MÉTODOS DE TESTAGEM DE *SOFTWARE* UTILIZADOS POR DESENVOLVEDORES DE *SOFTWARE* EM URUAÇU-GO E REGIÃO

SOFTWARE TESTING METHODS USED BY *SOFTWARE* DEVELOPERS IN URUAÇU-GO AND NEIGHBORHOOD

Alexandre Martins Ferreira Bueno[1]
alexandre.bueno@ifg.edu.br

Katiane De Souza Santos[2]
katianesantos4567@gmail.com

Resumo

Este projeto de pesquisa propôs investigar quais eram as principais técnicas de testagem de *software* utilizadas por profissionais desenvolvedores de *software* que atuam a partir de Uruaçu-GO e região. Ele também se propôs a entender como funcionavam os processos de desenvolvimento de *software* nos quais esses profissionais estavam envolvidos, além de aspectos envolvendo gestão/qualidade de *software*. Para isso, foram realizadas entrevistas com seis profissionais. Cada entrevista seguiu um roteiro definido, representado por um formulário. A partir da coleta, tabulação e análise dos dados, observou-se que cada empresa possuía um processo de desenvolvimento único, organizado em equipes pequenas e que faziam uso de métodos ágeis para o desenvolvimento de *software* em incrementos. Observou-se que as equipes possuíam profissionais dedicados à gestão de projetos e de garantia da qualidade, e que vários tipos de teste eram empregados pelos desenvolvedores. Também foram identificados pontos de melhoria nos processos que merecem atenção, como a adoção de forma ampla de testes unitários automatizados e maior participação dos usuários durante as etapas de desenvolvimento dos *softwares*. A partir dessas análises, entendeu-se que os principais objetivos desejados com esse trabalho foram alcançados.

Palavras-chaves: Teste de *software*; Processo de desenvolvimento de *software*; Qualidade de *software*; Gestão de *software*.

Abstract

This research project proposed to investigate the main *software* testing techniques used by professional *software* developers working in Uruaçu-GO and neighborhood. It also aimed to understand how worked the *software* development processes in which these professionals were involved, in addition to aspects involving *software* management/quality. For this, interviews were carried out with six professionals. Each interview followed a defined script, represented by a form. From data collection, tabulation and analysis, it was observed that each company had a unique development process, organized in small teams and that used agile methods to develop *software* in increments. It was observed that the teams had professionals dedicated to project management and quality assurance, and that various types of testing were used by the developers. Points for improvement in the processes analyzed that deserve attention were also identified, such as the larger adoption of automated unit tests and greater user participation during the *software* development stages. From these analyses, it was understood that the main objectives desired with this work were achieved.

Keywords: *Software* testing; *Software* quality; *Software* development process; *Software* management.

Introdução

A demanda por *softwares* e sistemas de melhor qualidade é uma realidade a nível mundial. Usuários e empresas cobram cada vez mais *softwares* que sejam úteis, seguros e que agreguem

[1] Bacharel e mestre em Engenharia de Computação e professor de informática do Instituto Federal de Goiás – Câmpus Uruaçu.

[2] Egressa do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Goiás – Câmpus Uruaçu.

valor. Existe uma cobrança por *softwares* com cada vez mais funcionalidades e que sejam entregues dentro de cronogramas "apertados" (SOMMERVILLE, 2018).

Sabe-se que dentro do processo de construção de um *software*, a testagem dos seus artefatos desempenha um papel essencial, colaborando para que possíveis erros/inconsistências sejam encontrados antes da entrega do *software* aos seus clientes. Isto permite com que estes defeitos sejam "consertados" e que o *software* entregue aos clientes possua maior garantia quanto à sua qualidade.

A preocupação com a qualidade de sistemas de *software* cresceu à medida que o *software* passou a se tornar cada vez mais integrado em cada aspecto da vida cotidiana (PRESMANN, 2021). Testar um *software* desde o início do seu processo de construção/desenvolvimento é uma etapa essencial para garantir a sua qualidade. Diante desta importância, empresas desenvolvedoras e profissionais fazem uso dos testes nos seus processos de desenvolvimento.

Sabe-se também que um *software* pode ser testado nas diversas fases do seu desenvolvimento e que muitas são as técnicas de desenvolvimento que podem ser utilizadas em cada uma destas fases. Em geral, as empresas e desenvolvedores customizam as práticas de teste de *software* de acordo com a sua necessidade/realidade.

Em uma tentativa por parte dos autores deste trabalho em investigar quais eram os principais métodos de testagem de *software* empregados por profissionais na cidade de Uruaçu-GO e região, não foi encontrado material publicado de forma estruturada. Não foi possível identificar uma "visão geral" sobre como os processos de testagem estão sendo empregados por estes profissionais.

Diante desta lacuna, este projeto de pesquisa propôs investigar quais são os principais métodos de testagem de *software* utilizados por profissionais desenvolvedores de *software* em Uruaçu e região. A partir de uma coleta de dados com estes profissionais e da análise dessas informações, pretendeu-se também: identificar outros aspectos utilizados relativos à gestão/qualidade de *software*; evidenciar os principais aspectos encontrados; e identificar possíveis lacunas e propor melhorias nos processos/métodos encontrados.

Fundamentação teórica

Qualidade de software

Segundo Presmann (2021), a qualidade de *software* é o resultado de um bom gerenciamento de projeto e uma prática consistente de engenharia de *software*. O gerenciamento e a prática são aplicados no contexto de quatro atividades amplas que ajudam uma equipe de *software* a atingir alto padrão de qualidade de *software*: métodos e processos de engenharia de *software*, técnicas de gerenciamento de projetos, ações de controle de qualidade e garantia da qualidade de *software*.

O controle de qualidade de *software* engloba um conjunto de ações de engenharia de *software* que ajudam a garantir que cada produto resultante atinja suas metas de qualidade. Os modelos são revistos de modo a garantir que sejam completos e consistentes. O código poderia ser inspecionado de modo a revelar e corrigir erros antes de os testes começarem. Aplica-se uma série

de etapas de teste para descobrir erros na lógica de processamento, na manipulação de dados e na comunicação da interface. Uma combinação de medições e realimentação (feedback) permite a uma equipe de *software* ajustar o processo quando qualquer um desses produtos resultantes deixe de atender às metas estabelecidas para a qualidade (PRESMANN, 2021).

A garantia da qualidade de *software* (SQA, do inglês *Software* Quality Assurance) estabelece a infraestrutura que suporta métodos sólidos de engenharia de *software*, gerenciamento racional de projeto e ações de controle de qualidade. Além disso, ela também consiste em um conjunto de funções de auditoria e de relatórios que possibilita uma avaliação da efetividade e da completude das ações de controle de qualidade. Se os dados fornecidos pela garantia da qualidade identificarem problemas, é responsabilidade do gerenciamento tratar desses problemas e aplicar os recursos necessários para resolver os problemas de qualidade (PRESMANN, 2021).

Um pressuposto do gerenciamento de qualidade de *software* é que a qualidade do *software* é diretamente relacionada à qualidade do processo de desenvolvimento de *software*. Não há dúvida de que o processo de desenvolvimento usado tenha uma influência significativa sobre a qualidade de *software* e que bons processos são mais suscetíveis de conduzirem a um *software* de boa qualidade. O gerenciamento e a melhoria de qualidade de processo podem gerar *softwares* com menos defeitos (SOMMERVILLE, 2018).

Uma importante etapa em um processo de desenvolvimento de *software* são os mecanismos de testes que esse processo utiliza. As próximas seções detalham os principais tipos de teste de *software* existentes e que são utilizados nos processos de desenvolvimento.

Testes de software

Esta seção e suas subseções foram escritas tendo como referência Sommerville (2018). O teste de *software* é destinado a mostrar que um programa faz o que é proposto a fazer e para descobrir os defeitos do programa antes do uso. Quando se testa o *software*, o programa é executado usando dados fictícios. Os resultados do teste são verificados à procura de erros, anomalias ou informações sobre os atributos não funcionais do programa.

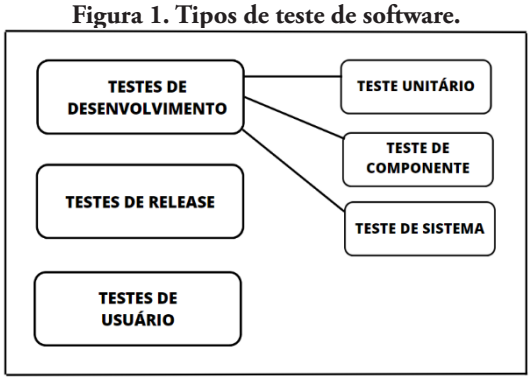
Segundo Presmann (2021), o objetivo do teste de *software* é descobrir erros. O teste de *software* absorve a maior porcentagem do esforço técnico em um processo de *software*. Independentemente do tipo de *software* criado, uma estratégia para planejamento sistemático de teste, execução e controle começa considerando pequenos elementos do *software* e se encaminha para fora no sentido de abranger o programa como um todo.

O teste de *software* é parte de um amplo processo denominado Verificação e Validação. Este processo também inclui inspeções e revisões. Estas técnicas analisam e verificam os requisitos de sistema, modelos de projeto, o código-fonte de programa e até mesmo os testes de sistema propostos. Essas são chamadas técnicas "estáticas" de verificação e validação, em que você não precisa executar o *software* para verificá-lo (ao contrário dos testes de *software*, que ocorrem a partir da execução do *software* ou de partes dele) (SOMMERVILLE, 2018; SCHACH, 2009). O processo de teste de *software* tem dois objetivos distintos: demonstrar ao desenvolvedor e ao

cliente que o *software* atende a seus requisitos (testes de validação) e descobrir situações em que o *software* se comporta de maneira incorreta, indesejável ou de forma diferente das especificações (testes de defeitos).

Em geral, um *software* passa por três estágios de teste (ilustrados na Figura 1): testes em desenvolvimento, em que o sistema é testado durante o desenvolvimento para descobrir bugs e defeitos; testes de release, em que uma equipe de teste independente testa uma versão completa do sistema antes que ele seja liberado para os usuários; e testes de usuário, em que os usuários ou potenciais usuários de um sistema testam o sistema (SOMMERVILLE, 2018).

Figura 1. Tipos de teste de software.



Fonte: Próprio autor (2024).

Testes de desenvolvimento

Os testes de desenvolvimento incluem todas as atividades de testes que são realizadas pela equipe de desenvolvimento do sistema. O testador do *software* geralmente é o programador que o desenvolveu. Este tipo de teste é essencialmente um processo de teste de defeitos, em que o objetivo do teste é descobrir bugs no *software* (SOMMERVILLE, 2018).

Durante o desenvolvimento de um *software*, o teste pode ocorrer em três níveis: teste unitário, teste de componentes e teste de sistema. Eles são descritos nas seções a seguir.

Teste unitário

O teste unitário ou de unidade é o processo de testar os pequenos componentes de um programa, como métodos ou classes de objetos. As funções individuais ou métodos são o tipo mais simples de componente de um *software*. Testes unitários são programados chamando esses métodos com parâmetros diferentes de entrada e observando as saídas apresentadas (SOMMERVILLE, 2018).

Além de dar enfoque à lógica interna da unidade, esse tipo de teste é feito para garantir que a informação (ou seja, os dados) flui de outras unidades para a unidade sob teste e da unidade testada para outras, garantindo que aquela pequena parte do sistema cumpra seu papel sem defeitos (MARTINS, 2016).

Sempre que possível, deve-se automatizar os testes unitários. Em testes unitários automatizados, pode-se usar um framework de automação de teste (como JUnit) para escrever

e executar testes de seu programa. Frameworks de testes unitários fornecem classes de teste genéricas que você pode estender para criar casos de teste específicos. Eles podem, então, executar todos os testes implementados e informar sobre o sucesso ou o fracasso deles.

Testes são custosos e demorados, por isso é importante a escolha casos efetivos de teste unitário. A depender da complexidade do *software*, ele pode ter centenas de testes unitários implementados. Um caso de teste é uma sequência de passos que devem ser executados no sistema, sendo que os dados de entrada e saída esperada para cada passo são especificados (MARTINS, 2016).

Diretrizes mais gerais que podem ser utilizadas para a criação de casos de testes são: escolha entradas que forcem o sistema a gerar todas as mensagens de erro; projete entradas que causem overflow de buffers de entrada; repita a mesma entrada ou uma série de entradas inúmeras vezes; obrigue a geração de saídas inválidas; e obrigue os resultados de cálculos a serem muito grandes ou muito pequenos.

Teste de componente

Os componentes do *software* são compostos de diversos objetos/funções que interagem. Testes de componentes centram-se em mostrar que a interface de um componente se comporta de acordo com sua especificação (assume-se que os testes unitários sobre os objetos individuais dentro do componente já foram concluídos) (SOMMERVILLE, 2018).

Não se pode garantir que mesmo com objetos/funções funcionando corretamente, que quando juntados/integrados, eles também funcionarão corretamente. Daí a necessidade de se testar os componentes. O teste de componente é uma maneira sistemática de identificar defeitos associados às interfaces (interface, aqui, se refere a forma como as unidades se comunicam, e não a interface gráfica do sistema) (MARTINS, 2016).

Como a maioria dos componentes não é independente, é importante se assegurar que, quando da integração de um componente ao programa que está em evolução, ele não leve o programa a falhas e defeitos (PRESMANN, 2021).

Os casos de teste nestes casos não são aplicados aos componentes individuais/objetos, mas sim à interface de componente criada pela combinação desses componentes. Erros de interface no componente podem não ser detectáveis por meio de testes em objetos individuais, pois esses erros resultam de interações entre os objetos do componente.

Geralmente, erros de interface são classificados em três classes:

- Mau uso de interface – Um componente chamador chama outro componente e comete um erro no uso de sua interface (por exemplo, os parâmetros passados podem ser de tipo errado ou ser passados na ordem errada);
- Mau-entendimento de interface – Um componente chamador desconhece a especificação da interface do componente chamado e faz suposições sobre seu comportamento (fazendo com que ele não se comporte conforme o esperado);
- Erros de *timing* – Eles ocorrem em sistemas que usam uma memória compartilhada

ou passagem de mensagens. O produtor e o consumidor de dados podem operar em velocidades diferentes, produzindo inconsistências.

Teste de sistema

O teste de sistema, durante o desenvolvimento, envolve a integração de componentes para criação de uma versão do sistema e, em seguida, o teste do sistema integrado. O teste de sistema verifica se os componentes são compatíveis, se interagem corretamente e transferem os dados certos no momento certo, por suas interfaces (SOMMERVILLE, 2018).

Nesse estágio, componentes desenvolvidos por diferentes membros da equipe ou grupos podem ser integrados. O teste de sistema é um processo coletivo, não individual.

Quando se integra componentes para criar um sistema, obtém-se um comportamento emergente. Isso significa que alguns elementos da funcionalidade do sistema só se tornam evidentes quando colocados juntos. Esse comportamento emergente precisa ser testado. É preciso desenvolver testes que verifiquem se o sistema está fazendo apenas o que ele supostamente deve fazer.

Para a maioria dos sistemas, é difícil saber o quanto o teste de sistema é essencial e quando se deve parar de testar. É impossível fazer testes exaustivos, em que cada sequência possível de execução do programa seja testada. Como alternativa, os testes podem basear-se na experiência de uso do sistema e centrar-se em testes de características como, por exemplo: todas as funções do sistema acessadas por meio de menus devem ser testadas; combinações de funções acessadas por meio de menu devem ser testadas; e, nos casos em que a entrada do usuário é fornecida, todas as funções devem ser testadas com entradas corretas e incorretas (SOMMERVILLE, 2018).

Testes de release

O teste de release ou de lançamento é o processo de testar uma versão particular de um sistema. Existem duas diferenças importantes entre o teste de release e o teste de sistema (durante o processo de desenvolvimento): uma equipe separada, que não esteve envolvida no desenvolvimento do sistema, deve ser responsável pelo teste de release; e testes de sistema feitos pela equipe de desenvolvimento devem centrar-se na descoberta de bugs no sistema (teste de defeitos), já o objetivo do teste de release é verificar se o sistema atende a seus requisitos e é bom o suficiente para uso externo (teste de validação) e, se assim for, o sistema poderá ser lançado como um produto ou entregue aos clientes (SOMMERVILLE, 2018).

Portanto, o teste de release precisa mostrar que o sistema oferece a funcionalidade, o desempenho e a confiança especificados e que não falhará durante o uso normal. Ele costuma ser um processo de teste de caixa-preta, no qual os testes são derivados da especificação de sistema. O sistema é tratado como uma caixa-preta, cujo comportamento só pode ser determinado por meio do estudo das entradas e saídas relacionadas (não havendo preocupação sobre como o sistema funciona internamente). Os testes do tipo caixa-preta também são chamados de testes comportamentais ou testes funcionais, pois focam em testar os requisitos funcionais do *software* (PRESMANN, 2021).

Os testes de release geralmente são realizados por membros do grupo de SQA ou, em caso da ausência desse grupo, por profissionais diretamente associados a garantir a qualidade da versão do *software* que está sendo testado (SCHACH, 2009).

Testes de usuário

Teste de usuário ou de cliente é um estágio no processo de teste em que os usuários ou clientes fornecem entradas e conselhos sobre o teste que fizeram com o sistema. Isso pode envolver o teste formal de um sistema que foi testado por um fornecedor externo ou processo informal em que os usuários experimentam um produto de *software* novo para ver se gostam e verificar se faz o que eles precisam. O teste de usuário é essencial, mesmo em sistemas abrangentes ou quando testes de *release* tenham sido realizados. A razão para isso é que as influências do ambiente de trabalho do usuário têm um efeito importante sobre a confiabilidade, o desempenho, a usabilidade e a robustez de um sistema (SOMMERVILLE, 2018).

Usualmente, existem três tipos de testes de usuário:

• Teste *alfa* – em que os usuários do *software* trabalham com a equipe de desenvolvimento para testar o *software* no local do desenvolvedor. Dessa forma, o *software* é usado com o desenvolvedor "espiando por cima dos ombros" dos usuários, registrando os erros e os problemas de uso (PRESMANN, 2021). Este tipo de teste geralmente é feito em *softwares* do tipo de prateleira (que são comprados prontos como, por exemplo, pacotes *office* e antivírus);

• Teste *beta* – em que um release do *software* é disponibilizado aos usuários para que possam experimentá-lo. Este teste é realizado no ambiente do usuário, com o desenvolvedor coletando o *feedback*. O teste beta é uma aplicação de uso do *software* em um ambiente que não pode ser controlado pelo desenvolvedor (PRESMANN, 2021). Geralmente, este tipo de teste é feito em *softwares* do tipo de prateleira e realizado após a conclusão dos testes *alfa*;

• Teste de aceitação – em que os clientes testam um sistema para decidir se ele está ou não pronto para ser aceito e implantado no ambiente do cliente. Ele pode ser usado no processo de desenvolvimento de sistemas customizados (ao contrário dos *softwares* de prateleira, onde normalmente este tipo de teste não é feito), ocorrendo após o teste de *release*.

Métodos e materiais

A partir da leitura de livros de Engenharia de *Software*, Metodologia Científica e pesquisas na Internet, foram identificados e documentados os principais métodos e ferramentas de teste de *software* existentes, dentre outros aspectos de qualidade de *software* e de pesquisa científica importantes.

A partir desta identificação, criou-se um formulário a ser aplicado durante entrevistas realizadas com profissionais desenvolvedores de *software*. Este formulário investiga como se dá o processo de desenvolvimento e de testagem de *software*, através da análise de aspectos como, por exemplo,

equipe, tipo de processo de desenvolvimento e de gestão utilizados, os principais testes utilizados e fases em que eles são aplicados, e ferramentas auxiliares para testagem/gestão utilizadas.

A etapa de coleta de dados foi feita através da aplicação de entrevistas online e do formulário criado. Com esta etapa concluída passou-se, então, para a tabulação e análise dos dados, onde buscou-se tirar conclusões sobre como se dava o processo de desenvolvimento de *software* nas empresas, quais eram os principais métodos de testagem utilizados e as lacunas que poderiam ser preenchidas.

Neste trabalho, optou-se pelo preenchimento do formulário criado durante as entrevistas (e não de forma assíncrona) devido à grande variedade de tipos de processos de desenvolvimento de *software*, de tipos de gestão e de testes de *software* que podem ser utilizados/adaptados. Certas nuances podem ser melhor detectadas e tratadas durante as entrevistas.

Diante do apresentado, percebe-se que este trabalho propôs uma investigação de caráter descritivo e de levantamento de dados. Segundo Wazlawick (2014), a pesquisa descritiva busca obter dados mais consistentes sobre determinada realidade, ela tenta descrever os fatos como eles são e a pesquisa de levantamento consiste em buscar os dados diretamente no ambiente onde eles estão. Ambos os métodos de pesquisa podem fazer uso de técnicas como observações, entrevistas e questionários/formulários para realizar a coleta de dados.

Uma entrevista é um encontro geralmente entre duas pessoas, um entrevistado e um entrevistador, com a finalidade de que o entrevistador obtenha informações a respeito de determinado assunto ou problema, mediante uma conversação de natureza profissional. É um procedimento muito utilizado para investigação e coleta de dados, auxiliando no diagnóstico ou tratamento de um problema (LAKATOS, MARCONI; 2006).

Neste trabalho foi utilizada uma entrevista do tipo padronizada, na qual o entrevistador seguiu o formulário previamente elaborado para a coleta de dados. Segundo Lakatos e Marconi (2006), uma entrevista padronizada ou estruturada é aquela em que o entrevistador segue um roteiro previamente estabelecido, onde as perguntas feitas ao entrevistado são predeterminadas. Ela se realiza de acordo com um formulário elaborado. O motivo da padronização é obter dos entrevistados respostas às mesmas perguntas, permitindo que todas elas sejam comparadas com o mesmo conjunto de perguntas.

Para o desenvolvimento desta pesquisa foi utilizado um computador pessoal próprio, com sistema operacional Windows 10 e acesso à Internet. A partir desse equipamento foi utilizada a ferramenta online Google Docs para a hospedagem do formulário planejado e tabulação inicial dos dados coletados nas entrevistas.

Resultados

O Quadro 1 apresenta o resultado parcial tabulado da coleta de dados. O questionário criado contou com 77 questões que buscaram explorar a descrição do profissional e da empresa na qual trabalha, como era composta a sua equipe de desenvolvimento, o processo de desenvolvimento e os métodos de gestão/qualidade utilizados, e quais eram as técnicas de testagem de *software* utilizadas.

A coleta de dados foi realizada individualmente através de entrevistas online (via Google Meet) com profissionais que atuam a partir de Uruaçu e região. Foram realizadas seis entrevistas. Elas foram realizadas entre 23 e 29 de maio de 2023 e tiveram duração média de 1h e 20min. O perfil dos entrevistados é jovem, com idade média inferior a 30 anos, e com tempo médio de atuação na área de desenvolvimento de cerca de 4,5 anos. A quantidade de amostras é considerada pequena, mas condizente com a realidade de poucas empresas e profissionais desenvolvedores de *software* sediados/residentes na região.

Todos os entrevistados são egressos ou concluintes do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas (ADS) do Instituto Federal de Goiás-Câmpus Uruaçu. Vale ressaltar que apenas um profissional atua em empresa local (o João atua em empresa sediada de Campinorte-GO), os demais profissionais atuam de forma remota para empresas sediadas em outras partes do país.

A ordem dos profissionais presentes no Quadro 1 obedece a uma ordem crescente (da esquerda para a direita) da complexidade do processo de desenvolvimento de *software* no qual o profissional está envolvido.

Quadro 1. Resultado parcial tabulado da coleta de dados.

1	No. Identificação						
2	Nome: (*nomes fictícios)	João*	Pedro*	Marcos*	Maria*	Eduardo*	Alessandro*
3	Qual empresa representa? (*nomes das empresas foram omitidos)	CLT como Programador na ***** desde arço/2021 (empresa brasileira, com 5 funcionários e sediada em Campinorte-GO)	Contratado PJ como Desenvolvedor Sênior na ***** desde setembro/2022 (empresa brasileira, sediada em Porto Alegre-MG, com 26 funcionários)	CLT como Desenvolvedor Pleno na ***** desde janeiro/2022 (empresa brasileira, sediada em Uberlândia-MG, com 35 funcionários)	PJ como Desenvolvedora Júnior na ***** desde janeiro/2022 (empresa brasileira, sediada em Pato Branco -PR, com 46 funcionários)	CLT Fullstack web na ***** desde 2021 (Startup brasileira, sediada em Brasília, com 10 funcionários)	CLT como Engenheiro Sênior Líder na ***** desde 2022 (Multinacional japonesa, com 500 mil funcionários, e filial de Uberlândia-MG)
4	Cidade onde trabalha?	Uruaçu-GO	Uruaçu-GO	Uruaçu-GO	Uruaçu-GO	Uruaçu-GO	Uruaçu-GO
5	Trabalha home-office?	Sim, dedicação exclusiva	Sim, por demanda	Sim, 40h semanais	Sim, por demanda	Sim, por demanda	Sim, 40h semanais

6	Em quais setores atua/presta serviço?	Mantém *softwares* para gestão em mineradoras, atuando nas várias áreas destas empresas	Sofware próprio de gestão de produtos para varejo e gestão/ relacionamento com clientes	Serviço de bot para cobranças bancárias e app para suporte ao produtor rural	Empresa com *software* ERP e *software* para gestão do agronegócio	Logística: Sistema de gestão de picking (separação de produtos). Presta serviço para distribuidores	Desenvolve *softwares* diversos. O Alessandro atua no desenvolvimento Web de soluções bancárias para PJ do banco ****
7	Presta serviços para quais regiões?	Todo o país e alguns países do exterior	Todo o país	Todo o país	Todo o país	Todo o país e iniciando para o exterior	Todo o país e a nível global
9	Há quanto tempo atua com desenvolvimento?	Desde 2021	Desde 2017	Desde 2019	Desde 2022	Desde 2021	Desde 2017
2.EQUIPE DE DESENVOLVIMENTO							
10							
	Qual é o tamanho dela?	5	8	6	8	6	11(mas faz parte de equipe maior, mais de 400 pessoas)
11	Descreva melhor como está organizada a sua equipe:	4 Devs (Desenvolvedores) Fullstack e 1 Gerente de Projeto	6 Devs Fullstack, 1 QA (Quality Assurance) e 1 CTO (Chefe de Operação de Tecnologia)	2 Devs Mobile, 2 Devs Fullstack, 1 Gerente de projetos e 1 QA	1 Gerente de Projeto e 7 Devs Fullstack	2 Devs Fullstack, 2 Devs Mobile, 1 CTO e 1 QA (Quality Assurance)	6 Devs Fullstack, 2 QA, 1 Gerente de Projetos, 1 Product Owner e 1 Tech Lead
12	Principais linguagens de programação e frameworks utilizados?	C# com asp.net core, Javascript, Bootstrap e Jquery	C# com Dot.net e Javascript	C# com .net, Entity Framework, React(mobile)	C# com .net, Angular, Entity e Postman	Web: Ruby on rails (backend) e VUE.JS(-frontend). Mobile: Kotlin	Java com Spring, Framework Angular, Xcode(IOS) e Java para Android
14	Desenvolve em qual plataforma?	Web e Mobile (desenvolvimento terceirizado)	Web e Mobile (desenvolvimento terceirizado)	Web (Marcos atua nele) e Mobile	Desktop, com partes Web	Web (Eduardo atua nele) e Mobile	Web (Alessandro atua nele) e Mobile, com foco em PJ
3.GESTÃO DE *Software*							
15	Tipo de processo de desenvolvimento:	Atividades gerenciadas com Trello (Kanban)	Scrum, com o líder sendo o CTO	Scrum, com líder sendo o Gerente de Projeto	Scrum, com o líder sendo o Gerente de Projeto	Scrum, com o líder sendo o CTO	Srum, sendo o líder o Gerente de Projetos

17	Como funciona o método de gestão de projetos de desenvolvimento?	O gerente de projetos gerencia as atividades pelo Trello(Kanban)	Scrum, com o CTO organizando os sprints e as tasks	Scrum, o Gerente gerencia os projetos e cria as cards/tasks	Scrum, o Gerente gerencia os projetos e cria as cards/tasks	Scrum, o CTO gerencia os projetos e cria as cards/tasks	Scrum com Kanban para o controle das sprints. OKRs para definir os objetivos dos sprints.
18	Equipe tem gerente de projetos?	Sim, 1 Gerente de Projetos	Sim, 1 CTO	Sim, 1 Gerente de Projetos	Sim, 1 Gerente de Projetos	Sim, 1 CTO	Sim, 1 Gerente de Projetos
20	Método utilizado de gestão da configuração e de versionamento:	Git com Gitlab	API do sistema no Git e o restante do código no TFS (Microsoft)	Git, via VisualStudio	Git para projetos internos e Azure para demandas com empresas	Git, através da plataforma BitBucket	Github, com configurações na AWS e datacenters
21	Existe profissional de qualidade?	Não possui	1 profissional de QA	1 profissional de QA	Não possui	1 profissional de QA	2 profissionais de QA
22	Como avalia a qualidade dos *softwares* entregues?	Muito boa, em geral	Em geral, o *software* entregue satisfaz aos clientes	Muito boa, em geral	Boa, em geral	Houve problemas recentes, mas foi inserido um QA para lidar com isso	Muito boa, são vários filtros de qualidade até que o *software* seja entregue para produção
23	Como avalia a satisfação dos clientes quanto aos *softwares*?	Muito boa	Bastante satisfeitos	Não soube informar	Boa, em geral	Boa	Muito boa, o *software* é avaliado pelos próprios usuários
24	Com que frequência os *softwares* são entregues dentro do cronograma e custos definidos?	Na maioria das vezes	Nem sempre os prazos são cumpridos pois, para se iniciar nova sprint, os erros encontrados na versão ante rior têm que ser corrigidos	Em geral, são cumpridos os prazos dos sprints	Na maior parte, as sprints eram finalizadas dentro do prazo estabelecido	Em geral, são cumpridos os prazos dos sprints	Em geral os sprints estouram os prazos previamente definidos

25	Descreva os principais pontos fortes do seu processo de desenvolvimento e de qualidade do *software*:	Funções bem definidas, atividades detalhadas, boa comunicação com o gerente e ferramentas	Equipe com boa comunicação e colaboração, profissional de QA muito criterioso, tasks muito detalhadas	Equipe com boa comunicação, tasks bem detalhadas e responsabilidades bem divididas	Equipe com boa comunicação, responsabilidades e funções bem definidas	Equipe com ótima comunicação, com bom processo de testagem, tasks bem detalhadas e de fácil em tendimento	Equipes bem definidas, com tasks detalhadas e processo bem definido e organizado
26	Descreva os principais pontos fracos do seu processo de desenvolvimento e de qualidade do *software*:	Falta de testes unitários e de padrões de criação de código	A plataforma TFS dificulta os aspectos de versionamento de código	Os devs não possuem acesso ao ambiente de homologação para descobrirem quais foram os erros encontrados	Organização das sprints e tasks não era eficiente, tasks com poucos deta lhes. Falta de servidor interno para testes	Profissional de QA sobrecarregado de funções (o que está sendo resolvido)	Processo burocrático, devido à necessidade do negócio
27	Satisfação da equipe quanto ao processo de desenvolvimento:	Bem satisfeita	Bem satisfeita	Bem satisfeita	Bem satisfeita	Bem satisfeitos, bom ambiente e processo organizado	50% satisfeita, devido à burocracia e qualidade exigida pelo negócio
4.TESTES DE *Software*							
29	Em linhas gerais, como se dá o planejamento e execução dos testes de *software* nos projetos:	Devs reali zam apenas testes manuais, não realizam testes automatizados	Não são codificados testes, são realizados testes manu ais na medida em que o código é feito.	Não são feitos testes automatizados, mas são realizados testes manuais	Eram realizados apenas testes manuais pelos devs	Para tasks novas são criados testes auto matizados. Para corre ções, nem sempre eles são criados	Para as tasks são realizados testes automatizados
30	Pela equipe, o quão impo rtante é test ar *software*?	Importante e obrigatório	Importante e obrigatório	Muito importante e obrigatório	Muito importante e obrigatório	Primordial e obrigatória	Muito importante e obrigatório
31	Qual é a estimativa de tempo necessários por projeto para plane jar, criar, executar e documentar testes:	Para cada task, entorno de 15% do tempo de desenvolvimento é gasto para executar os testes	Para cada task, entorno de 15% a 20% do tempo de desenvolvimento é gasto para executar os testes	Para cada task, entorno de 20% do tempo de desenvolvimento é gasto para executar os testes	Para cada task, entorno de 20% do tempo de desenvolvimento é gasto para executar os testes	Para cada task, entorno de 30% do tempo de desenvolvimento é gasto para criar os testes e testar	Para cada task, entorno de 30% do tempo de desenvolvimento é utilizado para criar os testes e executá-los

32	Como e com que frequência ocorrem as inspeções e revisões de documentos e de código dos projetos:	Os devs frequentemente revisam seus códigos. Em geral, uma task é realizada por um único dev.	Não são realizadas inspeções e revisões	Não são realizadas inspeções e revisões	1 Fullstack mais experiente sempre revisava os códigos dos demais devs	O código criado pelos devs é revisado pelo CTO antes que ele suba para o profissio nal de QA	Os devs líderes fazem inspeções e revisões do código antes do seu envio para o ambiente piloto
33	Pela equipe, qual é a importância de se realizar inspeções e revisões?	Importante	Diante do processo estabelecido, eles não são vistos com importância	Importante, mas a empresa e a equipe não adotaram essa prática	Importante, eram sugeridas muitas otimizações de código	Primordial e obrigatória	É uma etapa importante, pois ajuda a garantir a qualidade do código
4.1.Teste unitário							
34	Eles são em pregados?	Nunca	Nunca	Nunca	Sempre, mas realizados de forma manual	Sempre. Exis te um teste unitário para cada método importante	Sempre
35	Como os testes unitários são empregados nos projetos de desenvolvimento:	Não se aplica	Não se aplica	Não se aplica	Os testes manuais eram feitos de acordo com a necessidade da task	Os testes automatizados ficam registrados na máquina dos devs e na plataforma BitBicket	Os testes tem que cobrir funções e 95% do código de todos os arquivos
36	Quais são as diretrizes gerais utiliza das para a criação de testes unitários:	Não se aplica	Não se aplica	Não se aplica	Os testes manuais eram feitos de acordo com a necessidade da task	Eles são criados para métodos mais críticos e com manipulação de dados	Os testes são criados de acordo com a necessidade da task
37	Eles são auto matizados?	Não se aplica	Não se aplica	Não se aplica	Não	Sim	Sim
4.2.Teste de componente							
40	Eles são em pregados?	Nunca	Nunca	Nunca	Maioria das vezes	Nunca	Sempre
41	Como eles são empre gados nos projetos?	Não se aplica	Não se aplica	Não se aplica	Eram feitos para partes mais críticas do sistema	Não se aplica	Todos os componentes e interfaces são testados

42	Diretrizes utilizadas para a criaçã o de testes de compo nentes:	Não se aplica	Não se aplica	Não se aplica	Os testes são criados de acordo com a necessidade da task	Não se aplica	Os testes são criados de acordo com a necessidade da task
4.3.Teste de sistema							
46	Eles são em pregados?	Sempre	Sempre	Sempre	Sempre	Sempre	Sempre
47	Descreva como os testes de sistema são empregados nos projetos de desen-volvimento:	Sempre que uma feature é criada ou alterada, testes de sistema básicos são feitos pelos devs na interface do sistema	Sempre que uma feature é criada ou alterada, testes de sistema básicos são feitos pelos devs na interface do sistema	Sempre que uma feature é criada ou alterada, testes de sistema básicos são feitos pelos devs na interface do sistema	Sempre que uma feature é criada ou alterada, testes de sistema básicos são feitos pelos devs na interface do sistema	Sempre que uma feature é criada ou alterada, testes de sistema básicos são feitos pelos devs na interface do sistema	Sempre que uma feature é criada ou alterada, testes de sistema básicos são feitos pelos devs na interface do sistema
48	Diretrizes utilizadas para a criação de testes de sistema:	Os testes são criados de acordo com a necessidade da task	Baseado na task que gerou a alteração no sistema	Baseado na task que gerou a alteração no sistema	Os testes são criados de acordo com a necessidade da task	Baseado na task que gerou a alteração no sistema	Os testes são criados de acordo com a necessidade da task
51	Pela equipe, qual é a importância de se testar *software* via testes de sistema:	Obrigatório, sendo este basicamente o tipo de teste feito pela equipe	Importante e obrigatório, pois evita que erros sejam passados adiante	Essencial	Essencial	Muito importante, pois permitem encontrar erros ainda pela equipe	Essencial
52	Procedimentos no caso de não apro vação nesses testes:	O próprio dev realiza as correções necessárias	O próprio dev realiza as correções necessárias	O próprio dev realiza as correções necessárias	O próprio dev realiza as correções necessárias	O próprio dev realiza as correções necessárias	O próprio dev realiza as correções necessárias.
4.4.Teste de release							
53	Eles são em pregados?	Nunca	Sempre	Sempre	Metade das vezes	Sempre	Sempre
54	Possui profissional para realizar testes de release?	Não existe	Sim, existe 1 profissio-nal de QA	Sim, existe 1 profissio-nal de QA	Sim, 1 con-sultor ex-terno (que representa o cliente)	Sim, existe 1 profissio-nal de QA	Sim, 2 pro-fissionais de QA
55	São realiza dos por profissio-nais que partici param do desenvolvi-mento?	Não se aplica	Não	Não	Não	Não	Não

56	Descreva como os testes de release são empregados nas versões dos sistemas:	Não se aplica	O profissional de QA entende a task nova e realiza testes manuais, criando também testes automatizados. Cada sprint gera uma versão do *software*. A cada versão nova todos os testes automa tizados são executados	O profisisonal de QA entende a task nova e realiza testes manuais vinculados a ela. Esse procedimento é feito no ambiente web de homologação. O app já se conecta nele para testes.	O consultor recebia o release em um ambiente de homologação do cliente (em geral, para cada sprint era gerado um release do sistema) e realizava os testes de verificação das funcio na lidades	O profissional de QA entende a task nova e realiza testes vinculados a ela no ambiente de homologação e com o sistema de ponta a ponta (caso julgue necessário)	O profissional de QA recebia o release em um ambiente de homologação(em geral, para cada sprint era gerado um novo release) e realizava os testes de verificação das funcionalidades
60	Pela equipe, qual é a importância de se testar *software* via testes de release:	Não é importante, diante do processo de desenvolvimento estabelecido	Muito importante, pois ajuda a evitar que erros sejam colocados em produção	Muito importante, pois evita que erros vão para o ambiente de produção	Muito importante, pois evita que erros vão para o ambiente de produção	Muito importante, pois evita que erros sejam colo ca dos em produção	Muito importante, pois ajuda a evitar que erros sejam colocados em produção
61	Quais são os procedimentos adotados no caso de não aprovação do *software* nos testes de release:	Não se aplica	Caso o QA, caso encontre, no ambiente de homologa ção, um erro em uma task e/ ou sprint, ele notifica o dev através de um comentá rio na task e/ ou sprint, solicitando correções	O QA acrescenta na task as evidências de problemas e a devolve para o dev	O consultor retornava via e-mail os erros, o gerente verificava e solicitava aos devs as correções (via adição na task dos problemas encontrados)	O sotware não é liberado para produção e a task é devolvida para os devs, com as considerações sobre os problemas encontrados	O QA acrescenta na task as evidências de problemas e a devolve para o dev

62	Descreva como é feita a entrega do *software*/ release aos clientes/ usuários:	Após os testes de sistema pelo dev, o gerente sobe o release para o servidor web de homologação presente na Infra do cliente. Após aprovação pelo cliente, o release é configurado no servidor de produção do cliente	Após passar pela homologação e teste do usuário, a nova versão do sistema é disponibilizada no servidor web de produção (em horários não críticos, geralmente às 6h da manhã).	Após aprovação pelo QA no ambiente de homologação, o release é implantado no servidor web de produção	Após a aprovação do consultor e do cliente(usuário final), o release era implantado no ambiente de produção do cliente	Após a aprovação pelo QA, o release é implantando no servidor web de produção (geralmente pela manhã, pois a maior parte do uso do sistema ocorre durante a noite)	Após aprovação pelo QA no servidor de homologação, o código é disponibilizado no servidor piloto, onde usuários testam o sistema de forma controlada. Se aprovado por eles, o código é subido para produção e liberado para os clientes.
4.5.Teste de usuário							
63	Eles são em pregados?	Sempre	Na maioria das vezes. Os testes são acompanhados pelo CTO e às vezes pelo QA.	Nunca	Sempre	Para funcionalidades críticas são feitos testes com usuários	Sempre
64	Descreva como os testes de usuário são empregados nas versões dos sistemas:	Após os testes de sistema pelo dev, o gerente sobe o release para o servidor web de homologação presente na Infra do cliente. Após aprovação pelo cliente, o release é configurado no servidor de produção do cliente	Após passar pela homologação, a nova versão do sistema é disponibilizada em um servidor web de testes/pré-produção, onde os usuários testam o *software* por um período.	Não se aplica. Entretanto, o gerente de projetos sempre apresenta protótipos de telas do sistema ao usuário após a definição da funcionalidade.	Os usuários finais participavam apenas na etapa de homologação	Para funcionalidades de grande impacto/ críticas, é disponibilizado ao usuário o link para o ambiente de homologação, onde ele testará a funcionalidade	Em um servidor piloto, usuários selecionados testam a nova versão do sistema de forma controlada. Se aprovada por eles, esta nova versão também é testada por uma equipe de testadores que apoiam a alta gestão do banco.
66	Testes do tipo alfa são usados?	Não	Não	Não	Não	Não	Não

68	Testes beta são usados?	Não	Não	Não	Não	Não	Não
70	Testes de aceitação são usados?	Não	Não	Não	Não	Não	Não
72	Pela equipe, qual é a importância de testar soft ware através de testes de usuário:	Fundamental e obrigatório, pois eles tinham que aceitar o rele ase disponibilizado	Muito importante	Não se aplica	Fundamental, pois eles tinham que aceitar o release disponibilizado	Fundamental	Fundamental, pois eles tinham que aceitar o release disponibilizado
73	Quais são os procedimentos adotados no caso de não aprovação do *software* nos testes de usuário:	O gerente é notificado, através das evidências fornecidas em um documento .doc e comunica o dev para que sejam feitas as adequações	Caso o problema seja um erro, o QA atua para simulá-lo e repassá-lo para os devs. Caso seja uma melhoria não solicitada, o CTO atua para rever a funcionalidade	Não se aplica. Os usuários fazem uso do sistema já em ambiente de produção. Caso um erro seja reportado, o QA atua para simulá-lo e repassá-lo para os devs	O consultor retornava via e-mail os erros repassados pelo cliente, o gerente verificava e solicitava aos devs as correções	O CTO se reúne com os usuários para detalhamento dos problemas encontrados e novas solicitações/orientações são repassadas para a equipe de devs	Caso algum problema seja encontrado, o QA que acompanha os testes dos usuários acrescenta na task as evidências de problemas e a devolve para o dev

Fonte: Próprio autor (2024).

O tópico a seguir apresenta as principais observações e conclusões feitas a partir da análise dos dados presentes no Quadro 1.

Discussões

A partir dos dados tabulados no Quadro 1, houve o entendimento de como funciona os processos de desenvolvimento de *software* nas empresas nas quais os profissionais atuam. Pode-se observar que o processo de desenvolvimento e de testagem de *software* no qual está envolvido cada profissional é único. Ou seja, cada empresa adaptou seu processo de acordo com a sua realidade/necessidade. Esta é uma característica que já era esperada, haja visto que empresas desenvolvedoras podem atuar em nichos de mercado muito distintos e com demandas específicas, além de poderem optar por dezenas de métodos, técnicas e padrões para compor os seus processos de desenvolvimento e testagem de *software*.

No Quadro 1, pode-se perceber essa diversidade: de processos simples e que exigem poucas etapas de testagem a processos complexos/burocráticos e que exigem várias camadas de testagem de *software*.

Com relação às seis empresas em que atuam os profissionais, percebe-se que cada uma atua em uma área/setor específico (conforme questão nº 6 do Quadro 1). Todas atuam a nível

nacional, sendo que três delas possuem atuação também no exterior. Cinco das seis empresas são empresas de pequeno/médio porte (a empresa na qual o profissional Alessandro atua é uma multinacional de grande porte).

A questão nº 10 do Quadro 1 deixa claro que todas as esquipes de desenvolvimento nas quais os profissionais estavam inseridos eram pequenas (o profissional Alessandro atua na maior equipe, contando com 11 profissionais. Vale ressaltar, entretanto, que essa equipe é apenas um segmento de uma equipe de desenvolvimento bem maior). As questões nº 12 e 14 evidenciam que quatro das equipes de desenvolvimento utilizam a linguagem C# como principal linguagem de programação e que cinco equipes têm como foco principal o desenvolvimento Web.

Ao se observar os processos de desenvolvimento utilizados pelas empresas, percebe-se que todas elas equipes utilizam métodos ágeis[3] para desenvolvimento de *software* em incrementos, sendo que cinco delas utilizam o método de desenvolvimento/gestão Scrum, organizando as suas atividades em sprints, tasks e/ou cards (conforme questões nº 15 e 17 do Quadro 1). Sabe-se que os métodos ágeis priorizam a entrega rápida de incrementos de *software* e dos benefícios do seu emprego em equipes desenvolvedoras de pequeno/médio porte. A questão nº 18 do Quadro 1 mostra outro aspecto importante: todas as equipes possuem pelo menos um profissional gestor dedicado, seja ele um Gerente de Projetos ou CTO. A questão nº 21 mostra que quatro equipes possuem um profissional dedicado de Gestão de Qualidade. Sabe-se da importância desse profissional na avaliação da qualidade dos *softwares* que são entregues aos clientes.

De forma geral, os profissionais informaram que a qualidade dos *softwares* entregues é boa e que seus usuários/clientes avaliam de forma muito positiva o grau de satisfação com relação ao uso dos *softwares* (conforme questões nº 22 e 23 do Quadro 1). Estes profissionais também deixam claro que enxergam muitos pontos positivos nos processos de desenvolvimento que utilizam (como equipe com boa comunicação, bem gerida com e funções bem definidas) e que são satisfeitos com eles (conforme questões nº 25 e 27 do Quadro 1). Entretanto, o profissional Alessandro relatou que considera o processo de desenvolvimento no qual participa complexo e burocrático (mas que entende que isso é uma necessidade do negócio).

Dando início à análise das questões envolvendo diretamente os testes de *software*, todos os entrevistados informaram que testar o *software* que está sendo desenvolvido é uma atividade muito importante e obrigatória, e que eles gastam entorno de 15% a 30% do tempo do desenvolvimento para testar *software* (conforme questões nº 30 e 31 do Quadro 1). Vale ressaltar também que quatro das equipes de desenvolvimento realizam a inspeção/revisão do código criado e que essa atividade também é considerada muito importante (conforme questões nº 32 e 33 do Quadro 1).

Sobre os testes unitários, três profissionais informaram que eles não são usados, um informou que os utilizam apenas de forma manual e apenas duas equipes informaram que sempre programam estes tipos de testes e que os executam de forma automatizada (conforme questão nº 34 do Quadro 1). Esta informação surpreendeu os entrevistadores, pois sabe-se da importância que uma base de testes unitários automatizados tem para encontrar eventuais erros e colaborar para a garantia da qualidade do *software* desenvolvido.

[3] Maiores detalhes em: https://www.alura.com.br/artigos/o-que-e-metodologia-agil

Sobre os testes de componentes, apenas duas equipes informaram que estes tipos de testes fazem parte de sua rotina de desenvolvimento (conforme questão nº 40 do Quadro 1). Diante do tamanho reduzido das equipes e do emprego por elas de métodos ágeis, este era um resultado de certa forma esperado pelos entrevistadores.

Sobre os testes de sistema, todos os profissionais informaram que sempre fazem uso deste tipo de teste e que ele é uma etapa essencial do desenvolvimento, com eles realizando testes nas interfaces dos sistemas na medida em que o desenvolvimento da task/card avançava e efetuando as correções que fossem necessárias (conforme questões nº 46, 47, 51 e 52 do Quadro 1).

Com relação aos testes de release, cinco das seis equipes realizam este tipo de teste, com elas contando com um profissional de QA ou equivalente para planejá-lo ou executá-lo (conforme questões nº 53 e 54 do Quadro 1). Esta informação surpreendeu os entrevistadores pois, apesar das equipes serem de tamanho pequeno, elas têm o cuidado de terem um profissional dedicado às atividades de qualidade dos *softwares* desenvolvidos. Vale ressaltar que esses profissionais não participavam ativamente da programação dos sistemas e que, em geral, o release gerado a cada sprint era testado pelo profissional de QA em um ambiente de homologação (conforme questões nº 55 e 56 do Quadro 1). A aprovação da release pelo profissional de QA é uma etapa essencial na maioria das equipes, sendo essa aprovação necessária para implantação da release no ambiente de produção ou para que ela avançasse para a próxima etapa de testes (conforme questão nº 62 do Quadro 1).

Sobre os testes de usuário, eles são empregados por cinco das seis equipes, sendo por elas considerada uma etapa fundamental da testagem (conforme questões nº 63 e 72 do Quadro 1). Entretanto, pelas respostas obtidas na questão nº 64, percebe-se que os usuários não possuem participação durante a etapa de programação das tasks/cards. Basicamente, os usuários testam o sistema que já foi homologado. Sabe-se que os próprios métodos ágeis preconizam a participação ativa dos usuários durante todas as fases de desenvolvimento do *software*, desde a definição dos requisitos até a etapa de criação do código e entrega do sistema. Os testes do tipo alfa e beta não são realizados pelas equipes (conforme questões nº 66 e 68 do Quadro 1). Esta informação já era esperada, haja visto que os *softwares* desenvolvidos pelas equipes eram customizados para os seus usuários e que testes do tipo alfa e beta geralmente são empregados para *softwares* do tipo de prateleira.

Considerações Finais

Em uma tentativa de se obter uma "visão geral" sobre como empresas e profissionais desenvolvedores de *software* de Uruaçu-GO e região estão testando os *softwares* por eles criados/mantidos, esse trabalho realizou uma coleta de dados com seis profissionais que atuam a partir de Uruaçu (a quantidade de amostras é pequena, mas condizente com a realidade de profissionais que atuam na região).

Para que a coleta de dados fosse feita de forma estruturada, foi criado um formulário que foi respondido pelos profissionais durante as entrevistas online que foram realizadas. O

formulário era composto por 77 questões, que abrangiam aspectos como o processo de desenvolvimento utilizado pelas equipes, gestão/qualidade de *software* e, obviamente, as técnicas de teste de *software* empregados (dentre as inúmeras disponíveis).

Com os dados coletados e tabulados, foi realizada uma análise dos principais aspectos encontrados. Conseguiu-se, então, obter o entendimento sobre como os processos de desenvolvimento ocorriam. Observou-se que cada empresa adaptou o seu processo de acordo com a sua realidade/necessidade e que os processos variaram do simples ao complexo/burocrático. Em linhas gerais, os profissionais estavam envolvidos em equipes pequenas e que faziam uso de métodos ágeis para o desenvolvimento de *software* em incrementos. Os entrevistados se mostraram satisfeitos com seus processos de desenvolvimento, destacando a boa comunicação entre a equipe, boa gestão e funções bem definidas.

Ainda como aspectos positivos, observou-se que a quase totalidade das equipes possuíam profissionais dedicados à gestão de projetos e de garantia da qualidade, e que vários tipos de teste eram empregados nos processos, como os testes de sistema, de release e de usuário (todos os entrevistados destacaram a importância de se testar o *software* em seus vários estágios de desenvolvimento). Estas boas práticas contribuíam para a boa qualidade dos *softwares* entregues pelas equipes.

Dentre as lacunas identificadas, verificou-se que apenas a metade das equipes realizavam a programação de testes unitários automatizados e que os usuários (interessados pelos *softwares*) possuíam pouca interação com as equipes durante as etapas de codificação do sistema. Estes dois aspectos são importantes para a mitigação de erros e melhoria da qualidade dos *softwares* entregues e representam sugestões de melhorias que podem ser integradas aos processos de desenvolvimento analisados.

O planejamento e execução desse trabalho de pesquisa foi desafiador. Muito esforço foi necessário para o entendimento dos tipos de testes existentes e elaboração do formulário de forma a contemplar os elementos que seriam necessários para a coleta de dados. A realização das entrevistas e posterior tabulação de todos os dados coletados exigiu grande esforço, mas, ao final, esse processo foi bastante proveitoso e gratificante, pois conseguiu-se alcançar os objetivos desejados com esse trabalho.

Referências

LAKATOS, E. M.; MARCONI, M. A. **Fundamentos de metodologia científica.** 6. ed. São Paulo: Atlas, 2006.

MARTINS, M. D. C. **Testes de *software.*** Rio de Janeiro: SESES, 2016.

PRESMANN, R. S.; MAXIM, B. R. **Engenharia de *Software.*** 9. ed. Porto Alegre: Bookman, 2021.

SCHACH, S. R. **Engenharia de *Software:* Os Paradigmas Clássico e Orientado a Objetos.** 7. ed. São Paulo: McGraw-Hill, 2009.

SOMMERVILLE, I. **Engenharia de *Software.*** 10. ed. São Paulo: Pearson Prentice Hall, 2018.

WAZLAWICK, R. S. **Metodologia de pesquisa para ciência da computação.** 2. ed. Rio de Janeiro: Elsevier, 2014.