessário a atualização para o status de "Prejuízo", visto que a IF não tem mais receitas e nem despesas com essa operação de crédito concedida. Porém, para o cliente, é possível a quitação dessa dívida de modo que o mesmo possa ficar sem débitos ativos pendentes com a IF. Para isso foi implementado dois métodos no sistema produto, sendo o primeiro responsável pela passagem da operação para prejuízo e o segundo responsável pela liquidação de saída dessa situação;
3. Métodos com lógica duplicada para uma funcionalidade: um cenário de necessidade de cancelamento de um produto de crédito, por exemplo cartão de crédito ou cheque-especial, é possível que seja realizado o cancelamento a partir da solicitação do cliente ou então a partir de uma renegociação do produto, sendo esse exclusivo para clientes em situação de inadimplência. Para o cancelamento do produto de forma direta, basta que o cliente tenha saldo em sua conta corrente de forma que seja ao menos equivalente ao valor total da dívida. Para renegociação do produto, além da condição de inadimplência do cliente, o valor total da dívida é dado como um valor de crédito a ser realizado em sua conta, sendo um novo produto contratado que poderá ser pago em parcelas mensais. É destacado que nesse caso a consolidação da dívida ocorre de forma igual nos dois métodos; e
4. Funcões com tipos de dados inadequados: em uma busca de produtos ligados a uma conta corrente foi implementado uma lógica para busca baseado no número da conta fazendo um tratamento para o dado, é destacado que conta corrente possui um tipo de dado tipo inteiro, e na função foi implementada de forma proposital como *string*, sendo necessárias tratativas nativas da linguagem de programação para adequar a busca pelo produto.

Essas implementações realizadas foram feitas de modo que não estivessem no seu modo ideal, buscando exemplificar situações em que são necessárias mudanças no

estado original do sistema produto, especificamente as funcionalidades número 2 e 4 relacionam-se diretamente a manutenção de software por iminência de incidentes. Já as funcionalidades 1 e 3 representam evolução de software, uma vez que são *features* que estão inapropriadas, seja por não haver um local único para consolidação das dívidas ou por não representar operações com tipos de dados corretos, necessitando de refatoração de software por exemplo.

Lembrando que para fins de formalização, manutenção de software é tido como quaisquer erros ou anomalias causadas no sistema onde a correção definitiva necessita de mudança do estado original do software. Já para evolução de software considera-se qualquer tipo de melhoria no *software* representadas por novas funcionalidades ou refatoração de funcionalidades por exemplo.

5 ANÁLISE DOS RESULTADOS

Este capítulo descreve a análise e a apresentação dos resultados do *Survey* aplicado na IF estudada, além de uma pequena conclusão a respeito dos resultados. A Seção 5.1 apresenta a consolidação dos dados dos respondentes. A Seção 5.2 demonstra os resultados das análises com breves inferências dos cenários vividos pelo cotidiano. Finalmente, a Seção 5.3 descreve uma conclusão sobres os resultados.

5.1 Consolidação dos dados

A pesquisa foi enviada para oitenta membros da IF alvo, esses membros atuam em papéis de desenvolvedores de *software* e analistas de qualidade (testadores). Foram obtidas vinte e seis respostas, havendo, portanto, 32,5% de aderência ao questionário na relação de respostas obtidas dividida pelo número de enviados.

Com a coleta de resultados, foi necessário realizar uma modelagem nos dados, de forma que fosse possível utilizar os conceitos da escala *Likert* para análise dos resultados. Primeiramente foi criado uma tabela, as quais as colunas indicam as alternativas e o número respectivo de vezes que essa alternativa foi escolhida, e as linhas indicam o número da questão. Após essa tabulação, foram criadas novas colunas para apoio na construção de gráficos.

Essas colunas extras obedeceram as seguintes regras:

- Coluna extra 1: somatório de todas as respostas da respectiva linha;
- Coluna extra 2: criação do neutro negativo, isto é, transformar o número de respostas da alternativa neutra (nem concordo e nem discordo) e atribuir ela mesmo com sinal invertido;
- Coluna extra 3: criação da coluna para alternativa "discordo parcialmente"e atribuir o valor de respostas da respectiva linha com o sinal invertido;
- Coluna extra 4: criação da coluna para alternativa "discordo totalmente"e atribuir o valor de respostas da respectiva linha com o sinal invertido;
- Coluna extra 5: criação do neutro positivo, atribuindo o valor de respostas da alternativa neutra sem alteração do sinal;
- Coluna extra 6: criação da coluna para alternativa "concordo parcialmente"e atribuir a ela o valor de respostas da respectiva linha sem alteração do sinal; e

- Coluna extra 7: criação da coluna para alternativa "concordo totalmente"e atribuir a ela o valor de respostas da respectiva linha sem alteração do sinal.

Com essa modelagem realizada, foi possível obter as porcentagens representativas de cada alternativa, sendo possível medir de forma sintética os valores representados e também expor isso em gráficos para melhor visualização numérica e também facilitação da análise dos resultados.

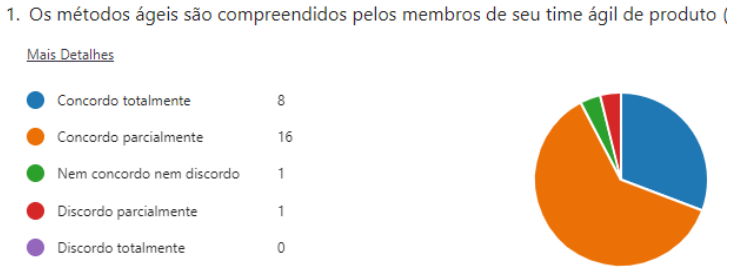
5.2 Resultados

Nesta seção, são detalhados os resultados das perguntas realizadas seguidos de algumas inferências para demonstrar a realidade do cotidiano vivido pelos profissionais, de forma que contribua para o tema objetivo desse trabalho.

Como mencionado no capítulo anterior, a primeira parte do questionário apresentava questões que representam o conflito entre os nichos de desenvolvimento de novas *features* e necessidades de manutenção do sistema produto, os conceitos das metodologias ágeis e visibilidade dessas entregas frente ao conceito de valor da entrega, isto é se a entrega desses itens fazem com que haja geração de valor para os clientes, sejam internos ou externos.

Com o intuito de explorar os conceitos das metodologias ágeis presentes no cotidiano dos profissionais, foi realizado a seguinte pergunta: "Os métodos ágeis são compreendidos pelos membros de seu time ágil de produto", representada na Figura 5.1. Percebe-se que a partir do gráfico pouco mais de 90% dos respondentes votaram com intensidade positiva, porém apenas 31% afirmaram estar totalmente de acordo, o que pode significar que nem todos os membros das equipes compreendem ou tem nível de confiança para afirmar a intenção da resposta de forma mais elevada.

Figura 5.1 – Survey - Questão 1

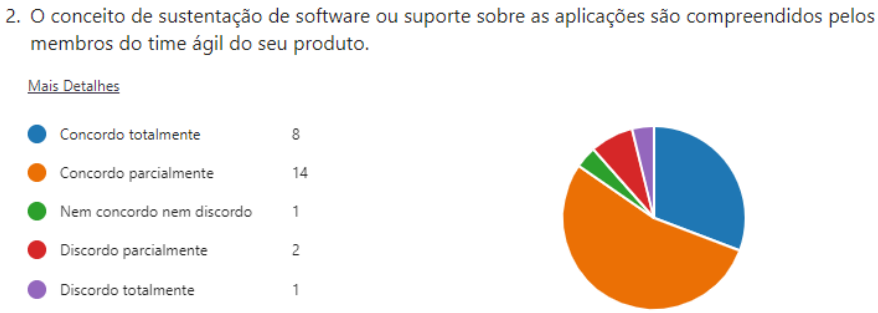


Fonte: Próprio Autor

Para realizar uma comparação direta com um outro cenário, presente também no ambiente dos profissionais, foi buscado entender se os conceitos de sustentação são compreendidos pelos respondentes. A pergunta realizada foi: "O conceito de sustentação de software ou suporte sobre as aplicações são compreendidos pelos membros do time ágil do seu produto", representada na Figura 5.2.

Vale observar que essa aceitação refletida para os métodos ágeis, também acontece para o nicho de sustentação de *software*, já que 85% dos respondentes avaliaram como positivo os processos envolvendo esse nicho, embora também seja possível inferir que haja alguma fragilidade ou não entendimento dos processos, já que nem todos os membros deram a intensidade máxima para a resposta.

Figura 5.2 – Survey - Questão 2

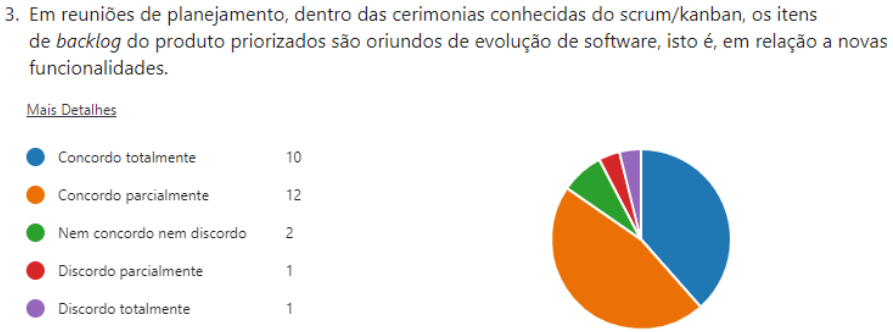


Fonte: Próprio Autor

Entretanto, a partir das próximas questões, foi proposto enunciados com intuito comparativo envolvendo reuniões de planejamento do *Scrum* com priorizações de itens do *backlog* para manutenção ou evolução de *software*, buscando trazer se há o conflito nesse momento. Para isso foram realizadas as seguintes perguntas: "Em reuniões de planejamento, dentro das cerimonias conhecidas do scrum/kanban, os itens de backlog do produto priorizados são oriundos de evolução de software, isto é, em relação a novas funcionalidades"e "Em reuniões de planejamento, dentro das cerimonias conhecidas do scrum/kanban, os itens de backlog do produto priorizados são oriundos de manutenção de software, isto é corrigir funcionalidades existentes", representadas nas Figuras 5.3 e 5.4, as quais trouxeram resultados que mostram que ambos os tipos de itens são apresentados e priorizados nas reuniões de planejamento, havendo maioria de concordância em ambos cenários, porém ficou um pouco explícito que nem sempre em reuniões de planejamentos os itens de manutenção de softwares são priorizados e apresentados, trazendo uma diferença na prioridade dos dois nichos.

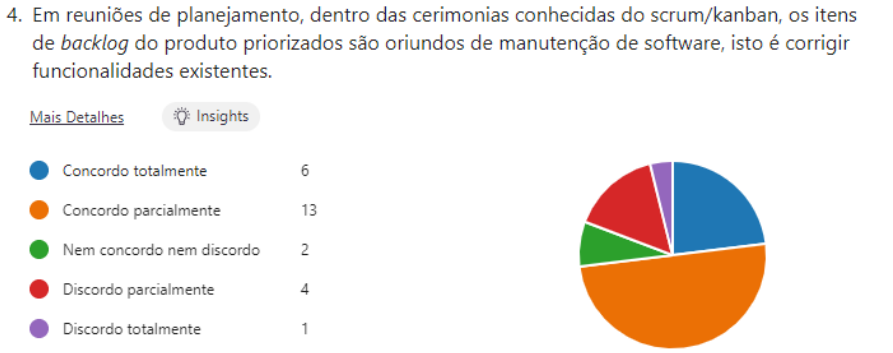
Em relação aos momentos diferentes das reuniões de planejamento que são ne-

Figura 5.3 – Survey - Questão 3



Fonte: Próprio Autor

Figura 5.4 – Survey - Questão 4

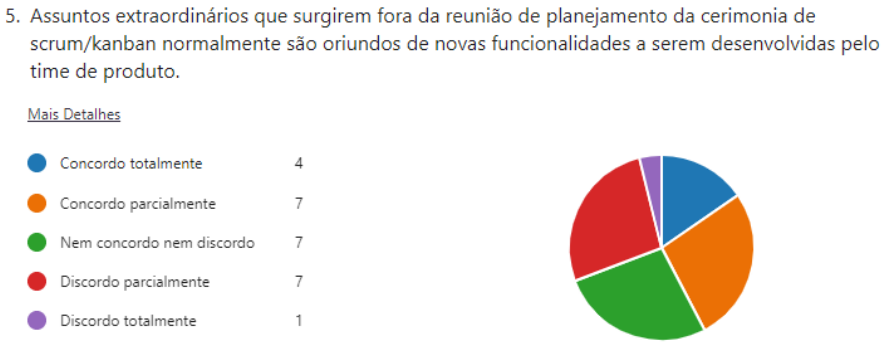


Fonte: Próprio Autor

cessários ingressos de itens extraordinários, foi buscado compreender qual tipo é mais comum de ser abordado. Nesse quesito, de forma intuitiva, é perceptível que incidentes ou erros do sistema produto sejam mais comum de surgirem, uma vez que o *software* encontra-se em uso pelos clientes externos e internos da IF. Através desse contexto, foi explorado nas perguntas quais tipos itens surgem de forma extraordinárias, isto é se são demandas de evolução ou manutenção, as perguntas 5 e 6 realizadas foram: "Assuntos extraordinários que surgirem fora da reunião de planejamento da cerimonia de scrum/kanban normalmente são oriundos de novas funcionalidades a serem desenvolvidas pelo time de produto"e "Assuntos extraordinários que surgirem fora da reunião de planejamento da cerimonia de scrum/kanban normalmente são oriundos de manutenção de software a serem analisadas pelo time de produto"respectivamente, representadas nas Figuras 5.5 e 5.6, as quais as respostas demonstram que naturalmente, os cenários que contém itens de manutenção (questão 6) são mais comuns, visto o alto grau de concordância dos respondentes. Já para os itens de evolução (questão 5), naturalmente, não são tão comuns surgirem de forma extraordinária, uma vez que novas funcionalidades normalmente são apresentadas em reuniões de planejamento, representadas por histórias de usuários em sua grande mai-

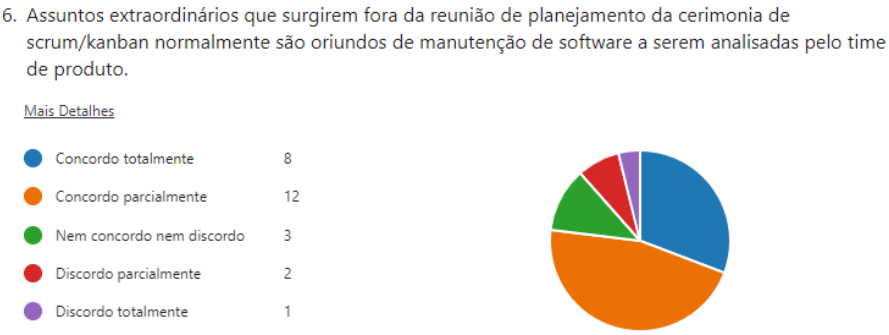
oria. Contudo, as respostas mostraram uma tendência de equilíbrio para a pergunta 5, representada na Figura 5.5, uma vez que há um grau de discordância leve de aproximadamente 27% dos respondentes. Infere-se que para certos times ágeis ocorram abordagem de itens de evolução de forma extraordinária, isto é, fora da cerimônia de planejamento.

Figura 5.5 – Survey - Questão 5



Fonte: Próprio Autor

Figura 5.6 – Survey - Questão 6



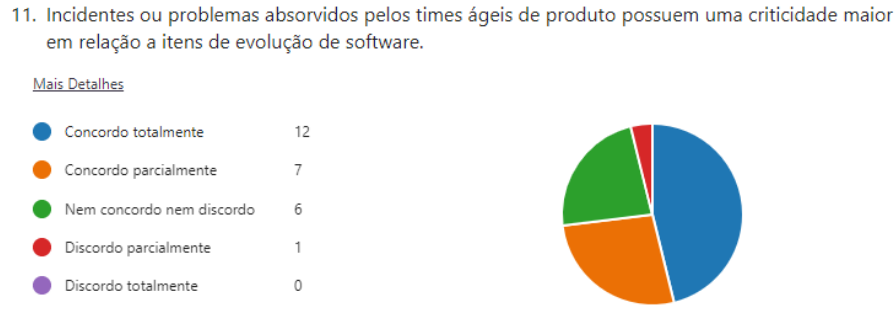
Fonte: Próprio Autor

Foi buscado compreender o senso de percepção e criticidade dos respondentes em relação a uma resolução de um item do tipo incidente ou problema ser entregue o quanto antes em relação a uma história de usuário, representando uma nova *feature*, logo uma evolução de software. Normalmente, um incidente ou problema teriam alta prioridade, uma vez que é oriundo de um comportamento inadequado no sistema produto em um ambiente produtivo da IF. Nesse contexto, foi realizado a pergunta (número 11): "Incidentes ou problemas absorvidos pelos times ágeis de produto possuem uma criticidade maior em relação a itens de evolução de software", representada na Figura 5.7, as quais respostam indicam que os respondentes tiveram um alto grau de concordância com a afirmativa, atingindo 73% em intenção positiva da resposta. No entanto, 12% dos respondentes tiveram posição neutra quanto a afirmação, o que indica que no entendimento dos profissionais,

essa prioridade nem sempre é veridica, buscando mostrar que há subjetividade ou necessidade de mais informações para que seja possível julgar, ou então de fato cada profissional tem opinião indiferente quanto essa comparação.

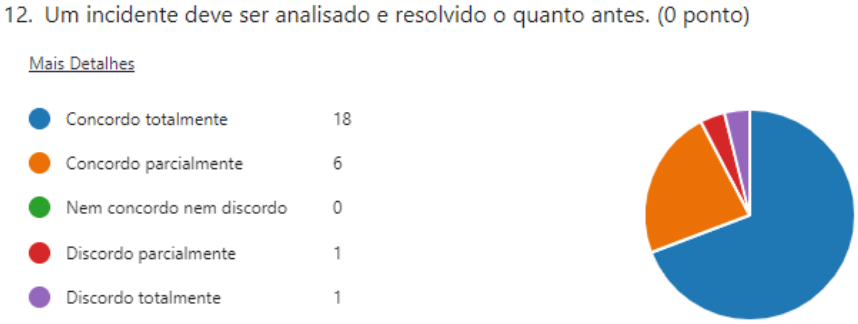
Ainda nesse contexto, foi realizada a seguinte pergunta (número 12): "Um incidente deve ser analisado e resolvido o quanto antes", visando indagar o público alvo para compreensão se um incidente deve ser priorizado e resolvido o quanto antes, dessa forma é esperado que pela definição o resultado seja positivo, isto é, que esse tipo de item deve ser resolvido o quanto antes. A Figura 5.8 nos traz que em sua elevada maioria os respondentes reagiram de acordo com o esperado.

Figura 5.7 – Survey - Questão 11



Fonte: Próprio Autor

Figura 5.8 – Survey - Questão 12

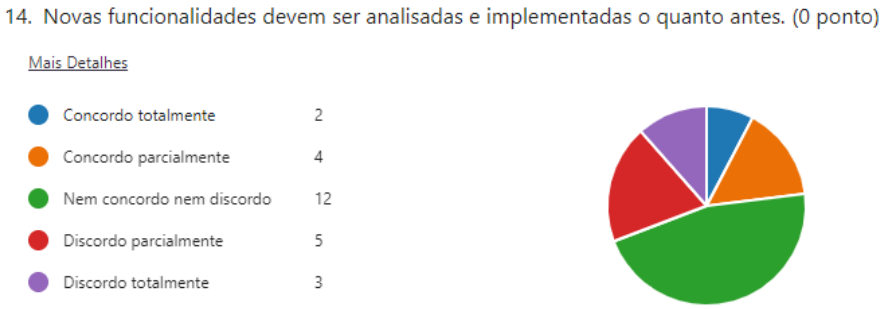


Fonte: Próprio Autor

Por outro lado, a pergunta número quatorze, a qual foi: "Novas funcionalidades devem ser analisadas e implementadas o quanto antes", buscou compreender se um item de evolução de software deveria ser analisada e implementada o quanto antes. Naturalmente, é esperado que os respondentes tenham uma posição indiferente quanto a isso, uma vez que não há razões para priorizações e tração elevada em comparação a um erro de sistema que precisa de análise mais rápida. A Figura 5.9 demonstrou na prática o esperado, um alto grau de neutralidade, porém é possível destacar um grande número

de discordâncias, sendo possível inferir que os profissionais consideram importantes a entrega de novas funcionalidades do sistema produto. Uma das razões possíveis é justamente tornar o produto mais competitivo junto ao mercado, sendo importante a entrega da nova funcionalidade no sistema para agregar nesse contexto.

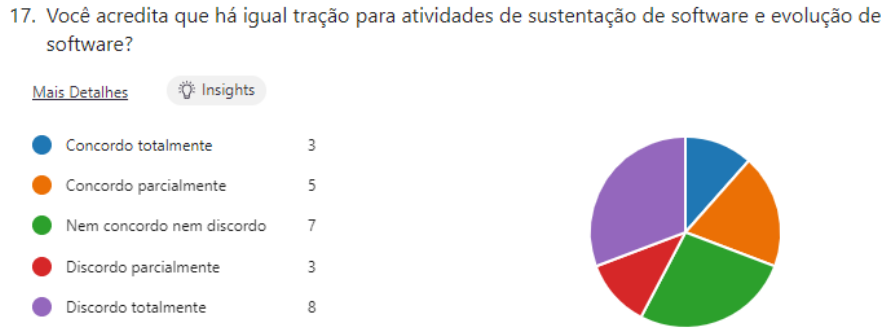
Figura 5.9 – Survey - Questão 14



Fonte: Próprio Autor

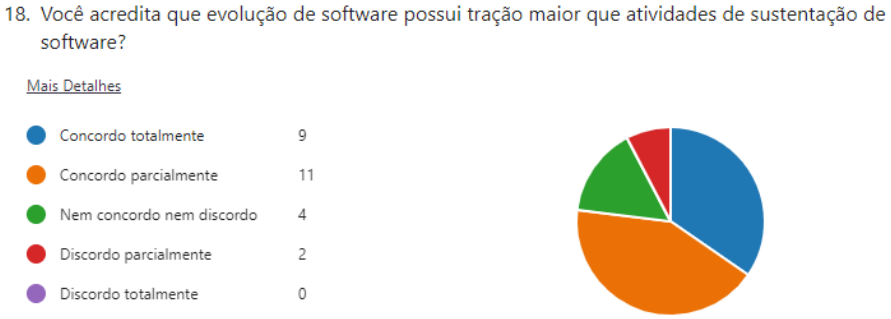
No quesito da comparação direta em relação a fluídez dos itens de evolução e manutenção foi buscado nas perguntas dezessete, dezoito e dezenove, as quais foram: "Você acredita que há igual tração para atividades de sustentação de software e evolução de software?", "Você acredita que evolução de software possui tração maior que atividades de sustentação de software?"e "Você acredita que atividades de sustentação de software possuem tração maior que atividades ligadas a evolução de software?"respectivamente, entender se há alguma diferença na tração desses nichos. Primeiro, foi explorado entender se na percepção dos respondentes há igualdade entre eles, posteriormente se evolução tem mais tração que manutenção e finalmente o cenário inverso. Os resultados, demonstrados, respectivamente nas Figuras 5.10, 5.11 e 5.12 demonstram que há desigualdade de tração nas atividades desses nichos, havendo tendência para maior fluidez nas atividades de evolução frente as de manutenção. Isso é perceptível através da análise das respostas da pergunta dezessete, o qual indica que há um grande número de respostas discordando que há equalidade de fluidez, sendo representados por 42% das respostas. Já para a dezoito o cenário é ainda mais incisivo, havendo 77% de respondentes que concordam que há fluidez maior por atividades de evolução em relação a manutenção. Finalmente, a pergunta dezenove contribui ainda mais para o cenário da pergunta anterior, visto que 42% das respostas apontaram para discordar que manutenção tem fluidez maior que evolução. Portanto, a partir dessas respostas, é possível inferir que há um maior interesse de atividades de evolução de *software* da IF em relação a manutenção, fazendo com que os profissionais tenham por consequência uma atenção maior a esses tipos de atividades.

Figura 5.10 – Survey - Questão 17



Fonte: Próprio Autor

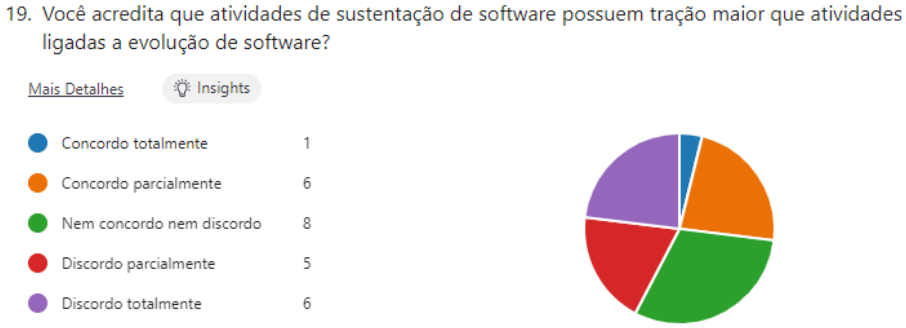
Figura 5.11 – Survey - Questão 18



Fonte: Próprio Autor

A justificativa possível para entendimento do porquê dessa diferença talvez esteja sendo demonstrado nas respostas das perguntas vinte e um e vinte e dois, as quais foram: "Itens entregues que possuem natureza de sustentação possuem maior visibilidade em relação a itens de evolução de software?"e "Itens entregues que possuem natureza de evolução de software possuem maior visibilidade em relação a natureza de sustentação de software?"respectivamente. Nessas duas perguntas foram buscados entender se haviam diferenças de visibilidade nas entregas de evolução em comparação com manutenção. Contribuindo com os cenários de fluidez maior para evolução, foi nítido que os profissionais acreditam que novas funcionalidades possuem visibilidade maior na cadeia de entregas em relação aos itens de sustentação. Novamente colaborando para que essa desigualdade entre os nichos seja justamente oriunda da IF dar maior atenção para a construção de algo novo em relação a manter e dar manutenção em algo já existente em seus sistemas produtos. Na pergunta vinte e dois 85% das respostas tiveram concordância positiva na afirmação que evolução de *software* possui maior visibilidade da entrega em relação a manutenção de *software*, conforme mostrado na Figura 5.13. Já o cenário que manutenção teria maior visibilidade quanto evolução, explorado pela pergunta vinte e um, demonstrou que 58% tiveram uma discordância negativa, conforma apresentado pela

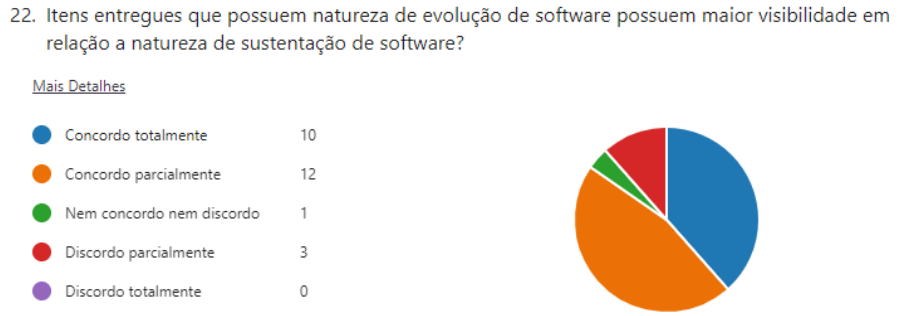
Figura 5.12 – Survey - Questão 19



Fonte: Próprio Autor

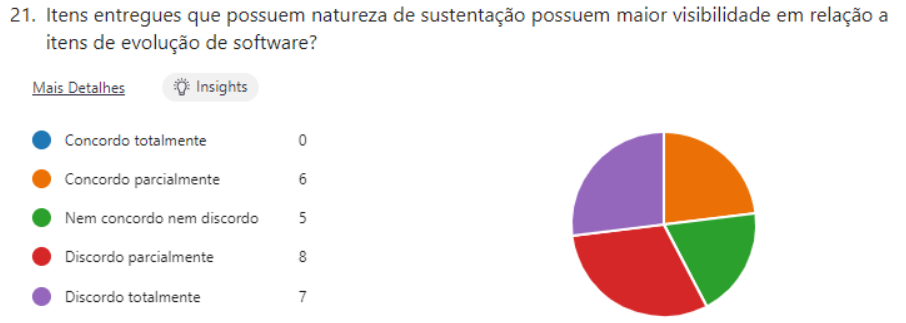
Figura 5.14, consolidando uma possível causa incisiva para que itens de novas funcionalidades tenham maior apreço e interesse da IF de fato.

Figura 5.13 – Survey - Questão 22



Fonte: Próprio Autor

Figura 5.14 – Survey - Questão 21



Fonte: Próprio Autor

As perguntas técnicas, presentes na segunda seção do questionário, buscam explorar o contexto e a percepção sobre a priorização, classificação e visibilidade do item apresentado. Para isso foram realizadas perguntas iguais para os distintos contextos das funcionalidades apresentadas. Logo abaixo são exemplificados os contextos para cada uma das 4 funcionalidades de forma respectiva.

A funcionalidade número 1, refere-se a um sistema de construção de propostas de crédito, a qual possui uma arquitetura de solução orientada a eventos, isto é, a medida que um evento (representado por uma criação ou atualização de uma proposta), é criado o mesmo vai para uma fila, de modo que periodicamente, é realizado o consumo desses eventos pelas filas consumidoras. Para evidenciar uma possível problemática para esse contexto, foi simulado uma situação a qual uma área interna da IF realizasse um pedido para revisão dessa lógica aplicada. O trecho de código representada pela Figura 5.15 nos mostra como foi realizado a implementação para representação desse cenário.

Figura 5.15 – Funcionalidade 1 - Trecho Código

```
3  treatEvent(event):
4      simulation = getSimulation()
5      switch event.eventType:
6          case CREATED:
7              createProposal(event, simulation)
8          case UPDATE:
9              if status == CANCELLED:
10                 cancelProposal(event)
11             else
12             ignoreEvent()
13         default:
14         ignoreEvent()
```

Fonte: Próprio Autor

Para a funcionalidade número 2, é apresentado um contexto para demonstração da implementação de métodos de liquidação da dívida de um produto específico de crédito em uma situação de inadimplência de longo período, chamado de "Prejuízo"pela instituição financeira. Nesse sentido, para elaborar um cenário de necessidade de alteração do estado original do sistema produto, foi realizado uma implementação com possibilidade de um erro específico, referente a não consolidar que o cliente, após o pagamento da dívida, estivesse com a contagem de número de dias em atraso não zerada, o qual pode provocar em insatisfação do cliente junto a IF. O trecho de código, representada pela Figura 5.16 mostra a implementação do método.

Figura 5.16 – Funcionalidade 2 - Trecho Código

```
2  entradaPrejuizo(conta):
3      if conta.diasAtraso >= 180 && conta.classificacaoRisco == 'H':
4      conta.emPrejuizo = true
5      salvarConta(conta)
6
7  saidaPrejuizo (conta, valorPagamento):
8      divida = getTotalDivida(conta)
9      saldo = getSaldo(conta)
10         if saldo >= valorPagamento:
11         conta.emPrejuizo = false
12         salvarConta(conta)
```

Fonte: Próprio Autor

A funcionalidade número 3, representa o contexto de haver duas funcionalidades realizando a mesma função, sendo portanto um método redundante na construção do sistema produto. A funcionalidade em si é relacionada ao contexto de cancelamento e renegociação de um produto de cheque-especial. Em ambos métodos é realizado a consolidação do total da dívida do cliente para que seja feito a validação se há saldo em conta corrente do mesmo cliente. Dessa forma é evidenciado a necessidade de solução de forma que não haja dois trechos de código para realizar uma mesma lógica. Essa implementação realizada está representada na Figura 5.17.

Figura 5.17 – Funcionalidade 3 - Trecho Código

```
cancelamentoProduto(conta, valorPagamento):
    dividaEncargos = getDividasEncargos(conta)
    dividaIOF = getDividasIOF(conta)
    saldo = getSaldo(conta)
    if saldo >= dividaEncargos + dividaIOF:
    conta.produtoCancelado = true
    salvarConta(conta)


renegociacaoProduto(conta)
    dividaTotal = getDividasEncargos(conta) + getDividasIOF(conta)
    valorParaCredito = dividaTotal
    criarPropostaCredito(conta, valorParaCredito)
```

Fonte: Próprio Autor

A funcionalidade número 4, demonstra uma problemática de funções com tipos de dados inadequados, sendo dado o contexto de uma busca de informações do produto baseado no número da conta corrente na IF. É destacado que conta corrente possui um tipo de dado inteiro, e na função foi implementada de forma proposital como *string*, havendo um tratamento com uma função nativa da linguagem utilizada para adequar a busca pelo produto através da lógica do método. A Figura demonstra a implementação realizada. Dessa forma é esperado que haja análise e adequação do tipo de dado a ser utilizado, sendo necessário mudanças no estado do *software*.

Figura 5.18 – Funcionalidade 4 - Trecho Código

```
consultaInformacoesConta(id, conta):
produtoRepositorio.buscarPorID(leftpad(conta, 6, "0"))
detalhesProduto(produtoRepositorio)
```

Fonte: Próprio Autor

Conforme mencionado no Capítulo 4 através da seção 4.2.2, as funcionalidades 2 e 4, representadas pelas perguntas 28, 29, 30, 31, 36, 37, 38 e 39, relacionam-se diretamente tipos de item de manutenção de software, já as funcionalidades 1 e 3, representadas pelas perguntas 24, 25, 26, 27, 32, 33, 34, 35 referem-se a itens do tipo de evolução de software.

No víes de priorização, foram realizadas perguntas com os enunciados: "esse item deveria ser priorizado na sprint atual?", explorados nas perguntas 24 para a funcionalidade 1, 28 para funcionalidade 2, 32 para a funcionalidade 3 e 36 para a funcionalidade 4.

As funcionalidades 1, 2 e 3 tiveram maior aderência de concordância dos respondentes em priorização, representando 69%, 81% e 62% respectivamente. Já a funcionalidade 4 teve divisões entre concordância e discordância, havendo uma divergência entre os profissionais, havendo 50% de concordância para priorização e pouco mais de 35% discordando com a priorização.

Para o quesito classificação do item perante o contexto apresentado das funcionalidades, foi explorado no questionário duas perguntas sendo elas: "O item seria classificado como manutenção de softwareo item seria classificado como manutenção de software", representados pelas perguntas 25 e 26 para a funcionalidade número 1, 29 e 30 para a funcionalidade número 2, 33 e 34 para a funcionalidade número 3, 37 e 38 para a funcionalidade número 4, as quais buscaram verificar qual seria o tipo de item pela percepção do respondente respectivamente.

A partir da funcionalidade 1, houve uma tendência maior para classificar o item como manutenção de software totalizando 65% de concordância positiva de respostas da pergunta 26. Contudo foi nítido a divergência entre os respondentes para classificação como evolução de software, representado na pergunta 25, já que houve um equilíbrio nas respostas analisadas, representados por 42% de concordância positiva e 46% de concordância negativa dos respondentes. Portanto é possível inferir que o público alvo apontou para que a funcionalidade 1 tivesse uma característica de evolução de software, contrariando o apontado anteriormente sobre o tipo correto do item atribuido a funcionalidade 1. Já para a funcionalidade 2, houve uma grande convergência para que o item fosse classificado como manutenção de software, sendo visível através da análise da resposta da pergunta 29, o qual totalizou 65% das respostas para que o item fosse apontado como manutenção de software. A pergunta 30 evidenciou que o item não fosse classificado como evolução, visto que houve um grande número de discordâncias para a pergunta, totalizando 58% de discordâncias. Na terceira funcionalidade, houve um equilíbrio maior nas respostas, apontando, de forma branda, uma tendência para que a funcionalidade fosse classificada como manutenção de software, visto que na pergunta 33 houve um total de 62% de respostas de concordância. Porém, na pergunta 34, houve uma divergência na opinião dos respondentes, já que 54% apontaram para uma concordância sobre evolução, enquanto que 42% apontaram para uma discordância sobre essa possível classificação.

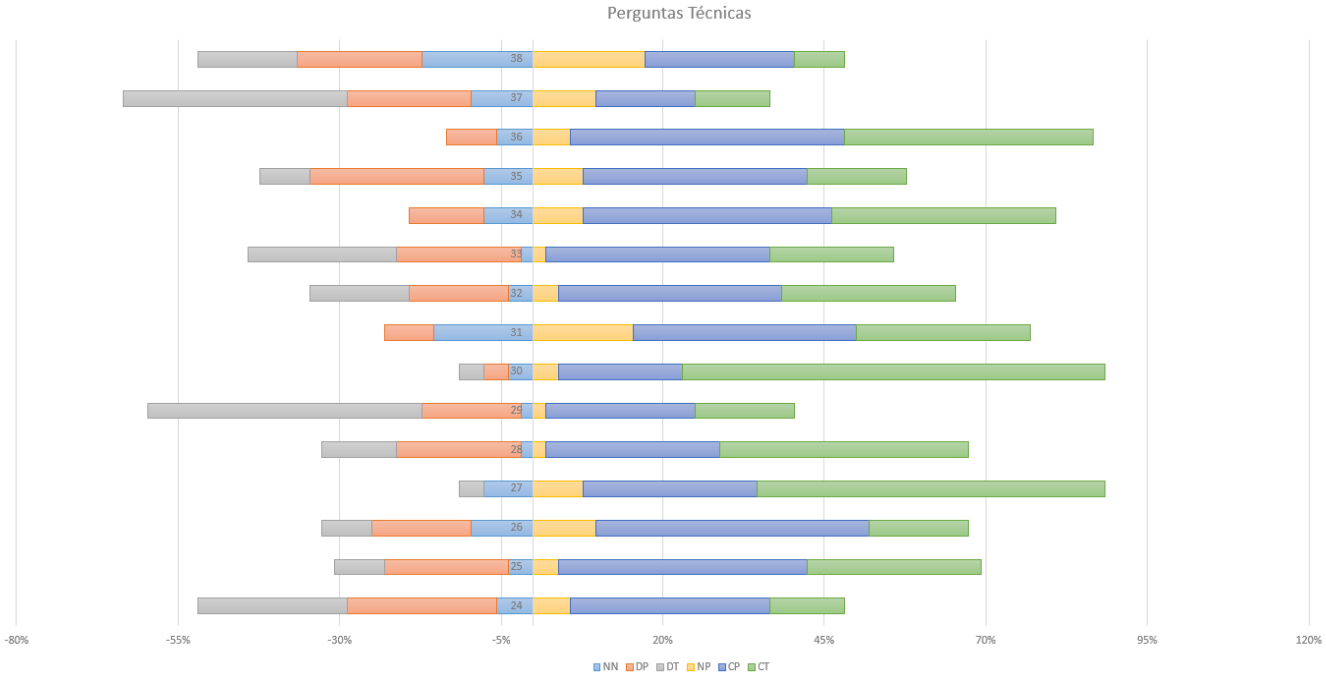
Na quarta funcionalidade, representadas pelas perguntas 37 e 38, houveram uma forte tendência para que o item fosse classificado como manutenção de software, totalizando 81% das respostas com concordância sobre manutenção de software, na pergunta 37, e 54% de discordância sobre evolução de software na pergunta 38.

Em relação a geração de valor, isto é, visibilidade perante as funcionalidades 1, 2, 3 e 4, representadas pelas perguntas 27, 31, 35 e 39 respectivamente, a pergunta realizada foi: "A resolução desse item traria valor para o cliente e/ou IF". Com esse questionamento, foi buscado a percepção dos profissionais perante a visibilidade da entregas sobre essas funcionalidades, de forma a buscar analisar se é possível haver um padrão de entendimento dos profissionais a respeito do assunto.

Houve grande tendência de concordância para as funcionalidades 1, 2 e 3, totalizando uma concordância de 58% na pergunta 27, 85% na pergunta 31, 73% na pergunta 35. Já para a funcionalidade 4, houve um equilíbrio maior das respostas, indicando de forma leve que a tendência seja de não gerar valor para o cliente final, não haver visibilidade da entrega para a empresa, já que houveram de forma concomitante 31% de concordância versus 35% de discordância para esse quesito.

A Figura 5.19 representa de forma consolidada e unificada a visão das perguntas técnicas com suas respostas.

Figura 5.19 – Survey - Consolidado Perguntas Técnicas



Fonte: Próprio Autor

5.3 Análise geral dos resultados

Com os resultados apresentados podemos inferir que, de forma incisiva, há uma tendência da IF estudada em ter uma tração maior nas atividades de evolução de software em relação a manutenção de software, conforme visto pelos resultados das perguntas exploradas no questionário. Isso se justifica pelo fato dos respondentes afirmarem que há maior geração de visibilidade e valor em entregas que possuem esse caráter de trazer novas funcionalidades. Atividades de sustentação, ou manutenção do sistema produto, demonstram não haver uma visibilidade estratégica da IF, de acordo com os respondentes.

Contudo, é preciso refletir que os sistemas de produto em ambientes produtivos estão sendo utilizados a todo momento, sendo natural a necessidade da sustentação dessa operacionalização, principalmente quando ocorrem incidentes ou erros de sistemas em função de anomalias de implementações realizadas no *software*. Não haver uma cadeia orgânica de manutenção e olhar mais crítico para temas com essa complexidade, implicam em um sistema menos confiável, e tratando-se serviços bancários e financeiros, a tolerância é praticamente nula na visão de cliente ou usuário.

Haver uma diretriz estratégica para implementar novas funcionalidades ao sistema produto é naturalmente compreensível, principalmente nos tempos atuais, onde há muitas *startups* de IF que estão concorrendo com muitas IF mais tradicionais, fazendo com que haja uma competição, e nessa cadeia de evolução, onde está o diferencial é que mora o sucesso, trazendo de forma orgânica mais clientes para si. Logo, a estratégia de uma IF também deve convergir para haver evoluções em seus sistemas produtos.

Apesar de ambas estrategias serem importantes, é recomendado que haja equílibrio entre ambos nichos, para que seja possível haver equalidade de tração nas atividades a nível operacional, isto é, no cotidiano dos times de produtos, os quais os profissionais respondentes do questionário estão presentes. O desiquilibrio pode causar o não interesse do profissional em atuar em uma atividade que não esteja no radar de visibilidade pela diretriz da IF, fazendo com que haja possível negligência de um nicho em relação ao outro.

5.4 Considerações Finais

Neste capítulo, foram apresentados todos os resultados da pesquisa empregada com o público alvo da IF estudada, contribuindo em si para a consolidação do cenário

objetivo desse trabalho. As perguntas exploraram os cenários do cotidiano vividos pelos profissionais de TI em ambientes que há de forma concomitante itens de evolução de *software* em comparação com manutenção de *software*. A partir da análise dos resultados foi possível buscar inferencias de possíveis preferências da IF perante a evolução de software, indicado pela visibilidade maior nas entregas desse quesito.

6 CONCLUSÃO

Esse trabalho apresentou uma comparação direta na atuação de desenvolvedores e analistas de qualidade para atuação em tarefas de evolução e manutenção de *software* de forma simultânea, utilizando-se de metodologias ágeis tradicionais como o *Scrum* e o *Kanban* em gerência de projetos de software com conceitos do tradicional PMBOK, os quais ainda são complementados com as regras de governança de TI, representadas pelo ITIL e COBIT.

O emprego da pesquisa fez com que fosse possível explorar de forma prática os conceitos abordados pela literatura perante os temas acima, na visão dos profissionais de TI da IF estudada, principalmente nos temas que comparam as atuações dos membros em atividades de evolução e manutenção dos sistemas, sendo possível verificar, após análise dos resultados, que a percepção desses profissionais indicam que a evolução de software, representados por novas funcionalidades ou *features* tem maior fluidez e aderência pela IF quando comparado com itens de sustentação, isto é, manutenção das funcionalidades já existentes, representados pela correção de incidentes, problemas ou quaisquer erros que impliquem em anomalias no sistema produto no contexto da IF estudada.

Essa divergência analisada foi justificada em função da visibilidade maior nas entregas de evolução em relação a manutenção, os quais podem indicar uma diretriz estratégica da instituição como um todo, visto evoluções normalmente são ligadas diretamente a inovação, e essa vem sendo uma carência a ser atacada de qualquer empresa no mercado como um todo.

Uma grande contribuição desse trabalho para os campos de engenharia de *software*, métodos ágeis e gestão de projetos foi trazer a tona o conflito e percepção dos profissionais a temas relacionados sustentar seus sistemas desenvolvidos com evolução e construção de novas features de forma simultânea, buscando mostrar o quão não saudável pode significar haver desiquilíbrio na estratégia das empresas que utilizam desses conceitos para manter seus sistemas produtos.

Com relação aos trabalhos futuros, o ideal seria expandir e complementar a pesquisa para envolver outras IF, para verificar se o cenário e conflito apresentado ocorre em outras realidades e ambientes distintos, buscando enriquecer a problemática e também evidenciar que o equilíbrio desses dois nichos seja a melhor solução. Outra possível sugestão é utilizar desses dados para contribuir na busca de perfis de profissionais que sejam mais aptos a tarefas relacionadas a ambos os nichos de forma mais versátil nos times ágeis

de produtos.

REFERÊNCIAS

ADAMS, S. **ITIL V3 foundation handbook**. [S.l.]: The Stationery Office, 2009.

Agile Manifesto. **Manifesto Ágil**. 2001. Disponível em: <http://agilemanifesto.org/iso/ptbr/principles.html>. Acesso em: 27 de Setembro de 2022.

AGUIAR, B.; CORREIA, W.; CAMPOS, F. Uso da escala likert na análise de jogos. **Salvador: SBC-Proceedings of SBGames Anais**, v. 7, n. 2, 2011.

ALBERTIN, A. L. Modelo de comércio eletrônico e um estudo no setor bancário. **Revista de Administração de Empresas**, SciELO Brasil, v. 39, p. 64–76, 1999.

ANDERSON, D. J. **Kanban: successful evolutionary change for your technology business**. [S.l.]: Blue Hole Press, 2010.

ANDRADE, S. C. de; TAIT, T. F. C. Uma aplicação do guia pmbok na gestão de projetos de software. **Revista Brasileira de Computação Aplicada**, v. 4, n. 1, p. 2–11, 2012.

BARBOSA, A. M.; LIMA, V. Governança em ti: Cobit; itil. **Revista científica Eletrônica de Administração**, 2011.

BERMUDES, W. L. et al. Tipos de escalas utilizadas em pesquisas e suas aplicações. **Revista Vértices**, v. 18, n. 2, p. 7–20, 2016.

CHAPIN, N. et al. Types of software evolution and software maintenance. **Journal of software maintenance and evolution: Research and Practice**, Wiley Online Library, v. 13, n. 1, p. 3–30, 2001.

CRUZ, F. **Scrum e PMBOK unidos no Gerenciamento de Projetos**. [S.l.]: Brasport, 2013.

Federação Brasileira de Bancos, h. A. e. . d. S. d. . **"Com pandemia, transações bancárias por celular ultrapassam 50% de operações feitas pelos brasileiros"**. 2022.

FONSECA, J. J. S. da. **Apostila de metodologia da pesquisa científica**. [S.l.]: João José Saraiva da Fonseca, 2002.

GERHARDT, T. E.; SILVEIRA, D. T. **Métodos de pesquisa**. [S.l.]: Plageder, 2009.

ISO, S. 14764 on software engineering-software maintenance. **ISO/IEC**, 1999.

KNIBERG, H. **Kanban vs. Scrum: Making the most of both, 2009**. 2016.

KOSCIANSKI, A.; SOARES, M. dos S. **Qualidade de Software-2ª Edição: Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software**. [S.l.]: Novatec Editora, 2007.

LEHMAN, M.; RAMIL, J. F. Software evolution in the age of component-based software engineering. **IEE Proceedings-Software**, IET, v. 147, n. 6, p. 249–255, 2000.

LEHMAN, M. M.; BELADY, L. A. **Program evolution: processes of software change**. [S.l.]: Academic Press Professional, Inc., 1985.

MANGALARAJ, G.; SINGH, A.; TANEJA, A. It governance frameworks and cobit-a literature review. 2014.

MORAES, E. A. P. Guia pmbok para gerenciamento de projetos. In: **Anais do Congresso Nacional de Excelência em Gestão, Rio de Janeiro, RJ, Brasil**. [S.l.: s.n.], 2012. v. 8.

NIESSINK, F.; VLIET, H. V. Software maintenance from a service perspective. **Journal of Software Maintenance: Research and Practice**, Wiley Online Library, v. 12, n. 2, p. 103–120, 2000.

OGHENEOVO, E. E. Software maintenance and evolution: The implication for software development. **West African Journal of Industrial and Academic Research**, v. 7, n. 1, p. 81–92, 2013.

OLIVEIRA, M. P. de; MALAGOLLI, G. A. O impacto da tecnologia da informação na evolução dos serviços bancários. **Revista Interface Tecnológica**, v. 13, n. 1, p. 39–52, 2016.

PFLEEGER, S. L.; ATLEE, J. M. **Software engineering: theory and practice**. [S.l.]: Pearson Education India, 2010.

PMBOK, G. d. P. Artigo engenharia de software 23-gestão de projetos segundo o pmbok. 2012.

POPPENDIECK, M.; POPPENDIECK, T. **Lean software development: an agile toolkit**. [S.l.]: Addison-Wesley, 2003.

SANTOS, A. R. d. Metodologia científica: a construção do conhecimento. In: **Metodologia científica: a construção do conhecimento**. [S.l.: s.n.], 2011. p. 139–139.

SCHWABER, K.; SUTHERLAND, J. Scrum. **Siehe: http://www. scrum. org/Resources/What-is-Scrum**, 2010.

SHINGO, S. **O sistema Toyota de produção**. [S.l.]: Bookman Editora, 1996.

SILVA, D.; SANTOS, F.; NETO, P. S. Os benefícios do uso de kanban na gerência de projetos de manutenção de software. In: **Anais do VIII Simpósio Brasileiro de Sistemas de Informação**. Porto Alegre, RS, Brasil: SBC, 2012. p. 715–725. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/sbsi/article/view/14454>.

STAMMEL, J. et al. **Software evolution for industrial automation systems: literature overview**. [S.l.]: KIT, Fakultät für Informatik, 2011.

TRIPP, L. L. et al. **IEEE Std 1219-1992, IEEE Standard for Software Maintenance.). Institute of Electrical and Electronics Engineers**. [S.l.]: Inc, 1998.

APÊNDICE A — SURVEY

Questionário TG - Leonardo Mendes

Meu nome é Leonardo Felipe Mendes, sou estudante de Engenharia de Computação pela Universidade Federal do Rio Grande do Sul.

O intuito desse questionário é fazer com o que seja possível analisar de forma exploratória as percepções sobre a atuação de profissionais de tecnologia da informação envolvidos em atividades de desenvolvimento de software (evolução) versus manutenção de software (sustentação).

Seus dados utilizados serão sigilosos e servirão para análise perante ao tema de modo a contribuir com o trabalho de conclusão do curso.

* Obrigatória

Bloco de perguntas teóricas.

Essa seção do questionário engloba perguntas teóricas a respeito dos conceitos em comum ao dia a dia dos profissionais de TI da IF.

1

Os métodos ágeis são compreendidos pelos membros de seu time ágil de produto *

- Concordo totalmente
- Concordo parcialmente
- Nem concordo nem discordo
- Discordo parcialmente
- Discordo totalmente

2

O conceito de sustentação de software ou suporte sobre as aplicações são compreendidos pelos membros do time ágil do seu produto. *

- Concordo totalmente
- Concordo parcialmente
- Nem concordo nem discordo
- Discordo parcialmente
- Discordo totalmente

3

Em reuniões de planejamento, dentro das cerimonias conhecidas do scrum/kanban, os itens de *backlog* do produto priorizados são oriundos de evolução de software, isto é, em relação a novas funcionalidades. *

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

4

Em reuniões de planejamento, dentro das cerimonias conhecidas do scrum/kanban, os itens de *backlog* do produto priorizados são oriundos de manutenção de software, isto é corrigir funcionalidades existentes. *

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

5

Assuntos extraordinários que surgirem fora da reunião de planejamento da cerimonia de scrum/kanban normalmente são oriundos de novas funcionalidades a serem desenvolvidas pelo time de produto. *

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

6

Assuntos extraordinários que surgirem fora da reunião de planejamento da cerimonia de scrum/kanban normalmente são oriundos de manutenção de software a serem analisadas pelo time de produto. *

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

7

Incidentes ou problemas são apresentados a equipe em reuniões de planejamento dos itens de *backlog* a serem absorvidos em uma nova *sprint*. *

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

8

Histórias de usuários e épicos são apresentados a equipe em reuniões de planejamento dos itens de *backlog* a serem absorvidos em uma nova *sprint*. *

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

9

Incidentes ou problemas são apresentados a equipe em momentos diferentes de planejamento de *sprint*. *

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

10

Histórias de usuário ou épicos são apresentados a equipe em momentos diferentes de planejamento de *sprint*. *

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

11

Incidentes ou problemas absorvidos pelos times ágeis de produto possuem uma criticidade maior em relação a itens de evolução de software. *

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

12

Um incidente deve ser analisado e resolvido o quanto antes. *

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

13

Um problema deve ser analisado e resolvido o quanto antes. *

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

14

Novas funcionalidades devem ser analisadas e implementadas o quanto antes. *

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

15

A resolução de um incidente traz valor para o cliente final. *

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

16

A entrega de uma nova funcionalidade em geral traz valor para o cliente final. *

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

17

Você acredita que há igual tração para atividades de sustentação de software e evolução de software? *

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

18

Você acredita que evolução de software possui tração maior que atividades de sustentação de software? *

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

19

Você acredita que atividades de sustentação de software possuem tração maior que atividades ligadas a evolução de software? *

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

20

Dentro de sua equipe todos atuam em atividades de sustentação de software? *

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

21

Itens entregues que possuem natureza de sustentação possuem maior visibilidade em relação a itens de evolução de software? *

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

22

Itens entregues que possuem natureza de evolução de software possuem maior visibilidade em relação a natureza de sustentação de software? *

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

Bloco de perguntas contexto técnico

Nessa seção de perguntas será apresentado o contexto por trás de cada cenário e um

23

Qual seu papel na sua equipe atualmente? *

- ◯ Desenvolvedor de Software
- ◯ Analista de Qualidade

24

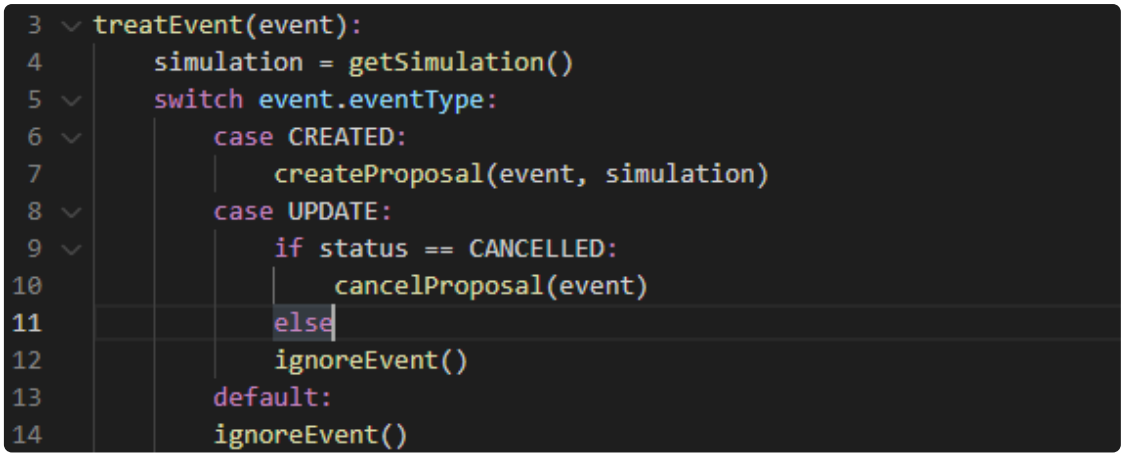
As perguntas número 24, 25, 26 e 27 referem-se a esse contexto abaixo:

Atualmente o sistema de construção de propostas de crédito possui uma arquitetura de solução orientada a eventos, isto é, a medida que um evento é criado o mesmo vai para uma fila, de modo que de tempos em tempos é realizado o consumo desses eventos pelos consumidores.

Além da construção de propostas, também há a necessidade de efetuar mudanças nas propostas para fins de atualização. Dessa forma o trecho abaixo de código expõe a lógica empregada:

Diante do cenário, foi realizado o pedido da infraestrutura para realização de análise, já que estão sendo gerados eventos desnecessários em algumas situações.

A partir dessa afirmação, esse item deveria ser priorizado em sua sprint. *



```
treatEvent(event):
    simulation = getSimulation()
    switch event.eventType:
        case CREATED:
            createProposal(event, simulation)
        case UPDATE:
            if status == CANCELLED:
                cancelProposal(event)
            else
            ignoreEvent()
        default:
        ignoreEvent()
```

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

25

O item é classificado como uma evolução de software. *

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

26

O item é classificado como manutenção de software. *

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

27

A resolução desse item traria valor para o cliente e ou IF. *

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

28

As perguntas número 28, 29, 30 e 31 referem-se a esse contexto abaixo:

No contexto de sistema de crédito, após um período de inadimplência, é necessário a atualização para o status de "Prejuízo", visto que a IF não terá mais receitas e nem despesas com essa operação de crédito concedida. Porém, para o cliente, é possível a quitação dessa dívida de modo que o mesmo possa ficar sem débitos ativos pendentes com a IF.

Diante desse cenário, foi reportado a equipe de desenvolvimento a necessidade de verificar o processo de saída de prejuízo, já que mesmo com a quitação total da dívida, ainda está sendo contabilizado o número de dias em atraso.

O trecho de código abaixo demonstra as condições para entrada e saída de prejuízo.

A partir desse contexto, o item deveria ser priorizado na sprint. *

```
2  entradaPrejuizo(conta):
3      if conta.diasAtraso >= 180 && conta.classificacaoRisco == 'H':
4      conta.emPrejuizo = true
5      salvarConta(conta)
6
7  saidaPrejuizo (conta, valorPagamento):
8      divida = getTotalDivida(conta)
9      saldo = getSaldo(conta)
10         if saldo >= valorPagamento:
11         conta.emPrejuizo = false
12         salvarConta(conta)
```

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

29

O item seria classificado como manutenção de software. *

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

30

O item seria classificado como evolução de software. *

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

31

A resolução desse item traria valor para o cliente e/ou IF. *

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

32

As perguntas número 32, 33, 34 e 35 referem-se a esse contexto abaixo:

Em um cenário de necessidade de cancelamento de um produto de crédito, por exemplo cartão de crédito ou cheque-especial, é possível que seja realizado o cancelamento a partir da solicitação do cliente ou então a partir de uma renegociação do produto, sendo esse exclusivo para clientes em situação de inadimplência.

Para o cancelamento do produto de forma direta, basta que o cliente tenha saldo em sua conta corrente de forma que seja ao menos equivalente ao valor total da dívida.

Para renegociação do produto, além da condição de inadimplência do cliente, o valor total da dívida é dado como um valor de crédito a ser realizado em sua conta, sendo um novo produto contratado que poderá ser pago em parcelas mensais.

O trecho de código abaixo demonstra o funcionamento dessas funcionalidades.

Foi verificada a necessidade de uma revisão do fluxo dessas duas funcionalidades.

Essa necessidade de revisão deveria ser priorizada na sprint atual. *

```
cancelamentoProduto(conta, valorPagamento):
    dividaEncargos = getDividasEncargos(conta)
    dividaIOF = getDividasIOF(conta)
    saldo = getSaldo(conta)
    if saldo >= dividaEncargos + dividaIOF:
    conta.produtoCancelado = true
    salvarConta(conta)


renegociacaoProduto(conta)
    dividaTotal = getDividasEncargos(conta) + getDividasIOF(conta)
    valorParaCredito = dividaTotal
    criarPropostaCredito(conta, valorParaCredito)
```

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

33

O item seria classificado como manutenção de software. *

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

34

O item seria classificado como evolução de software. *

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

35

A resolução desse item traria valor para o cliente e ou IF. *

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

36

As perguntas número 36, 37, 38 e 39 referem-se a esse contexto abaixo:

Em um sistema de consultas de informações de sua conta, é comum que sejam necessários buscar os produtos contratados para visualização ao cliente.

Nesse sentido as contas correntes possuem um número de máximo de algarismos, sendo o mesmo 6 algarismos.

A função abaixo realiza a busca das informações dos produtos a partir da conta, contudo nota-se que há um tratamento através da função nativa para modelagem do dado a ser buscado.

Foi constatado pela equipe a necessidade efetuar a mudança dessa lógica de modo a igualar os tipos de dados utilizados com o banco de dados.

Esse item deveria ser priorizado pela sua equipe na sprint. *

```
consultaInformacoesConta(id, conta):
produtoRepositorio.buscarPorID(leftpad(conta, 6, "0"))
detalhesProduto(produtoRepositorio)
```

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

37

O item seria classificado como manutenção de software. *

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

38

O item seria classificado como evolução de software. *

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente

39

A resolução desse item traria valor para o cliente e ou IF. *

- ◯ Concordo totalmente
- ◯ Concordo parcialmente
- ◯ Nem concordo nem discordo
- ◯ Discordo parcialmente
- ◯ Discordo totalmente