UNIVERSIDADE FEEVALE

MARISTELA OLIVEIRA DE PAULA

AUTOMATIZAÇÃO DE TESTE DE UI EM AMBIENTE ÁGIL: A PROPOSTA DE UM MODELO DE PROCESSO

Novo Hamburgo

2019

MARISTELA OLIVEIRA DE PAULA

AUTOMATIZAÇÃO DE TESTE DE UI EM AMBIENTE ÁGIL: A PROPOSTA DE UM MODELO DE PROCESSO

Trabalho de Conclusão de Curso, apresentado como requisito parcial à obtenção do grau de Bacharel em Sistemas de Informação pela Universidade Feevale.

Orientador: Prof. Dr. Adriana Neves dos Reis

Novo Hamburgo

2019

MARISTELA OLIVEIRA DE PAULA

Trabalho de conclusão do Curso Sistemas de Informação, com título **AUTOMATIZAÇÃO DE TESTES DE UI EM AMBIENTE ÁGIL: A PROSPOSTA DE UM MODELO DE PROCESSO**, submetido ao corpo docente da Universidade Feevale, como requisito necessário para obtenção do Grau de Bacharel em Sistemas de Informação.

Inclui Termo de Consentimento da Empresa/Entrevistado: [ ] Sim; [ ] Não

Aprovado por:

_____

Orientador: [Nome do orientador]

_____

Professor avaliador

_____

Professor avaliador

Agradecimentos

Agradeço a todos aqueles que, de alguma forma, contribuíram para a realização deste trabalho. Em especial à professora Dr. Adriana, por todo o apoio, dedicação e incentivo dado durante o desenvolvimento da pesquisa.

Aos meus amigos e familiares que souberam compreender minha ausência e sempre me apoiaram em toda a minha fase da graduação.

E aos meus pais que, com muito esforço em vida, foram os principais incentivadores desta trajetória, e agora observam do céu a conclusão de mais uma etapa.

Muito Obrigada!

RESUMO

Nos últimos anos, progressivamente, a indústria de desenvolvimento de software tem aderido às Metodologias Ágeis com o propósito de responder cada vez mais rápido às mudanças de requisitos do cliente e, consequentemente, realizar entregas de maior valor. Diante disto, há a motivação em utilizar a automatização de testes em seus processos, e, embora seja consolidada a ideia inicial em priorizar a automatização dos testes unitários e de serviços neste contexto, tem-se ainda uma oneração de tempo quanto aos testes manuais voltados para interface gráfica do usuário, principalmente ao olhar de testes regressivos a cada ciclo de entrega. Assim, fazendo uso do método de pesquisa *Design Science Research* (DSR), inicialmente consolidou-se, através de uma pesquisa de campo, a conscientização da problemática da pesquisa elencando os principais fatores ligados às dificuldades na automatização de testes de interface de usuário. A partir disto, foram levantados critérios e requisitos para desenvolver a pesquisa, evidenciando através da literatura propostas e práticas que tangem ao processo de desenvolvimento de software em conjunto às abordagens em contexto ágil. Logo, apresenta-se um Modelo de Processo de Automatização de Testes de Software para *User Interface* – (MPATS – UI) voltado a equipes que empregam Metodologias Ágeis com gerenciamento Scrum. Para validação da utilidade prática da proposta, aplicaram-se métodos para uma análise qualitativa, através de uma segunda pesquisa de campo e com entrevistas direcionadas a especialistas da área de qualidade e software, automação de testes e Scrum. Assim, ficou reconhecida a viabilidade de uso do artefato gerado por este trabalho ao contexto prático, consolidado à contribuição de pesquisa científica.

Palavras-chave: Automação; Automatização de Teste; Scrum; Teste de Software; Metodologia Ágil.

ABSTRACT

Over the past few years, the software development industry has progressively adhered to Agile Methodologies in order to respond ever faster to changing customer requirements and thereby deliver higher value deliverables. Given this, there is a motivation to use test automation in their processes and although the initial idea to prioritize unit and service automation in this context is consolidated, there is still a time burden on manual testing for graphical user interface, especially when looking at regressive tests with each delivery cycle. Thus, making use of the research method Design Science Research (DSR), it was initially consolidated, through a field research, the awareness of the research problematic, listing the main factors related to the difficulties in the automation of user interface tests. From these information, criteria and requirements have been brought up to develop the research, evidencing through the literature proposed and practices that incorporate the software development process along with the agile context approaches. Thus, a User Interface Software Testing Automation Process Model (MPATS - UI) is presented to teams that employ Agile Scrum Management Methodologies. To validate the practical utility of the proposal, methods have been applied for a qualitative analysis, through a second field research and interviews with experts on quality and software, test automation and Scrum. Thus, it was recognized the viability of using the artifact generated by this work to the practical context, consolidated to the contribution of scientific research.

Key words: Agile Methodology; Automation; Scrum; Test Automation; Software Testing; Test Automation.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE SIGLAS

ATS	Automatização de Teste de Software
ATDD	*Acceptance Test-Driven Development*
BDD	*Behavior Driven Development*
DDD	*Domain Driven Design*
DSR	*Design Science Research*
GUI	*Graphical User Interface*
PO	*Product Owner*
MPATS – UI	Modelo de Processo de Automatização de Testes de Software para *User Interface*
QA	*Quality Assurance*
TDD	*Test Driven Development*
UI	*User Interface*
XP	*eXtreme Programming*

SUMÁRIO

1 INTRODUÇÃO

Testes de software agregam importância no desenvolvimento e na entrega do produto em relação à percepção de qualidade. Segundo Rios (2005), os testes têm por objetivo reduzir os riscos que possam prejudicar o negócio ligados a falhas do software. Para Naik e Tripathy (2008), além de proporcionar valor estratégico comercial, a atividade de teste deve estar alinhada com a missão e os objetivos da empresa.

Entretanto, a crescente adoção das metodologias ágeis no processo de desenvolvimento de software visando atender à solicitação de um mercado que exige entregas cada vez mais rápidas, conforme destacado na pesquisa do VersionOne (2019), transpõe para área de teste de software o desafio de testar mais rápido sem comprometer a qualidade. Contudo, a tarefa de testes é árdua para as equipes, pois, a sua execução tende a consumir uma grande fatia de tempo e de esforços técnicos dentro de um processo de desenvolvimento de software (PRESSMAN e MAXIM, 2016).

E, olhando para o contexto ágil com entregas rápidas que visam maior valor ao cliente, é possível notar que para garantir ambos requisitos faz-se necessário a otimização de tempo no processo com soluções que impactem diretamente na garantia da qualidade de entrega e que gerem rápidos *feedbacks* à equipe. Para tal, têm-se então a alternativa de automatizar processos ligados diretamente à qualidade, como o teste funcional, que na maioria das vezes persiste em execuções manuais via interface gráfica do usuário, em inglês *Graphical User Interface* (GUI) ou ainda referenciado como *User Interface* (UI), termo mais comumente utilizado e adotado neste trabalho.

Testes de regressão consistem em retestar uma aplicação após sua implementação sempre que alguma mudança é realizada no software, com o intuito de garantir que as funcionalidades já desenvolvidas anteriormente não tenham sido afetadas pelas alterações implementadas (BASTOS et al., 2007). Pelas boas práticas de engenharia e qualidade de software, este tipo de teste deve sempre ser executado ao anteceder uma entrega, mas na prática, infelizmente, tendem a não serem sempre executados, justamente pelo curto tempo no escopo do projeto. Peres (2016, p. 39) defende que “Para testes de regressão e desempenho não há dúvida que o automatizado seja o mais adequado. ” e Sommerville (2007, p. 147) salienta “Esta forma é geralmente mais rápida que o teste manual, especialmente quando envolve teste de regressão - reexecução de testes anteriores […]”.

E ao pensar em ciclos curtos para o desenvolvimento do software com abordagens ágeis, por exemplo, a realização de testes de regressão a cunho manual torna-se praticamente

impossível. Em consequência, reduz a cobertura de teste e impacta diretamente na qualidade da entrega e no aumento de risco no produto para o negócio. Ainda, um dos princípios adotados do Manifesto Ágil é "Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado." (GOMES, 2016, p. 3). Dentro desta cultura é praticamente indispensável o uso de Automatização de Teste de Software (ATS). De acordo com Cohn (2011), a ATS deve ser considerada uma parte incorporada ao processo de desenvolvimento de software em uma entrega contínua, assim dando suporte para que a equipe tenha sucesso na *Sprint*, alcançando os valores rapidamente.

Conforme pesquisa realizada pela QASymphony e a TechWell Survey (2018), em torno de 72% das empresas já praticam a automatização de teste, sendo que dessas 76% são adeptas às Metodologias Ágeis. Ainda que empresas adotem a ATS, ela consiste em ser um desafio quanto à sua implementação com sucesso, principalmente quando voltada à interface gráfica. Para Mulchandani (2013), esse insucesso está ligado à inexistência de um processo guiado para tal dentro das equipes. Crispin e Gregory (2009) chamam a atenção para o desestímulo das equipes gerado pela ausência de métricas no decorrer do processo ou o emprego de relatórios. Costa (2004) enfatiza ainda sobre a falta de consciência quanto à similaridade de um processo de desenvolvimento de software ao de automação de testes, a qual contribui para o fracasso de uma boa implementação.

Na literatura é possível identificar práticas para o processo de desenvolvimento de software voltado ao contexto ágil e orientadas ao teste, duas delas são o *Test Driven Development* (TDD) e o *Acceptance Test-Driven Development* (ATDD). A primeira orienta a adoção de uma abordagem na qual os testes automatizados de unidade sejam escritos antes mesmo do real desenvolvimento, que por sua vez, será realizado com iterações baseadas nestes testes (BECK, 2010). Já o ATDD, embora siga a mesma premissa sobre a ordem da escrita de testes e desenvolvimento, dissemina a prática de envolver diferentes perspectivas (desenvolvedores, testadores e clientes) para a escrita dos testes de aceitação (GARTNER, 2013).

Contudo, ambas as práticas não indicam especificações quanto ao processo para implementação de ATS para UI, assim como *frameworks* Scrum e XP (*eXtreme Programming*), que, embora atendam às metodologias ágeis a nível de gerenciamento de processo e valorizem a utilização de testes automatizados no desenvolvimento, não propõem um modelo para conduzi-la no processo quanto à implementação. Diante de tal, assume-se, então, como questão de pesquisa: "Como seria um modelo de processo para automatização

de teste de interface gráfica em ambiente ágil quanto à sua implementação para que contribua com a qualidade e uma entrega de software contínua? ”.

1.2 OBJETIVOS

1.2.1 OBJETIVO GERAL

O objetivo geral desta pesquisa é elaborar um modelo de processo para implementação de testes automatizados em um ambiente que emprega metodologias ágeis no desenvolvimento de software. Assim, espera-se contribuir para uma adoção da automatização de teste de interface gráfica de usuário que atenda aos princípios da agilidade e favoreça a efetividade e a qualidade do processo.

1.2.2 OBJETIVOS ESPECÍFICOS

- Fazer um levantamento sobre limitações e expectativas realistas sobre automação de testes em equipes reais;
- Sistematizar os conceitos básicos quanto à automatização de testes de software existentes na literatura;
- Mapear os conceitos e características da metodologia ágil dentro do desenvolvimento de software;
- Desenvolver o modelo de processo para automatização de teste em ambiente ágil;
- Aplicar uma pesquisa na área de interesse para a posterior análise de viabilidade de adoção do modelo proposto.

1.3 ORGANIZAÇÃO DO TRABALHO

O presente trabalho está estruturado de forma que atenda seus objetivos específicos com clareza quanto à sua leitura e o rigor da metodologia de pesquisa empregada. Para tal, a organização do texto desta pesquisa tem a seguinte estrutura:

- **Capítulo 2:** apresenta a metodologia utilizada na pesquisa e o método de condução da mesma utilizado na abordagem da DSR para alcançar o rigor metodológico.

- **Capítulo 3:** apresenta a conscientização do problema e a análise do resultado obtido através da aplicação de instrumento de pesquisa qualitativa, caracterizando a primeira e a segunda etapas da metodologia;
- **Capítulo 4:** apresenta o referencial teórico desta pesquisa, o qual sistematiza os conceitos básicos da automatização de testes de software existentes na literatura, as características da metodologia ágil no desenvolvimento de software, assim como técnicas de desenvolvimento e modelos de gerenciamento de processos mais utilizados no contexto ágil. Este capítulo corresponde à terceira etapa da metodologia empregada.
- **Capítulo 5:** apresenta uma revisão sistemática de publicações com tema relevante à presente pesquisa, buscando contribuições a partir de observações sobre problemáticas e/ou fatores de sucesso relatados nas mesmas. Este capítulo complementa a terceira etapa da metodologia.
- **Capítulo 6:** apresenta, como quarta etapa da metodologia, o desenvolvimento detalhado do artefato correspondente à solução satisfatória para o objetivo desta pesquisa.
- **Capítulo 7:** abrange a quinta etapa da metodologia com a apresentação da construção e análise da avaliação do artefato realizada através de um instrumento de pesquisa aplicado ao público de interesse e o estudo qualitativo proposto por entrevistas com especialistas da área.
- **Capítulo 8:** apresenta as considerações finais relacionadas à pesquisa, tais como, resultados obtidos, limitações e sugestões de trabalhos futuros, caracterizando assim o fechamento com a última etapa da metodologia.

2 METODOLOGIA

Este trabalho é classificado como pesquisa aplicada por sua natureza, com finalidade de aplicação dos conhecimentos com a construção de uma proposta de modelo que suportará sua adoção em ambientes conforme definido na pesquisa (PRODANOV; FREITAS, 2013). A condução desta pesquisa emprega o método científico *Design Science Research* (DSR), que objetiva desenvolver um artefato que possibilite gerar soluções satisfatórias e práticas para um determinado problema. (DRESCH; LACERDA; ANTUNES JR, 2015). Para Pimentel (2017), o artefato possibilita verificar as competências para o qual direciona o seu desenvolvimento, produzindo conhecimento como ciência.

Um artefato é considerado a junção entre um ambiente interno, composto por componentes do próprio artefato, com um ambiente externo, que caracteriza as condições que amparam o funcionamento deste artefato. Ou seja, componentes internos, quando organizados, propõem atingir objetivos específicos de ambientes externos (DRESCH; LACERDA; ANTUNES JR, 2015).

O método DSR assemelha-se aos métodos tradicionais sob a visão de identificar um problema, propor uma solução, realizar a construção e validação de um artefato. No entanto, para caracterizar a criação do artefato, é fundamental guiar o processo de pesquisa com os seguintes critérios (DRESCH; LACERDA; ANTUNES JR, 2015):

1. Propor a criação de um novo artefato;
2. Este artefato deve atender a uma problemática;
3. Explicitar adequadamente sua utilidade;
4. Dispor de contribuição para o avanço do conhecimento;
5. Assegurar a validade da pesquisa na condução de seu rigor ao método;
6. Realizar pesquisas para o entendimento do problema afim de propor possíveis soluções;
7. Transparecer os resultados de toda a pesquisa aos interessados.

Sobretudo as abordagens na literatura, em sua maioria, descrevem para o método DSR uma composição básica quanto às suas etapas, sendo elas: identificação do problema, conscientização do problema, proposta de solução, desenvolvimento do artefato, avaliação do artefato e conclusão. Diante disto, este estudo utilizou como guia para orientação ao método DSR a proposta de modelo sugerida por Dresch, Lacerda e Antunes Jr (2015), conforme ilustra a Figura 1.

Figura 1 - Etapas da metodologia do trabalho

Fonte: Adaptada Dresch, Lacerda e Antunes Jr (2015, p. 125).

A primeira etapa de identificação do problema para o processo de pesquisa parte da motivação de uma observação da realidade ou da literatura, em conjunto de interesse do pesquisador em desenvolver o conhecimento sobre uma nova área ou o intuito de propor soluções importantes para questões de cunho prático (DRESCH, LACERDA E ANTUNES JR, 2015).

Seguida pela etapa de conscientização do problema, o método propõe a busca, por parte do pesquisador, por informações que assegurem a compreensão do problema através de seus aspectos. Gerando, assim, como principal artefato de saída desta fase, a formalização da caracterização do problema a ser solucionado. Dresch, Lacerda e Antunes Jr (2015, p.127) argumentam que nesta etapa "[...] o pesquisador precisa compreender e formalizar os requisitos necessários para que o artefato seja capaz de solucionar o problema.".

A terceira etapa condiz à revisão sistemática da literatura e ao conjunto do referencial teórico, que propõe melhoria de efetividade ao pesquisador para etapa de proposição do artefato. Dresch, Lacerda e Antunes Jr (2015) explicam que identificar propostas já realizadas traz ao pesquisador uma melhor compreensão para futuras soluções ao considerar observações de pesquisas anteriores, além de apoiar na identificação dos requisitos a serem sistematizados no decorrer da pesquisa.

A quarta etapa corresponde ao desenvolvimento do artefato, caracterizada pela contribuição do pesquisador ao gerar conhecimento para a resolução ou melhorias da problemática em questão, detalhando a construção estrutural do artefato. Na quinta etapa, avaliação do artefato, ocorre a revisão dos requisitos elencados na conscientização do problema em comparação aos objetivos esperados, a partir de métodos de pesquisas que conduzam a interação direta do artefato ao usuário final ao qual se destina o artefato proposto (DRESCH, LACERDA E ANTUNES JR, 2015).

A conclusão caracteriza a última etapa, devendo contextualizar sobre os resultados da pesquisa, observações quanto a limitações e decisões tomadas a fim de servir como base para orientar futuros trabalhos (DRESCH, LACERDA E ANTUNES JR, 2015).Esta pesquisa iniciou com a observação da problemática de interesse do autor, seguida pela etapa de conscientização do problema, através de um instrumento de pesquisa de campo, conforme determinação da metodologia empregada. Com o resultado obtido deste instrumento, o problema foi evidenciado e foram, assim, definidos os objetivos da pesquisa.

Através da análise dos resultados obtidos, elencaram-se critérios para o aporte ao referencial teórico e trabalho correlatos que, apoiaram nos requisitos para a proposição de uma solução satisfatória. A partir disto, obteve-se uma proposta de solução, gerando o artefato de saída deste trabalho, composto por um conjunto de elementos estruturados conforme prevê a metodologia empregada na pesquisa.

A etapa da avaliação do artefato consistiu em verificar a viabilidade quanto à sua utilidade no contexto externo, através da aplicação de instrumento de pesquisa aberto ao público de interesse, em conjunto à contribuição de especialistas através de entrevista. Ainda nesta etapa foi realizada a análise dos dados obtidos, extraindo as informações de acordo com requisitos definidos na pesquisa. E por fim, a etapa de conclusão expõe a sugestão de trabalhos futuros e considerações finais.

3 CONSCIENTIZAÇÃO DO PROBLEMA

Definiu-se um estudo investigativo para a composição da problematização, a fim de atingir o objetivo específico em fazer um levantamento das limitações e expectativas realistas sobre automatização de testes em equipes de desenvolvimento de software. Para tal, identificou-se aspectos através da leitura de trabalhos, fóruns da internet e revistas da área que analisados contribuíram para gerar as questões do instrumento de pesquisa. O mesmo encontra-se no Apêndice B deste trabalho, trazendo todas as perguntas e as opções de respostas, bem como, os direcionamentos para as seções quando necessário.

3.1 PROPOSIÇÃO DA PESQUISA

Para a uma primeira validação do questionário, este foi aplicado a um grupo de 10 profissionais da área de qualidade de software. Percebeu-se então a necessidade de adequação da questão "Sobre os testes automatizados: " quanto ao direcionamento do participante entre as seções a partir da escolha de resposta, conforme evidenciado no decorrer da leitura, para uma melhor coleta de informações de acordo com o objetivo da pesquisa. A partir disto, com o questionário apto, a aplicação foi realizada oficialmente na data de 27/05/2019 a 05/06/2019, obtendo 344 respondentes, desconsiderando as respostas dadas na pré-avaliação do instrumento.

Com o intuito de absorver o máximo de experiência dos participantes em relação à automatização de testes, foi implementada a seguinte orientação antecedendo às perguntas "Responda ao questionário considerando sua última experiência com automatização de testes." e com isso, as demais perguntas estão relacionadas diretamente a este período de experiência do respondente.

As questões "A equipe emprega metodologias ágeis?", "A equipe utiliza algum *framework* ágil ou alguma técnica de desenvolvimento de software?", "Qual seu papel na equipe? " , "A empresa é considerada como:" e "Na percepção da equipe, sobre o grau de relevância dos testes quanto à entrega do produto:", formam a base para compreender melhor o ambiente em que o público da pesquisa está inserido e, então, o correlacionar com os fatores problemáticos da ATS. Já as questões diretamente ligadas à automatização de testes, sendo elas, "Quais as motivações para implementação dos testes automatizados?", "Quais testes foram automatizados?", "Houve dificuldades no processo de implementação de testes automatizados? " e "Indique quais as dificuldades:", amparam quanto à visão de qual tipo de

automatização as equipes mais implementam, com ou sem sucesso, além de escalar as dificuldades mais evidentes neste processo.

Na indicação das dificuldades, o respondente poderia optar por mais de uma, além de elencar outra de cunho livre. Para poder dispor dificuldades coerentes ao público participante e além das percepções do pesquisador, buscou-se na literatura o embasamento para tal, elencando então à publicação de Rodrigues (2018). O autor realizou uma pesquisa com a revisão sistemática da literatura e um *survey* com especialistas de testes, identificando e caracterizado os fatores críticos e práticas de sucesso ligados a ATS, as quais serviram de apoio para formulação deste instrumento de pesquisa.

Todas as questões objetivas relatadas acima eram de preenchimento obrigatório, mas também foi disponibilizada uma questão dissertativa, "Sinta-se à vontade para contribuir com qualquer relato quanto a DIFICULDADES, MOTIVAÇÕES ou PRÁTICAS ADOTADAS sobre o processo ou tentativa de implementação de testes automatizados." que, por sua vez, mantêm o respondente livre de escolha para preenchê-la ou não. Para esta, propôs-se a ideia de incorporar ao instrumento de pesquisa as percepções particulares do usuário quanto ao processo de ATS, sem respostas pré-dispostas na questão.

A análise do presente instrumento encontra-se descrita nas seções posteriores, subdividia de modo a facilitar seu enquadramento conforme os temas diretamente ligados a esta pesquisa. Para tal, estruturou-se em Perfil dos participantes, Metodologias Ágeis e Automatização de testes.

3.2 PERFIL DOS PARTICIPANTES

A estratégia abortada para aplicação da pesquisa se deu através da divulgação da mesma pela rede social LinkedIn, adotando por premissa buscar pessoas que pertenciam à área de desenvolvimento de software e elencando com perfis específicos quanto à profissão.

Em um primeiro momento, em sua maioria, os convites foram enviados para profissionais de qualidade de software, em inglês *Quality Assurance* (QA) que engloba os seguintes perfis: Analista de Automação de Testes de Software, Analista de Qualidade de Software, Analista de Testes de Software, Desenvolvedor de Testes Automatizados, Engenheiro de Qualidade de Software, Líder de Qualidade, Consultor de Qualidade de Software, Testador de Software. Nessa primeira iniciativa, de 344 respostas 281 foram correspondentes a estes perfis.

Em uma segunda abordagem da captação de respondentes, foram envolvidos perfis que contemplassem uma equipe de desenvolvimento de software, na qual houve 63 respondentes caracterizados por 39 Desenvolvedores de Software, 9 Engenheiros de Software, 6 Analistas de Sistemas, 4 Gerentes de Projetos, 2 Scrum Master, 1 Líder de Equipe, 1 Designer, 1 Agile Coach, 1 Analista de Sustentação, 1 Gerente da Área de atendimento e 1 Estagiário de software.

Essa definição de perfil se caracterizou ao ser respondia à questão 4 “Qual seu papel na equipe? ” com a proposta de ter utilizado o papel e não profissão, a partir da necessidade de compreender de fato a visão do participante na equipe de desenvolvimento, pois, muitas vezes, o nome dado à profissão não coincide com o papel que este desenvolve dentro da equipe. Isso é comum dentro organizações de cunho tecnológico que adotam papéis em suas equipes, por utilizarem de alguns padrões de projeto que propõe esta prática.

Diante dessa percepção, manteve-se na questão uma opção em aberto para que o respondente se definisse quanto ao seu papel na equipe, o que gerou inclusive a seguinte resposta “Para mim muitos destes são a mesma coisa. Meu papel é arquitetar, desenvolver, implementar e rodar os testes automatizados”. Assim como essa descrição, outros papéis aparecem como “*Scrum Master*” e “*Agile Coach*/Agilista”.

Quanto ao porte da empresa (Questão 6), 301 responderam trabalhar em empresa de grande porte (Acima de 99 funcionários), 18 empresas de médio porte (De 50 a 99 funcionários), 18 empresas de pequeno porte (De 10 a 49 funcionários) e 7 em Microempresa (Até 9 funcionários). Essa visão ajuda a compreender o nicho por porte das organizações tecnológicas a qual a pesquisa teve maior abrangência.

Ainda, ao realizar a análise quanto à importância dos testes para equipes, das 344 respostas 201 evidenciaram que o produto não é liberado sem que todos os testes tenham sido realizados, o que apenas dentro do contexto de empresa de grande porte representam 89% com 180 respostas. E, apontando os testes como parcialmente relevantes, mas com a obrigatoriedade de testes que garantam a funcionalidade mínima do produto, foram 131 respostas, que demonstra claramente a importância da atividade de teste no ciclo do produto ao observar que apenas 12 respostas referenciaram a entrega sem teste.

3.3 METODOLOGIAS ÁGEIS

De acordo com a proposta de pesquisa deste trabalho, a questão 1 “A equipe emprega metodologias ágeis? ” e a questão 2 “A equipe utiliza algum *framework* ágil ou alguma

técnica de desenvolvimento de software?" revelam dentro do total de respondentes a amostra em relação à adesão da cultura ágil e que *frameworks* ou práticas desta estão sendo utilizadas dentro do processo de desenvolvimento.

Apenas desta amostra de respostas, foi possível notar que a utilização de metodologias ágeis já é equivalente a quase 8 vezes mais aos que não a utilizam. Isso já demonstra uma nítida percepção do mercado de desenvolvimento de software sobre a necessidade de adequação a essa cultura cada vez mais crescente. Traduzida esta observação em números, a pesquisa obteve 304 respondentes utilizando metodologias ágeis contra 40 que não fazem uso. Embora a pesquisa tenha demonstrado que 40 empresas não adotam metodologias ágeis, destas, 28 ainda gerenciam seus processos com algum *framework* ou prática ágil.

Através das respostas obtidas na questão 2, que possibilitou múltiplas escolhas quanto ao uso de algum *framework* ágil ou técnica de desenvolvimento de software, foi possível perceber um contexto híbrido por empresas que mesclam o XP com o Scrum na condução dos seus projetos, caracterizado por 5% do total dos respondentes.

Quanto às técnicas de desenvolvimento utilizadas, a mais citada em conjunto com o *framework* Scrum, foi o BDD (*Behavior Driven Development*) com 29% seguido de 6% que aderem o TDD, como um padrão de desenvolvimento. Para equipes que empregam a automatização de teste, é comum a adaptação do projeto ao BDD por atribuir uma melhor afinidade quanto à linguagem de comunicação com a área de negócios. Em relação à adesão ao uso de algum *framework* para ágil, o Scrum foi o mais citado na condução dos projetos, com 271 respostas positivas quanto à sua adoção.

Ainda conforme observado nas respostas da questão 6, como citado na análise de perfil, foi possível identificar uma flexibilidade quanto à adoção de práticas ou *frameworks* ágeis, onde, 329 respondentes afirmam serem adeptos a pelo menos uma destas citadas, independente do porte organizacional.

3.4 AUTOMATIZAÇÃO DE TESTES

A abordagem chave da conscientização sobre a problematização e o levantamento das dificuldades para a pesquisa se fez na questão 3 "Sobre os testes automatizados:" que apresentou as opções de respostas "Houve a implementação de ATS", "Houve a tentativa de implementação, mas falhou", "Não houve a implementação, mas existiu o interesse" e "Nunca foi utilizada, nem existiu interesse", em conjunto com a questão 9 "Houve

dificuldades no processo de implementação de testes automatizados?", onde o respondente apontou quais foram as dificuldades, podendo o mesmo assinalar mais de uma respostas e ainda preencher uma de cunho livre.

Na questão 3, foram 300 respostas positivas quanto ao uso do processo de implementação de testes automatizados. Dentro desse espaço amostral, 225 pessoas afirmaram ter tido dificuldades como a implementação de testes automatizados, sendo que 30 das respostas demonstram que houve a tentativa de implementação, mas falhou. Ainda 37 respondentes marcaram a alternativa "Não houve implementação de automatização de testes, mas existiu o interesse.", sendo que 17 apontaram a não implementação por a submeterem a previsíveis dificuldades futuras. Com base nisso, é possível compreender que apenas 13% das implementações da automatização de testes não encontram dificuldades no decorrer do processo, e que em torno de 12% acabam por não implementar por conta de alguma dificuldade e 75% implementam, porém com dificuldades.

Com relação ao tipo de automatização realizada, foi evidenciado através da questão 8 "Quais testes foram automatizados" que a automatização em testes de interface é a mais utilizada, com 256 respostas, seguido de 149 para testes unitários e 165 para testes de serviços, nesta pergunta o respondente pode escolher mais de uma opção. Quando cruzadas estas respostas à apenas aqueles que se denotaram em contexto ágil, a adoção da automatização dos testes por estes compreendeu 90% das respostas onde, 219 são voltados para interface, 143 para unitários e 154 para serviços.

Ao elencar os dados correspondentes ao tipo de automação utilizada apenas no ambiente ágil à questão 3, foi possível compreender que o percentual maior de dificuldades está correlacionado à implementação de testes automatizados para UI, que corresponde a 57% dentro do espaço amostral das 300 respostas positivas à adoção de testes automatizados (Questão 3).

De modo a mapear as dificuldades com ATS, a questão 9 "Indique quais as dificuldades.", predispôs em suas respostas algumas alternativas para marcação seguida de uma opção de descrição livre, como evidencia o Apêndice C. As respostas obtidas nesta questão preenchem a Tabela 1, tornando possível visualizar como a maior dificuldade exposta pelos respondentes o "Pouco tempo para desenvolver a automação.". Hayes (2004) apud Rodrigues (2018) já apontava que tempo e recursos na ATS são fatores de suma importância para o sucesso da implementação, pois sua curva de implementação é longa e comumente comparada erroneamente à execução manual quanto ao seu consumo de recursos.

Na mesma linha Rafi et al. (2012) agregam que o processo de amadurecimento da ATS requer tempo para a criação da infraestrutura e dos testes que serão automatizados.

Ao analisar o segundo item mais apontado como dificuldade nesse cenário, tem-se a "Ausência de uma equipe dedicada à automação de testes.". Por conseguinte, a alocação do recurso de um testador manual para automatização em tempo parcial é uma prática comum da área, pois geralmente quem está nesse particionamento de tempo, por vezes, não tem o conhecimento necessário para o desenvolvimento da automatização (RAFI et al., 2012). O que ainda se conecta ao terceiro item das dificuldades mais pautadas nas respostas, a "Ausência de uma equipe treinada a automação de testes.".

Tabela 1 - Dificuldades elencadas na pesquisa

Dificuldades elencadas	Quantidade de vezes citada
Pouco tempo para desenvolver a automação	142
Ausência de uma equipe dedicada a automação de testes	121
Ausência de uma equipe treinada a automação de testes	117
Diversas alterações do produto (difícil manutenção do projeto)	104
Ausência de documentação	93
Processo de testes não definido	74
Pouca/nenhuma utilização de métricas de qualidade (usada para medir o sucesso do resultado esperado)	66
Falta de viabilidade técnica (tecnologia não passível de automatização)	47
Falta de viabilidade econômica (benefícios podem não superar o investimento)	26

Fonte: elaborada pela autora.

No cenário corrente da análise das dificuldades, 202 pessoas responderam que o projeto usa de abordagem ágil em seu desenvolvimento. O que correlaciona o terceiro item do Tabela 1 de dificuldades a "Diversas alterações do produto (difícil manutenção do projeto) ", uma característica intrínseca na adoção do desenvolvimento ágil, pois de acordo com Martin e Martin (2011), a aproximação do cliente para com o desenvolvimento tende a gerar diversas alterações no produto, embora não comumente grandes alterações, os requisitos tendem em passarem por constantes mudanças. Enfatizam ainda que, para não fracassar a esse modelo de frequentes mudanças, é necessário criar um plano no qual se tenha garantia quanto à sua flexibilidade para as adaptações às mudanças do negócio e tecnológicas. Rafi et al. (2012) salientam que as diversas alterações em um produto, sendo por mudança tecnológica ou evoluções, muitas vezes levam a dificuldades de manutenção do projeto de ATS. Andrade (2015) adverte que um processo insuficientemente controlado e mal

gerenciado caracteriza imaturidade, gerando resultados negativos e manutenibilidade desnecessária ao invés de evoluções.

Na mesma linha de análise ao item anterior, citado como características do ambiente ágil, a "Ausência de documentação" é apontada como o quarto maior fator de dificuldade quanto à implementação de ATS. Martin e Martin (2016) correlacionam esse fator a um dos valores da metodologia ágil em que prioriza software em funcionamento mais que documentação abrangente. Porém, explicam os autores que este valor comumente sofre uma errônea interpretação em equipes que iniciam a adoção do ágil, por não produzirem qualquer documentação. E, embora a produção de documentação, por vezes, torna-se massiva quando referenciadas a softwares grandes demais, Martin e Martin (2016, p. 35) alertam que "software sem documentação é um desastre.", pois a ausência de documentação impacta em diversas áreas do projeto de desenvolvimento do produto.

A análise apresentada possibilitou evidenciar algumas das dificuldades a serem exploradas no processo de ATS, conforme ilustradas na Tabela 1, e, também, a percepção quanto à automação de interface como atividade de automatização mais utilizada pelas equipes. Assim, com base nesse levantamento, a Tabela 2 apresenta as dificuldades mais citadas que estão diretamente relacionadas ao contexto ágil e à automação de interface.

Tabela 2 - Dificuldades com automação de UI em ambiente ágil

Dificuldades elencadas	Quantidade de vezes citada
Pouco tempo para desenvolver a automação	108
Ausência de uma equipe dedicada a automação de testes	81
Ausência de uma equipe treinada a automação de testes	81
Diversas alterações do produto (difícil manutenção do projeto)	77
Ausência de documentação	67
Processo de testes não definido	51

Fonte: elaborada pela autora.

Diante deste cenário, a presente pesquisa apresenta uma proposta de modelo de processo para automatização de testes de interface gráfica (MPATS – UI) em ambiente ágil, visando atender as dificuldades mais evidentes nesta análise com a automatização de interface, diretamente relacionado à ambiente ágil e elencadas na Tabela 2.

3.5 CONSIDERAÇÕES À CONSCIENTIZAÇÃO DO PROBLEMA

A referente etapa proporcionou a este trabalho a compreensão quanto à contextualização sobre a identificação do problema, este motivado a partir de uma

observação da realidade e de interesse do autor. Dando ênfase, então, à questão de pesquisa: “Como seria um modelo de processo para automatização de teste de UI em ambiente ágil quanto à sua implementação para que contribua com a qualidade e uma entrega de software contínua?”.

Também contribuiu para a percepção dos requisitos a serem explorados no decorrer desta pesquisa como apoio para formular a resposta de questão do trabalho. Assim, para contemplar a base da construção do artefato, seguindo as etapas do modelo adotado para metodologia empregada, o próximo capítulo aborda a revisão sistemática dos trabalhos correlatos à área de pesquisa seguida da identificação da classe de problemas.

4 REVISÃO SISTEMÁTICA

Neste capítulo são apresentadas publicações encontradas que mais se relacionam ao tema deste trabalho, objetivando absorver destas um conjunto de boas práticas que combinado ao referencial teórico da pesquisa, contribuam para a construção do artefato de proposta da mesma. Para uma revisão sistemática deve-se considerar a ferramenta de busca dos estudos, a avaliação e a qualidade dos mesmos em coerência com o referente assunto de pesquisa (LACERDA et al.,2015).

Diante disto, como principais fontes de busca, utilizou-se de plataformas da web *IEEE Xplore Digital Libray*, *ACM Digital Library* e Unique da Universidade Feevale. O filtro para pesquisa compreendeu os idiomas português e inglês, utilizando as seguintes *strings* de busca: "*Software Test Automation*" e "Automação de teste de software". Por padrão a busca foi submetida considerando as *strings* no título da publicação com o range inicial do tempo de publicação a partir de 2010 até 2019. O resultado desta primeira consulta é apresentado na Tabela 3.

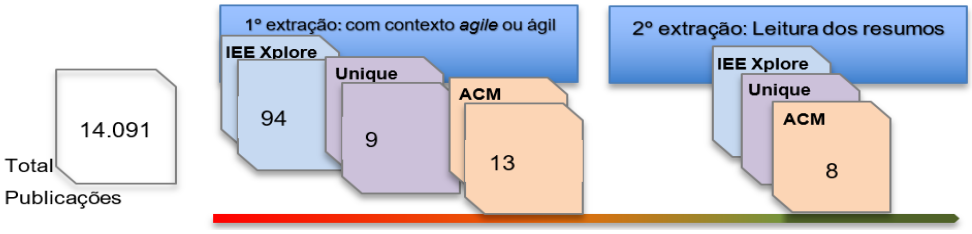
Tabela 3 - Resultados obtidos por biblioteca

Fontes Web	Quantidade retornada	Referência de busca para o título
IEEExplorer	4.798	('*Software Test Automation*'), ('Automacao de teste de software')
Unique Feevale	210	('*Software Test Automation*'), ('Automacao de teste de software')
ACM	9.083	[*Software Test Automation*], [Automacao de teste de software]

Fonte: elaborada pela autora.

A partir dos resultados encontrados na primeira busca, partiu-se para etapa de exclusão das pesquisas que não se correlacionaram à metodologia ágil, utilizando recursos contidas nas próprias bibliotecas virtuais sendo possível filtrar pesquisas cujo em seus resumos apresentassem a palavra ágil no contexto. Para a segunda extração, leu-se cada resumo, a fim de encontrar as publicações de maior afinidade com esta pesquisa. A Figura 2 ilustra o processo de extração e de classificação realizado.

Figura 2 - Resultado por etapa da extração



Fonte: elaborada pela autora.

Com a realização do processo de exclusão, a partir da leitura completa dos trabalhos, foi possível elencar as publicações correlatas ao assunto desta pesquisa, conforme evidenciado no Quadro 1.

Quadro 1 - Lista de publicações aceitas

Título	Autores
Modelo de automação de testes funcionais para desenvolvimento ágil de software	[COLLINS, ELIANE F., 2013]
An Industrial Experience on the Application of Distributed Testing in an Agile Software Development Environment	[COLLINS, et al, 2012]
Strategies for Agile Software Testing Automation: An Industrial Experience	[COLLINS; DIAS-NETO e LUCENA JR. ,2012]
Software Test Automation Practices in Agile Development Environment: An Industry Experience Report	[COLLINS e LUCENA JR.,2012]
Experiência em Automação do Processo de Testes em Ambiente ágil com Scrum e Ferramenta *OpenSource*	[COLLINS, LOBÃO, 2010]

Fonte: elaborada pela autora.

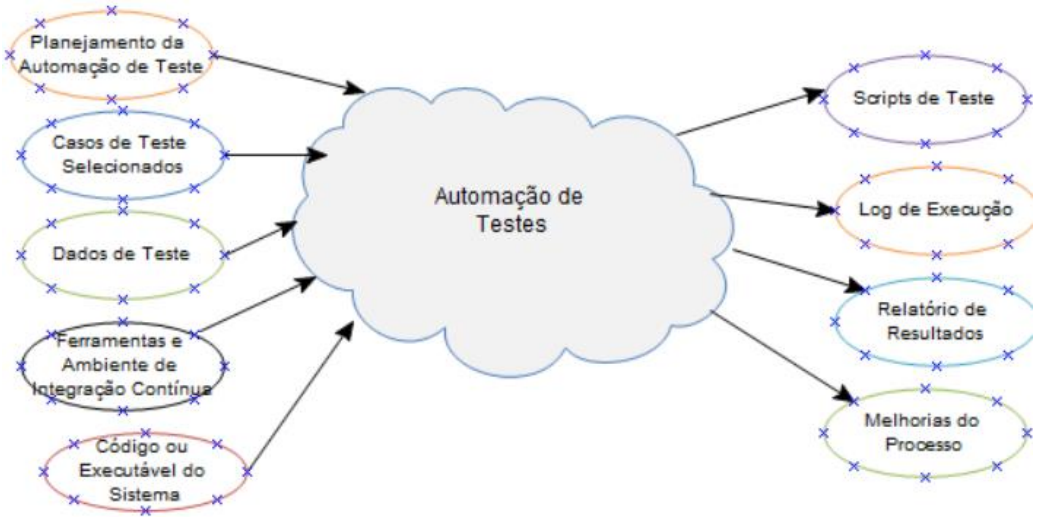
Dada a realização desta revisão, a próxima seção apresenta, a partir das publicações filtradas, artefatos correlatos a este trabalho e suas respectivas observações quanto ao sucesso ou problemas encontrados.

4.1 IDENTIFICAÇÃO DOS ARTEFATOS E CLASSE DE PROBLEMAS

Das publicações elencadas no Quadro 1, a pesquisa realizada por Collins (2013) se relaciona diretamente à proposta desta pesquisa, pois a autora apresenta em sua dissertação um modelo para implementação de ATS, com o estudo de caso envolvendo a observação de 4 projetos de desenvolvimento de softwares que utilizam metodologia ágil com *framework* Scrum, observações estas proveniente das publicações de Collins dos anos de 2010 e 2012, também elencadas no Quadro 1. Por fim, a autora realiza a aplicação do modelo proposto em um projeto de pesquisa no Centro de Tecnologia Eletrônica e da Informação (CETELI), da Universidade Federal do Amazonas (UFAM). Cabe enfatizar que a automação de interface gráfica não está contida neste estudo de caso, pois segundo a autora, o projeto não realizava esta automatização.

O modelo de Collins (2013) prevê 5 artefatos de entradas para gerar a atividade de ATS e 4 artefatos de saída da atividade, como ilustra a Figura 3.

Figura 3 - Entradas e Saídas da Automação de Teste modelo Collins



Fonte: Collins (2013, p. 51).

Como entrada, o planejamento da ATS é consolidado com todo o time no início do projeto durante a cerimônia de *Sprint Planning*, tendo seu registro junto à documentação de teste do projeto, com as informações básicas de ferramentas escolhidas para automação, os responsáveis pelo desenvolvimento e os critérios de início e parada da automação. A orientação proposta quanto à classificação dos casos de testes a serem automatizados está relacionada à quantidade de vezes que estes são executados e sua criticidade quanto à alguma funcionalidade do sistema. Quanto aos dados, esses devem estar disponíveis para que os testes possam ser realizados. Para prover um rápido *feedback*, o modelo espera um ambiente de integração configurado e com acessibilidade disponível para todo o time. Assim como, que se tenham, ao menos, as classes do código do sistema já desenvolvidas.

Os *scripts*, caracterizados no modelo como o artefato de saída mais importante da atividade de ATS, devem estar estruturados em um ambiente de desenvolvimento, a implementação do *log* deve estar diretamente relacionada com os passos dos *scripts* executados pela automação, provendo informações de tempo e respostas do sistema de forma que auxiliem aos desenvolvedores identificarem possíveis falhas com mais facilidade.

Com a finalização da atividade deve haver relatório com os resultados da execução contendo obrigatoriamente os dados: “nome da suíte e dos *scripts* de teste executados, resultado da execução (sucesso ou falha) e um gráfico representando a porcentagem de falha nos resultados.” (Collins, 2013, p. 52).

Embora o modelo de Collins tenha respondido às questões propostas pela autora, a mesma elencou pontos necessários voltados à melhoria do modelo quanto à atualização dos

testes automatizados impactada pelas mudanças de requisitos, melhorias nas tarefas de testes não funcionais e a organização quanto à priorização automática da execução dos *scripts* de teste.

O trabalho de Collins (2013) expõe também um quadro de pontos negativos e positivos relacionados com a ATS e abordados nas reuniões de retrospectivas dos 4 projetos desenvolvidos no INdT que serviram de base para sua pesquisa. É possível notar, ao realizar um comparativo com a questão sobre as dificuldades citadas já no instrumento de pesquisa da problemática deste trabalho, que alguns itens ainda representam equivalência no período atual de mercado. O Quadro 2 proporciona uma melhor visualização destes itens em questão.

Quadro 2 - Classe de problemas quanto ao processo de ATS

<table>
<tr><th colspan="2">Problemáticas Collins(2013)</th><th>Problemáticas apontadas na pesquisa de campo deste trabalho</th></tr>
<tr><td>Projeto 1</td><td>● Muito esforço para automatizar testes de interface.
● Versão para teste instável.
● Dificuldade de comunicação entre equipe de teste remota
● Desenvolvedores entregam versão para teste no último dia.
● Não houve transferência de conhecimento entre o time.
● Falta de proatividade do time de teste.
● Desenvolvedores não executam a suíte automática de teste.
● Retrabalho por causa de falhas de performance e segurança.</td><td rowspan="4">● Pouco tempo para desenvolver a ATS.
● Ausência de uma equipe treinada.
● Diversas alterações do produto.
● Ausência de documentação.
● Processo de teste não definido.</td></tr>
<tr><td>Projeto 2</td><td>● Pouca transferência de conhecimento.
● Muito esforço para automatizar testes de interface.
● Desenvolvedores entregam versão para teste no último dia.
● Faltaram testes de integração o que comprometeu a acurácia de dados gerados pelo sistema.
● Separação entre equipe de teste e equipe de desenvolvimento.
● Falta de sincronização entre teste e desenvolvimento.</td></tr>
<tr><td>Projeto 3</td><td>● Pouca transferência de conhecimento.
● Teste de Sistema feito apenas no final do projeto.
● Separação entre time de teste e de desenvolvimento.
● Correções e melhorias de código postergadas.
● Muita dívida técnica para ser trabalhada.</td></tr>
<tr><td>Projeto 4</td><td>● Muito esforço inicial para configurar o ambiente de automação de teste e IC.
● Retrabalho no código, na tela e nos testes quando o cliente solicitava mudança de requisitos implementados.</td></tr>
</table>

Fonte: Adaptado de Collins (2013, p. 78).

O artigo de Collins e Lobão (2010), embora não tenha gerado contribuição por artefato, reporta a experiência sobre o processo de ATS de algumas das atividades de testes funcionais voltados para software Web, em uma equipe com metodologia ágil com gerenciamento Scrum e com ferramenta de desenvolvimento *OpenSource*. A redução de documentação baseada em requisitos foi um dos fatores citados como colaborativo para o aumento de tempo dedicado para a atividade de desenvolvimento da automação, garantindo o mínimo de qualidade na entrega do produto.

Observam as autoras que, por particularidade do Scrum algumas das funcionalidades tornam-se incompletas ao término de cada *Sprint*, essas atividades geravam posterior retrabalho na atualização dos *scripts* de testes automatizados, denotando como uma imaturidade da funcionalidade para que possa de fato ser desenvolvida na automação. Agregam ainda, a observação quanto ao tempo gasto para correções ou atualizações de código da automação que eram postergadas por impactarem na prioridade de entrega, gerando para a próxima *Sprint* o acúmulo de tempo em atividades de ATS.

A partir da leitura das publicações aqui apresentadas em conjunto à análise do instrumento de pesquisa aplicado por este trabalho na conscientização da problemática, montou-se a percepção dos requisitos necessários para a construção do referencial teórico apresentado no próximo capítulo.

5 REFERENCIAL TEÓRICO

Este capítulo tem por objetivo trazer os conceitos e características da metodologia ágil dentro do desenvolvimento de software, assim como, sistematizar os conceitos básicos para automatização de testes existentes na literatura, apoiando, então, a visibilidade dos atributos que irão compor o artefato fabulado neste trabalho.

5.1 AUTOMATIZAÇÃO DE TESTES DE SOFTWARE

Segundo Myers, Badgett, e Sandler, (2012, p.2, tradução nossa), o teste de software consiste em "[…] um processo, ou uma série de processos, projetado para garantir que o código do computador faça o que foi projetado e, inversamente, que ele não faça nada não intencional.". A atividade de teste de software é aplicada no decorrer do processo de desenvolvimento e manutenção do software e está diretamente ligada ao objetivo de melhoria da qualidade (MARTINS, 2016). E, pode ser realizada de forma manual ou automatizada.

A atividade manual geralmente é utilizada em testes exploratórios ou de usabilidade (HUMBLE; FARLEY, 2014). Porém, realizar testes manuais, demanda um grande esforço humano e propicia a uma maior vulnerabilidade a falhas no próprio processo de testes. Já a atividade de teste quando automatizada consiste na utilização de um software interagindo com a aplicação imitando o processo manual (MOLINARI, 2010).

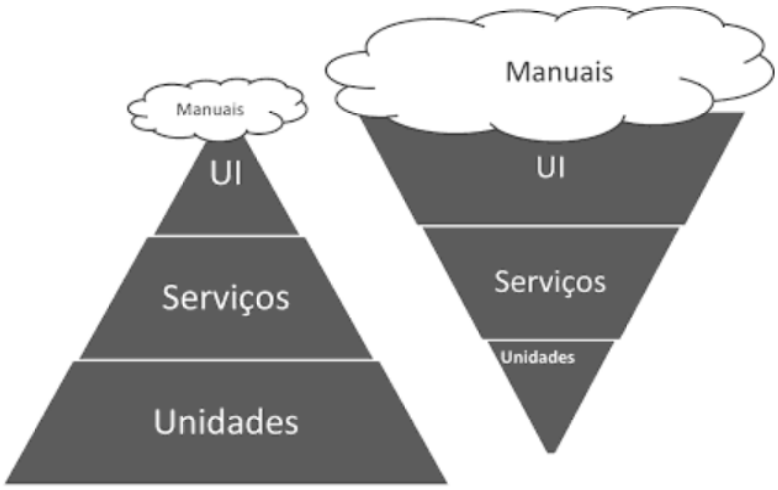
Na literatura é possível encontrar referências tanto ao termo Automação de Testes Software como Automatização de Testes de Software abordando o assunto relacionado a este trabalho. Sommerville (2007, p. 339) define automação de testes como "[…] uso de ferramentas de software para ajudar a reduzir o tempo e o esforço envolvidos nos processos de teste.". A *International Software Testing Qualifications Board* (ISTQB) cita que "A automatização dos testes normalmente requer a automação tanto da execução do teste como da verificação do teste […]". Vincenzi et al. (2018) abordam o tema utilizando o termo "Automatização de teste" e o definem pelo seu objetivo por "[…] permitir que maior quantidade de testes seja realizada em menor tempo, evitando que o testador despenda seus esforços em atividades repetitivas e que poderiam ser realizadas de forma automatizada.".

Segundo Rodrigues (2018), a ATS apresenta benefícios na redução de tempo e erros humanos na execução de testes, assim, gerando um impacto positivo para o aumento da qualidade do software.

A ATS cada vez mais tem estado dentro dos projetos de softwares em consequência da constante exigência do mercado pela rápida entrega do produto. E, para um modelo que

emprega essa entrega contínua, se faz necessária a adoção de uma estratégia de execução dos testes automatizados. Cohn (2011) propôs duas pirâmides de testes (Figura 4) as identificando como uma pirâmide ideal e não ideal em cenários de testes automatizados.

Figura 4 - Pirâmide de testes ideal vs pirâmide de testes não ideal

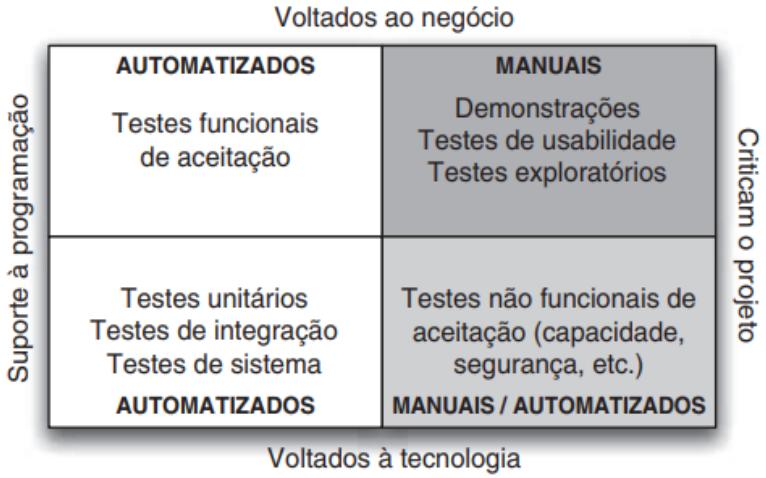


Fonte: Adaptada de Muniz et al. (2019, p.108).

Para Cohn (2011), o esforço maior da automatização dos testes deve ser concentrado nos testes unitários, deixando o mínimo de esforços de testes manuais para específicos cenários que não suportam automatização. Segundo Muniz et al. (2019) essa estratégia é consolidada à abordagem do TDD, propiciando a equipe um *feedback* rápido sobre alguma falha encontrada e ser rapidamente corrigida.

A pirâmide de Cohn atende ao lado esquerdo do quadrante de tipos de teste (Figura 5) proposto por Brian Marick, que categorizou como testes voltados para negócio ou para tecnologia e exibe onde são comumente aplicados testes automatizados (HUMBLE; FARLEY, 2014). Do outro lado do quadrante, os testes voltados ao negócio, dão foco a conhecidos como testes de aceitação ou funcional. Conforme Humble e Farley (2014, p. 85) esses tipos de testes "[…] garantem que os critérios de aceitação de uma história sejam satisfeitos. […]". E Sato (2014, p. 134) observa que são testes "[…] descritos do ponto de vista do negócio, usando uma linguagem que faz sentido para um especialista de domínio.".

Figura 5 - Quadrante de tipo de testes



Fonte: Humble e Farley (2014, p.85).

Para testes voltados à tecnologia, Humble e Farley (2014) explicam que se enquadram testes escritos e mantidos pelo desenvolvimento e como "três tipos de testes que pertencem a essa categoria: testes unitários, testes de componentes e testes de implantação" (HUMBLE; FARLEY, 2014, p. 89). Sato (2014) chama atenção para o quadrante inferior direito quanto à automatização dos testes, o autor afirma:

> Existem formas de automatizar alguns tipos de teste deste quadrante, enquanto outros ainda precisam ser manuais. No entanto, o problema mais comum é que esse quadrante é geralmente ignorado. Problemas de desempenho ou escalabilidade só são encontrados em produção, quando já é tarde demais. (SATO 2014, p. 134).

Embora deva haver esforços para se automatizar o máximo possível de testes, ainda existem testes que não são possíveis de tal feito, tais como testes de usabilidade, testes exploratórios e validação de aparência visual do sistema (HUMBLE; FARLEY, 2014).

A pirâmide proposta por Cohn (2011) enfatiza a priorização da automatização dos testes de acordo com as camadas/níveis de tipo de testes que tragam um *feedback* mais rápido para correção da falha e que empregam menos esforços para automatizar. Diante disto, nos subcapítulos seguintes será abordado cada um dos níveis da pirâmide elencados aos tipos de testes automatizados nestes, com o intuito de interpolar seu entendimento à relação futura do proposto artefato desta pesquisa que, por sua vez, aborda a o processo de ATS ligada à camada mais inferior da pirâmide.

5.2 NÍVEIS DE TESTES AUTOMATIZADOS

Conforme demonstrado na seção anterior a pirâmide de Cohn (2011) apresenta três níveis de testes a serem explorados dentro da automatização. Diante disso, nas seguintes subseções será compreendida cada nível quanto à utilização na ATS.

5.2.1 Testes Unitários

Testes unitários, também conhecidos como testes de unidade, consistem em validar o comportamento de pequenas partes da aplicação de forma isolada, sendo desnecessária a utilização de um ambiente de produção ou o banco de dados da aplicação (HUMBLE; FARLEY, 2014). Segundo Pressman e Maxim (2016, p. 473), este tipo de teste "enfoca a lógica interna de processamento e as estruturas de dados dentro dos limites de um componente.".

O teste unitário está diretamente relacionado à fase de codificação do software. Pressman e Maxim (2016) enfatizam sobre a escrita dos testes unitários serem realizadas antes mesmo do desenvolvimento do código da aplicação. Esse tipo de teste tem como objetivo "mostrar que uma parte específica da aplicação faz o que o programador quer; isso não é, de forma alguma, a mesma coisa que dizer que ela faz o que o usuário precisa que ela faça." (HUMBLE; FARLEY, 2014, p.188).

Técnicas de desenvolvimento como o TDD já sugerem que a implementação do teste de unidade mesmo antes do desenvolvimento do código. O *framework* de metodologias ágeis XP (*eXtreme Programming*) emprega também dentro de suas práticas que desenvolvedores criem primeiramente o teste unitário para só depois realizar o desenvolvimento do código da aplicação (MYERS; BADGETT; SANDLER, 2012).

Frequentemente esses testes desse tipo são confundidos com testes automatizados, porém são tipos de testes candidatos à automatização. Segundo Prikladnicki, Milani e Willi (2014, p. 203), "[…] essa confusão é fruto do nome dado aos primeiros *frameworks* de automação de testes, que originaram a família conhecida como XUnit – JUnit, CPPUnit, NUnit, etc. […]". A automatização dos testes unitários é fortemente defendida em um cenário de entrega contínua. Wildt et al. (2015, p. 130) definem a entrega contínua como "um conjunto de princípios e práticas com o objetivo de compilar, testar e liberar software ao cliente de forma mais rápida e frequente.".

Humble e Farley (2014) destacam como um dos princípios da entrega contínua a automatização de quase tudo dentro do processo de desenvolvimento e apontam ainda para

que testes unitários sejam automatizados e executados em um processo de integração contínua por serem testes de execução rápida. Para os autores uma forma de realizar esse processo de integração é adicionar um conjunto de testes unitários no estágio de *commit* do código, conforme exibido na Figura 6.

Figura 6 - Exemplo de um *pipeline* de implantação

Fonte: Humble e Farley (2014, p. 04).

Na mesma linha Cohn (2011) defende a priorização da automação de testes unitários, formando assim a base da pirâmide ideal (Figura 4), que segundo o autor representam uma cobertura maior cobertura dos testes e uma automação mais rápida de ser desenvolvida. Embora retém destas vantagens, o autor atenta para o fato de que esse tipo de teste pode cobrir apenas uma parcela dos testes da aplicação, havendo a necessidade ainda de atribuir testes de serviços e testes de UI.

5.2.2 Testes de Serviços

O nível de serviço não é restrito apenas a arquiteturas orientadas a serviços ou micro serviços, este prevê a cobertura de testes dos serviços que a aplicação utiliza externamente. Cohn (2011, p. 333) define que realizar o teste no nível de serviço implica em "testar os serviços de um aplicativo separadamente de sua interface de usuário. " Outros autores, como Humble e Farley (2014, p. 97), fazem referência como testes de integração para este nível de serviço, explicando que são "[…] testes que garantem que cada parte independente de sua aplicação funciona corretamente com os serviços dos quais depende.".

Os mesmos chamam a atenção para a maneira de execução deste tipo de teste devendo garantir para que não se tenha de fato a comunicação com um serviço externo. De modo a não correr o risco de transmitir informações de testes para serviços de produção. Para tal, os autores orientam a utilização de réplicas controladas pelo provedor do serviço ou estruturas de testes geradas por parte do código. Para Humble e Farley (2014) projetos que implementam a entrega contínua os testes de serviços são indispensáveis, apontam ainda como estratégia discutir previamente sobre como será realizada a integração externa e os estágios em que o teste deverá ser executado. Para realizar a automatização dos testes neste

nível, é possível utilizar apenas recursos de programação, pois alguns desenvolvedores, por exemplo, já desenvolvem nessa camada com *frameworks* da automação de testes unitários, tais como JUnit ou NUnit (ANICHE, 2017).

Muniz et al. (2019) salientam que os testes de integração são completamente úteis para garantir a validação do fluxo e conexões de componentes internos e externos da aplicação, porém são testes que empregam mais esforços para serem codificados. Mas para Stephens e Rosenberg (2010) esses esforços compensam ao compreender que os testes automatizados neste nível propiciam o encontro de falhas por consequências externas, tais com a quebra de algum serviço de terceiro ou conexão com banco de dados. E enfatizam que "[..] testes de integração são importantes e valem a pena configurá-los e mantê-los funcionando, porque sem eles, você acaba de fazer testes de unidade.". (STEPHENS; ROSENBERG, 2010, p. 253).

Cohn (2011) ressalta que automatizar os testes a nível de serviço isenta, em algumas situações, de se realizar testes massivos na camada de interface gráfica do usuário, e exemplifica usando uma aplicação de calculadora " […] em vez de executar cerca de uma dúzia de casos de teste de multiplicação na interface de usuário da calculadora, executaremos esses testes no nível de serviços. ". (COHN, 2011, p.333).

5.2.3 Testes de UI

Testes de interface gráfica caracterizam-se pela direta interação do usuário com a aplicação, também são referenciados em algumas literaturas como testes *end-to-end* (e2e) (FILHO, 2016). Nesta camada pode haver testes de aceitação, funcionais e usabilidade. Porém, como relatado anteriormente, nem todos os testes são passíveis de serem automatizados. Contudo, os testes funcionais e de aceitação, quando executados pela interface de usuário, são candidatos a automatização. Segundo Cohn (2011) a camada de nível UI é posicionada no topo da pirâmide pelo fato de os testes, mesmo que automatizados, terem uma frequência de execução menor a que os demais níveis.

Na mesma linha Humble e Farley (2014) sugerem que se evitem testes de UI em *commit*, descrevendo duas dificuldades quanto a esse processo. A primeira é a dependência de muitos componentes ou outros níveis de aplicação para o teste, o que impacta em tempo pela necessidade de que tudo esteja de fato pronto para ser testado. A segunda é a escalabilidade deste tipo de teste desenvolvida para uso humano que quando operada computacionalmente torna-se mais lenta e impacta no processo de desenvolvimento.

Para realizar a automatização de teste de UI é possível a utilização de ferramentas que utilizam da técnica de gravação e reprodução (*record and play*) da execução dos passos do usuário gerando *scripts* de testes. *Test complete* é uma ferramenta com esse propósito e funciona através da captura de posicionamento do elemento em tela através de determinada ação do usuário trabalhando com a proposta de poder automatizar tudo que pode ser visualizado na interface. Assim, facilitando a automatização de testes funcionais via interface podendo ser adaptada a diversas plataformas como Web, Windows, iOS, Android e outras (DELAMARO, 2016).

Embora ferramentas como essas facilitem o processo de automação de UI, ao gerarem um *script* de teste, este por sua vez, é complexo, de difícil manutenibilidade e de atualização pela equipe. Humble e Farley (2014) sinalizam sobre a fragilidade quanto à estratégia que esses tipos de ferramenta utilizam:

> [..] muitas das ferramentas de teste de interface do usuário adotam uma estratégia ingênua que os acopla à interface do usuário; o resultado disso é que, quando a interface do usuário muda, mesmo que pouco, os testes quebram. Isso gera muitos falsos positivos: testes que quebram não porque há um problema no comportamento da aplicação, mas porque um campo de formulário teve seu nome alterado. ” (HUMBLE; FARLEY, 2014, p. 88).

Para contornar essa situação, existe a possibilidade de a própria equipe de desenvolvimento criar uma aplicação de testes de interface utilizando a linguagem de desenvolvimento que preferir e com apoio de *frameworks* próprios para criação de testes automatizados. O Selenium, por exemplo, é um dos *frameworks Open Source* mais utilizado na automação de interface, oferecendo suporte com um conjunto de componentes que disponibiliza seu uso em diferentes contextos de aplicações (MARTINS, 2016).

Quando aplicações possuem alto nível de complexidade, inúmeras telas e regras de negócio, a utilização desse tipo de estratégia de automação de interface é mais aconselhável do que a adoção de ferramentas gravação e reprodução. Segundo Martins (2016) ferramentas como Selenium são mais robustas e, além de possibilitar a customização do código da automação que, por consequência, fornece testes mais específicos, muitas vezes proporciona execuções mais eficientes que ferramentas de *record and play*.

Embora se tenha inúmeros suportes para automatizar a interface de usuário, Cohn (2011, p. 332) observa algumas características quanto a esse processo que o disponibiliza no topo de sua pirâmide:

- **Frágil:** pequenas alterações na interface da aplicação ocasionam a invalidação de muitos cenários de testes já automatizados necessitando de manutenção constante e levando à desistência do projeto de ATS por parte da equipe.
- **Caro para escrever:** são testes demorados de serem escritos e quanto utilizado de ferramentas *record and play*, geralmente, são muito mais frágeis.
- **Demorado:** são testes demorados quanto à sua execução e, por vezes, levam mais de uma noite a serem executados.

Mesmo que tenham essas características negativas automatizar teste de interface gera para a equipe e profissionais diretamente ligados aos testes manuais, um ganho de tempo que possibilita ser gasto com outras atividades ou melhorias. Ainda, em reflexão a testes de regressão, quando efetuados diretamente na interface, são massivos e repetitivos, pois costumam ser executados a cada entrega do produto, e, se automatizados, reduz consideravelmente o esforço manual.

Diante do contexto dos níveis apresentados, é possível perceber a necessidade técnicas que auxiliem na execução prática de cada nível dentro do processo de ATS. Deste modo, na próxima seção serão apresentadas abordagens quanto às técnicas de desenvolvimento que contribuam para esta execução.

5.3 TÉCNICAS DE DESENVOLVIMENTO

Testes automatizados fornecem maior confiabilidade ao produto e proporcionam a médio prazo uma forma de menor custo de garantir o funcionamento contínuo do produto através deste processo. Pádua Filho (2010, p. 370) salienta quanto à importância da automação em tempos atuais, afirmando ser indispensável "[…] pois só testes automatizados podem executar de forma confiável o grande volume de casos de testes que é necessário para conseguir uma boa cobertura.".

Contudo, deve haver o alinhamento da equipe quanto à importância que se tem aos testes para com a entrega do produto, já que testes estão diretamente ligados à qualidade. Assim, com base no atual contexto de mercado em prol de entregas cada vez mais rápidas, é comum ver a adoção de técnicas de desenvolvimento focadas em testes, tais como, TDD, ATDD e o BDD (VERSIONONE, 2019).

5.3.1 *Test Driven Development* (TDD)

Considerado um estilo de desenvolvimento guiado por testes, proposto por Kent Beck em 2003 e que tem por objetivo encontrar a correção de falhas de forma antecipada em um processo de desenvolvimento de software (SBROCCO; MACEDO, 2012). Para Beck (2010), a busca por uma nova técnica de desenvolvimento se fez também pelo impulso de se ter código limpo no desenvolvimento usufruindo do teste para tal feito, por consequência, proporcionando qualidade à aplicação.

Para conceituar os padrões de desenvolvimento que englobam o TDD, Beck (2010) inicia sugerindo compreender o que significa o teste, quando testar, qual a lógica a ser testada e a escolha dos dados de testes. Sob a visão do autor, testar é uma forma de realizar a avaliação do que foi desenvolvido e que abrange procedimentos que conduzem à aceitação ou rejeição.

Beck (2010) afirma que projetos de desenvolvimento quando expostos a um alto nível de estresse para rápida entrega, tendem a reduzirem os testes e para ele "Quanto menos você testar, mais erros vai cometer." (Beck, 2010, p. 144). Diante disso, o mesmo propõe substituir o teste manual por testes automáticos, adotando a justificativa de que quanto maior o estresse mais testes poderão ser executados.

Sobre quando testar, é uma das convenções do padrão mais citadas no meio da literatura que aborda TDD e metodologias ágeis, pois para Beck (2010) o teste deve ser escrito antecedente ao código a ser desenvolvido afirmando que "[…] testando primeiro, reduzimos o estresse, o que nos faz mais aptos a testar. " (Beck, 2010, p. 150).

Em um primeiro momento dessa abordagem pode-se imaginar que escrever primeiramente um teste pode demorar mais para o desenvolvimento da aplicação. E sim, conforme George e Willians (2003, apud COHN, 2011, p.180) essa técnica tende a consumir 15% de tempo de desenvolvimento. Porém estudos comprovam que há compensação no uso quanto à redução de defeitos.

> Dois estudos da Microsoft descobriram que o número de erros encontrado diminuiu em 20% e 38% com o uso do TDD (Sanchez, Williams e Maximiliano, 2007, 6). Portanto, sim, o TDD pode demorar mais inicialmente, mas o tempo será devolvido à equipe na forma de menos correção de erros e manutenção. (COHN, 2011, p. 180).

Para iniciar a escrita dos testes antes do desenvolvimento o foco é testes unitários, Beck (2010) sugere criar uma lista para elencar itens que realmente irá implementar com

dados, após uma versão com dados nulos para as operações e, por fim, refletir sobre as refatorações necessárias. Ainda pelo mesmo, é importante não utilizar a mesma constante com diversos significados ao referenciar os dados de testes, para que com isto, seja possível padronizar cada método de teste de acordo com seu propósito.

Diante teste contexto, nota-se que há um ciclo que o TDD emprega em seu modelo. Sbrocco e Macedo (2012, p. 236) o descrevem de forma simplificada como "[…] iterações, que começam pela implementação de um caso de teste, depois pelo código necessário para executar o teste e terminando com o aperfeiçoamento do código criado, objetivando acomodá-lo a mudanças que foram feitas.".

O TDD é ilustrado por muitos autores como uma técnica de desenvolvimento confiável quanto à qualidade embarcada ao produto desenvolvido. Cohn (2011) sugere considerar o TDD não apenas como uma prática no projeto de desenvolvimento, mas também absorver as vantagens testes que ele proporciona para todo um ciclo de desenvolvimento. Beck (2010, p. 105) complementa enfatizando que "[..] testes que são um subproduto natural de TDD são certamente bastante úteis para serem mantidos enquanto o sistema estiver rodando.", mas chama a atenção que a adoção da técnica de TDD não isenta de realizar outros testes como de desempenho, estresse e usabilidade.

5.3.2 *Acceptance Test-Driven Development* (ATDD)

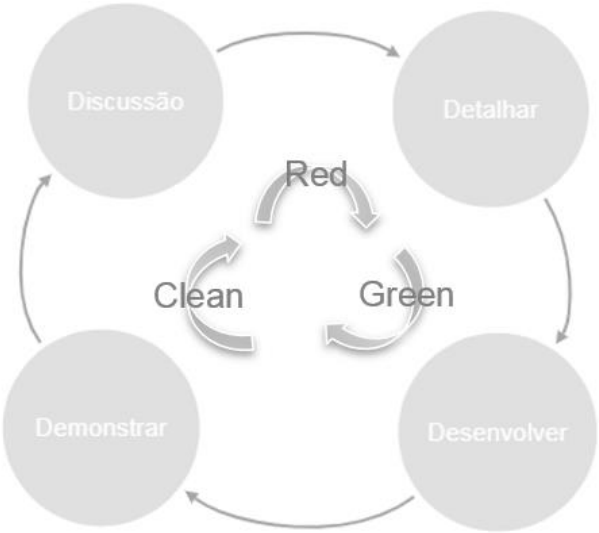
Segundo Cohn (2011), ATDD é análogo à técnica do TDD, porém com o desenvolvimento guiado a testes de aceitação. Cruz (2018) agrega o conceito explicando que o ATDD foca em testes sob a percepção do cliente visando atender as necessidades de negócio.

O ciclo de ATDD, conforme demonstrado na Figura 7, é composto por 4 etapas. A etapa da Discussão é guiada em uma reunião que visa coletar os critérios de aceitação do cliente. Para Cruz (2018), uma maneira de realizar a discussão dos itens de aceitação é pela a abordagem consolidada em uma reunião de planejamento da iteração de negócios com o cliente. Assim, com os critérios de aceitação já alinhados possibilita documentar, na etapa de Detalhamento, os testes de maneira padronizada e formal às necessidades do cliente.

Massari (2018) destaca que a documentação nesta fase deve ser amigável e integrada em *frameworks* que possibilitam a integração com a automação de testes. Desta forma, na etapa de desenvolvimento, a escrita dos testes automatizados segue o fluxo criado na etapa anterior. Por último, na etapa de Demonstração, onde o cliente deverá validar se os requisitos

estão de acordo com os critérios de aceitação no produto previamente identificados na demais etapas.

Figura 7 - Ciclo de desenvolvimento ATDD



Fonte: Adaptado de Massari (2018, p. 312).

Cohn (2011) observa que a adoção da prática do ATDD agrega para as equipes reduzirem o fluxo de trabalho desnecessário dentro de um curto ciclo de desenvolvimento (*Sprint*), explicando que as equipes se beneficiam ao fazer uso inicialmente das condições de satisfação do cliente para configurar o todo o fluxo de desenvolvimento e ainda "libera o dono do produto da necessidade de estar envolvido em todo o ciclo ATDD." (COHN, 2011, p. 337).

Essa visão de envolvimento do cliente para uma definição inicial, que compõe os testes de aceitação, traz um contraponto de mudança cultural no qual Beck (2010) alerta sobre possível resistência das organizações, pois para o mesmo este tipo de mudança de responsabilidade exige, em um primeiro momento, que todos os esforços sejam realizados por muitos da equipe e concentrados a obter os testes escritos.

Muito autores na literatura ágil fazem referência do uso do ATDD de forma combinatória com *frameworks* de gerenciamento de processos de software ágil, tal como Scrum. Assim como Cohn (2011), para Cruz (2018) também há vantagens em se utilizar a técnica com o Scrum por contribuir ao desenvolver as funcionalidades já orientadas ao aceite por testes pré-definidos do cliente. O autor ainda observa que tanto o ATDD e o Scrum trabalham em sincronia quanto à entrega de valor ao cliente, fazendo das práticas ótimas aliadas.

5.3.3 *Behavior Driven Development* (BDD)

Um grande desafio na área de desenvolvimento de sistemas é atender os requisitos de software esperados pelo cliente. Para tal é fundamental que sejam compreendidos claramente estes requisitos e que a comunicação entre os envolvidos seja constante e sem ruídos. Contudo, ainda há dificuldades para se garantir que os requisitos compreendidos sejam de fato pertinentes à implementação do código e dos testes.

Dan North em 2003 identificou uma prática de desenvolvimento orientada ao comportamento denominada de *Behavior Driven Development* (BDD), traduzida para o português como Desenvolvimento Orientado ao Comportamento, prática esta que coloca em foco o comportamento ao invés da estrutura para cada nível de desenvolvimento, pensando mais sobre interações entre pessoas e sistemas (CHELIMSKY et al., 2010).

North (2003, apud YE, 2013, p. 6, tradução nossa) define que o BDD "[…] descreve um ciclo de interações com saídas bem definidas, resultando em uma entrega de software testado e com o funcionamento do que é importante.". Ye (2013) complementa que o BDD além de herdar muitos benefícios e práticas do TDD combinada à ideia de Projeto Orientado a Domínio, em inglês, *Domain Driver Design* (DDD), fornecendo às equipes de desenvolvimento e aos analistas de negócios ferramentas e processos compartilhados para entendimento dos requisitos.

O BDD tem por objetivo auxiliar na comunicação de uma forma mais simples quanto à escrita de um cenário de teste através de uma tríade de palavras formado por Dado, Quando e Então, ligadas diretamente ao comportamento da aplicação. Assim, implementando uma linguagem única (ubíqua) e de fácil compreensão para analistas, desenvolvedores e testadores, que segundo, Ye (2013), contribui para uma redução significativa do mal-entendidos na comunicação e no desperdício de tempo com código e funcionalidades desnecessárias.

O presente capítulo evidenciou as técnicas que auxiliam no desenvolvimento de software e consequentemente na implementação dos níveis de testes relacionados a ATS e apresentados na seção 5.2. Contudo, a presente pesquisa tem por foco o ambiente ágil de desenvolvimento assim, a próxima seção abordará os processos e práticas que norteiam esse ambiente a nível de gerenciamento, partindo inicialmente da contextualizado sobre a metodologia em si.

5.4 METODOLOGIAS ÁGEIS

Buscando uma alternativa para mudanças de processos e práticas burocráticas que tradicionalmente eram utilizadas no desenvolvimento de software, um grupo formado por líderes da comunidade de software, no ano de 2001, reuniu-se e definiu uma mudança de adaptação que visava o mínimo necessário para realizar um trabalho, buscando aceitar e tratar transições mais rapidamente e objetiva, surgindo então o Manifesto Ágil (WILDT et al., 2015).

O Manifesto Ágil contém a descrição dos valores e princípios que dão fundamentação ao desenvolvimento ágil de software, quanto aos valores são (BECK et al., 2015):

- Indivíduos e interações mais que processos e ferramentas;
- Software em funcionamento mais que documentação abrangente;
- Colaboração com o cliente mais que negociação de contratos;
- Responder a mudanças mais que seguir um plano.

A adoção de metodologias ágeis tem sido a cada ano maior a nível mundial, segundo a 13ª pesquisa referente ao ano de 2018 e divulgada pela VersionOne (2019) 97% dos participantes relataram que em suas organizações é realizada a prática de métodos de desenvolvimento ágil, percentual este que em relação à divulgação do ano de 2018 pela mesma, apresentou um aumento de 18%.

Cohn (2011, p. 25) justifica a crescente adoção do ágil pelo resultado que este agrega ao processo de desenvolvimento gerando "[…] ganhos significativos em produtividade com reduções equivalentes no custo. Conseguem introduzir produtos no mercado com muito mais rapidez e com um nível maior de satisfação do cliente.". O autor complementa chamando a atenção para a necessidade de não apenas ser ágil, mas também adotar práticas no processo que promovam uma visão mais organizada do projeto de desenvolvimento como todo, sugerindo como exemplo, o uso de *frameworks* como o Scrum.

Assim como Cohn (2011), os autores Martin e Martin (2010, n.p.) defendem que para alcançar a capacidade de desenvolver softwares de maneira ágil

> […] precisamos usar práticas que forneçam a disciplina e o retorno necessários. Precisamos empregar princípios de projeto que mantenham nosso software flexível e passível de manutenção, e precisamos conhecer os padrões de projeto que têm se mostrado capazes de equilibrar esses princípios para problemas específicos.

Além do Scrum, conforme citado por Cohn (2011), a pesquisa da VersionOne (2019) trouxe também o *eXtreme Programming* (XP) como outra referência de maior uso sobre essa abordagem de projetos e ainda a assídua adesão da cultura DevOps. Para melhor compreender essa contextualização da aplicação de *frameworks* na adoção das metodologias ágeis, nas seções subsequentes serão descritas as principais abordagens que estão sendo utilizadas para implementar essa essência do ágil nos projetos de desenvolvimento de software.

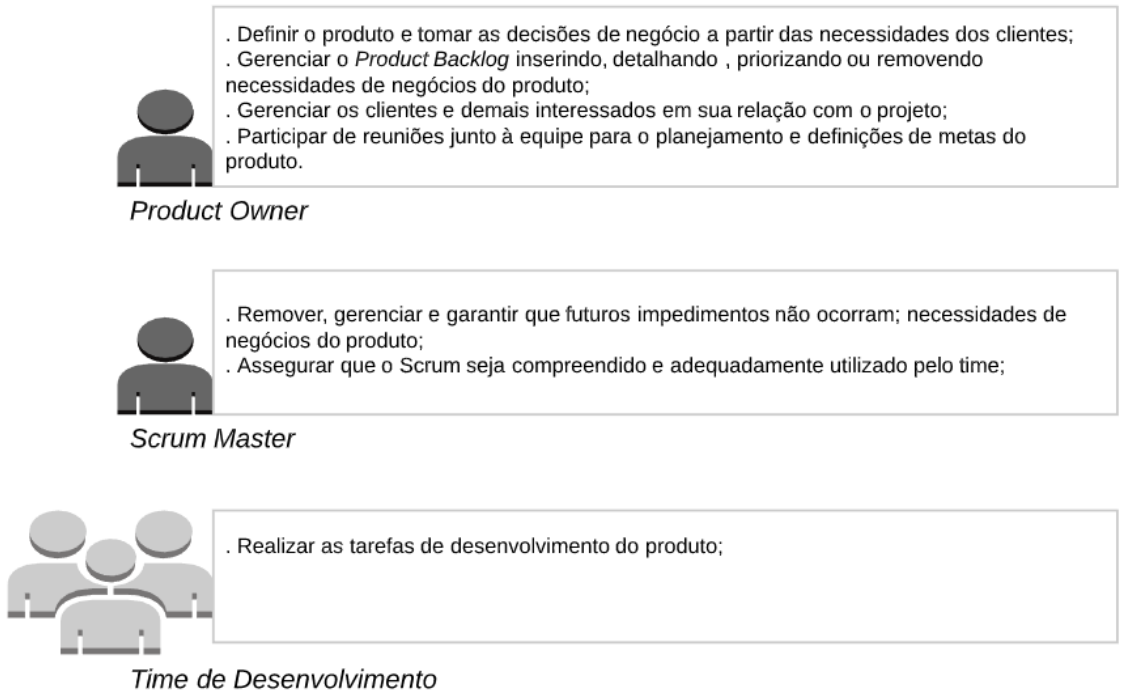
5.4.1 Scrum

Em 1993, em uma empresa de software nos EUA chamada *Easel Corporation*, foi criado o primeiro time Scrum através de Jeff Sutherland que em 1995 o apresentou para Ken Schwaber, CEO na empresa *Advanced Development Methods*, o qual também trabalhava com as mesmas ideias de Sutherland. Ambos em 2001 participaram da criação do Manifesto Ágil e descrevem o *framework* Scrum (SABBAGH, 2013).

Segundo Cruz (2018, p. 49) o Scrum é "[…] um *framework* para gerenciamento de projetos ágeis que, apesar de muito comum na área de desenvolvimento de software, pode ser utilizado também para o planejamento, gerenciamento e desenvolvimento de qualquer produto.". Sabbagh (2013) complementa que o *framework* Scrum é fundamentado empiricamente com abordagem iterativa e incrementa, o que gera um entrega de valor com frequência e reduz riscos para o projeto.

O Scrum, além de abordar valores e princípios do ágil, trabalha com pilares definido por papéis, artefatos e eventos, servindo de guia ou suporte para o projeto, porém não prescreve como o produto deve ser desenvolvido. Quanto aos papéis atribuídos pelo Scrum estes são três: *Product Owner*, Time de Desenvolvimento e *Scrum Master*. Todos são igualmente responsabilizados pelo resultado do produto. Na Figura 8 é possível ver como o Scrum define esses papéis e algumas atividades atribuídas para cada um.

Figura 8 - Scrum Papéis e atribuições



Fonte: Adaptado do texto de Sabbagh (2013, p. 120 a 179).

Sabbagh (2013) explica que para o Scrum não há como se ter um conjunto de práticas específicas que pudessem atender a todos os contextos, mas sim o oposto " […] as pessoas gerando o produto, a partir dos papéis, eventos, artefatos e regras do Scrum, avaliam, adquirem e desenvolvem um conjunto de práticas que melhor lhe servirão, o que é constantemente reavaliado. " (SABBAGH, 2013, p.70).

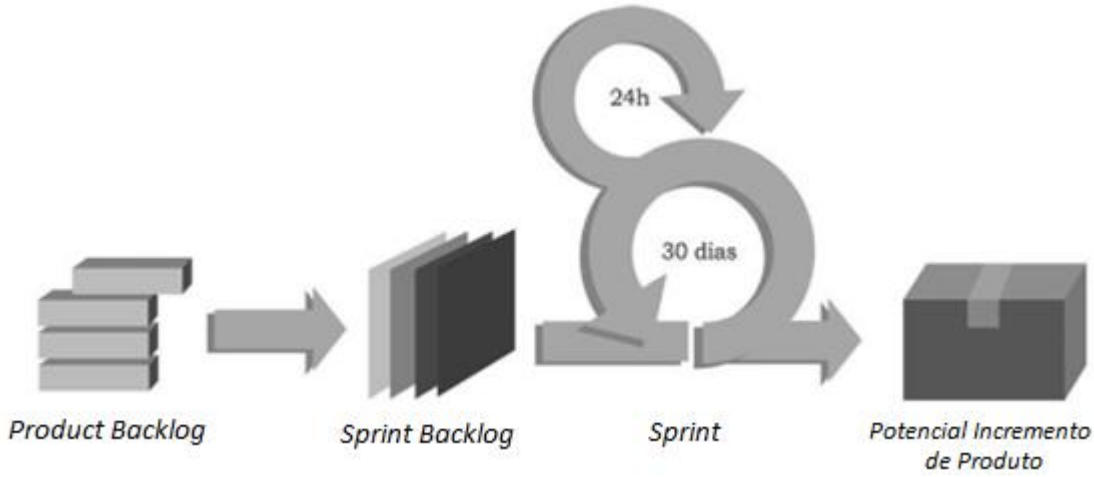
O Scrum define a utilização de quatro artefatos (SABBAGH, 2013);

- ***Product Backlog***: engloba as necessidades ou objetivos do negócio quanto ao produto (novas solicitações, melhorias ou correções);
- ***Sprint Backlog***: consiste em uma lista de itens selecionados do *Product Backlog* que serão trabalhadas no decorrer da Sprint;
- **Definição de Pronto**: varia de equipe para equipe e apoia-se em um acordo do *Product Owner* com o time de desenvolvimento quanto à finalização do que foi desenvolvido. Essa definição deve estar visível e compreendida por todos do Time do Scrum;
- **Incremento no Produto:** é o resultado dos itens trabalho a partir do Sprint Backlog, ou seja, tudo que foi produzido na Sprint.

Segundo Massari (2018) o Scrum realiza cerimônias em seu ciclo (Figura 9), nos quais estipula tempos limites de duração. Para um *Sprint* o tempo pode variar de duas semanas a no máximo 30 dias. Para uma Reunião de Planejamento de *Sprint* o *framework* estipula o máximo de 8 horas, sendo estas divididas em carga de 4 horas. Para as *Daily Scrum* (reuniões diárias), momento em que o time de desenvolvimento compartilha o conhecimento, empecilhos e status das tarefas executadas, devem tem tempo máximo de 15 minutos. Para limitar esse tempo, Massari (2018) explica que os assuntos devem limitar-se a responder três perguntas: "O que fiz ontem?", "O que vou fazer hoje?" e "Quais são os impedimentos?". O autor ressalta ainda que cabe ao Scrum *Master* guiar a reunião e tratar de assunto adversos em um outro momento.

Outra cerimônia é a Revisão de *Sprint* (*Sprint Review*), realizada sempre ao fim de cada ciclo de *Sprint* para apresentação do incremento do produto trabalho ao *Product Owner*, o tempo gasto para esta cerimônia limita-se em 4 horas (MASSARI, 2018). Por fim, a Retrospectiva de *Sprint* exercida também ao final de cada *Sprint* com o intuito de refletir sobre o andamento da mesma. Massari (2018) ressalta que é guiada por três questões: "O que deu certo durante a *Sprint*?", "O que precisa ser melhorado para as próximas *Sprints*" e "Quais ações de melhorias serão incorporadas na próxima *Sprint?*" e o tempo limita-se em um máximo de 3 horas.

Figura 9 - Ciclo de Vida no Scrum



Fonte: Massari (2018, p. 6).

Embora o Scrum seja um *framework* com foco em ágil, é comum que equipes dentro de metodologias tradicionais adotem seus artefatos ou suas cerimônias para melhorias em seus processos. De acordo com Massari (2018), existem modelos híbridos de processos, que

consistem na combinação de métodos ágeis aos tradicionais, fazendo uso, por exemplo, de cerimônias do Scrum. Mas para Wildt et al. (2015) não basta apenas adotar um modelo como Scrum no dia a dia para desenhar boas histórias dos requisitos, pois o mesmo afirma que "[…] o que faz essas histórias sair do outro lado como software rodando sem bugs em produção é a capacidade e dedicação da equipe[..]".

5.4.2 eXtreme Programming (XP)

Alguns autores definem o XP como uma metodologia, como Wildt et al (2015, p.16) ao afirmar que "[…] *eXtreme Programming* é uma metodologia ágil de desenvolvimento de software voltada para times de pequeno a médio porte, no qual os requisitos são vagos e mudam frequentemente.". Já para Martin e Martin (2011, p. 47) consiste em um *framework* por prover "[…] um conjunto de práticas simples e concretas que se combinam em um processo de desenvolvimento ágil.". Na mesma linha de definição de Wildt et al. (2015), Sbrocco e Macedo (2012) ressaltam ainda que o XP é facilmente adaptável em pequenas e médias equipes nas quais o projeto é guiado por requisitos leves e de rápida modificação. Apontam também, como características marcantes do XP, o *feedback* constante, abordagem incremental e o encorajamento à comunicação entre todos os envolvidos.

eXtreme Programming, também conhecido em português como Programação Extrema, foi proposto por Kent Beck, Ward Cunningham e Ron Jeffries, sendo implementado pela primeira vez em uma equipe no ano de 1996 (PÁDUA FILHO, 2010). Totalmente orientado às pessoas e com ênfase na qualidade de codificação, se opondo ao pensamento de gerência no qual pessoas sejam permutáveis entre os processos de desenvolvimento e valoriza fortemente a automatização de testes no processo (WILDT et al., 2015).

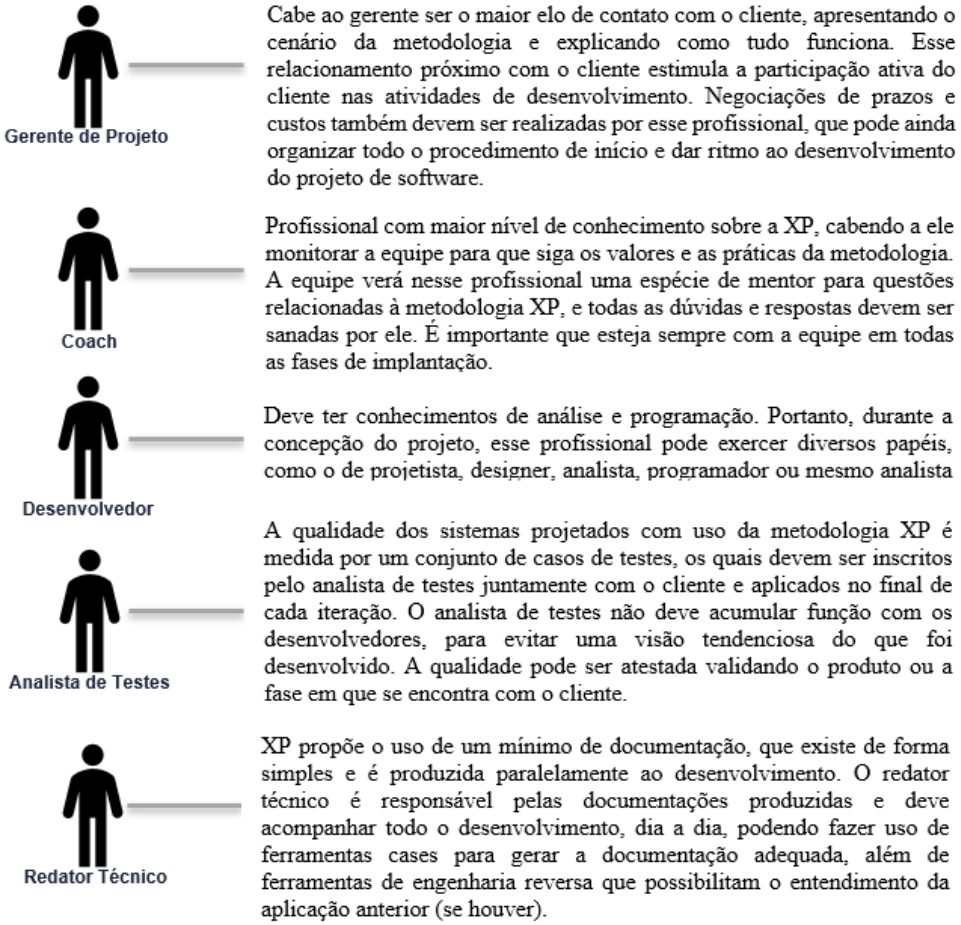
O *framework* adota cinco valores que são essenciais para que seus papéis e práticas estejam coesos à essência da metodologia ágil, estes são descritos e caracterizados por Cruz (2018) a seguir:

- **Comunicação**: garantir que partes envolvidas com o projeto possam se compreender de forma colaborativa, objetiva e mais produtiva possível;
- ***Feedback***: deve ser constante e garantido através de estratégias iterativas e incrementais, permitindo que erros descobertos o mais cedo possível e reparados mais rapidamente.
- **Coragem**: promover o conceito e o princípio de *whole team* (equipe unida), permitindo que opiniões e resultados de trabalhos sejam expostos sem medo;

- **Simplicidade**: focar na entrega do que realmente gera valor ao cliente;
- **Respeito**: garantir um bom relacionamento dos integrantes da equipe, sabendo ouvir o ponto de vista um do outro, dando atenção às necessidades e expectativas aos envolvidos no projeto.

Para o XP cada papel tem seu objetivo, porém se complementam. Sbrocco e Macedo (2012) apresentam um resumo sobre a definição de cada papel na formação de equipe conforme mostra a Figura 10.

Figura 10 - XP Papéis e atribuições



Fonte: Adaptado de Sbrocco e Macedo (2012, p. 151 a 152).

Através de um conjunto de boas práticas alinhado com os valores anteriormente explicados, o XP objetiva garantir um ciclo de desenvolvimento altamente independente. De acordo com Sbrocco e Macedo (2012) essas práticas são:

- **Padrão de desenvolvimento**: adotar um padrão para que facilite a compreensão mais rapidamente do código fonte;

- ***Design* simples:** propor um *design* da aplicação de maneira iterativa com uma proposta de sempre buscar a solução mais simples e que atenda as reais necessidades do projeto, evitando programação sem necessidade e dando foco ao que traga valor para o cliente;
- **Cliente sempre disponível ou presente:** ter o acompanhamento do cliente no desenvolvimento de forma que participe de todo o processo para um entrega final que venha a atender suas necessidades de maneira plena;
- **Jogo de planejamento:** organizar as funcionalidades (histórias) a serem desenvolvidas categorizando-as por ordem de desenvolvimento conforme sua prioridade. Com essa organização, possibilita estimar o custo de cada funcionalidade e melhor planejar sua execução, bem como as possíveis iterações no ciclo de desenvolvimento;
- ***Stand up meeting*:** realizar uma rápida reunião, na qual os participantes se mantenham em pé objetivando ser clara e concisa na troca de experiência e ou dificuldades encontradas quanto à implementação de alguma funcionalidade. Provém uma duração máxima de 20 minutos;
- **Programação em pares:** todo código implementado deve ser realizado em dupla, de forma que enquanto um desenvolvedor escreve o código outro de forma simultânea possa ir revisando. Promovendo constante troca de experiência e facilidade na resolução de problemas;
- ***Refactoring*:** trabalhar na melhoria do código, objetivando uma melhor legibilidade e até mesmo futuras falhas;
- **Desenvolvimento guiado por testes:** teste automatizado intrínseco no processo de desenvolvimento para que seja possível verificar de forma instantânea se cada implementação atende ao requisito do cliente;
- **Código coletivo:** disponibilizar o acesso de todo o código para todos os desenvolvedores junto com a liberdade de edição, assim provendo a responsabilidade coletiva quanto à implementação;
- **Metáfora:** criar analogias com os processos utilizados evitando termos técnicos com o cliente, contribuindo também para melhoria na comunicação e consequentemente gerando *feedback* mais rápido;
- **Ritmo sustentável:** não dispor de horas extras de trabalho, a fim de evitar a dispersão de atenção do desenvolvedor em função do cansaço, podendo ocasionar erros de implementação;

- **Integração contínua:** integrar o sistema de maneira com que se tenha um software devidamente atualizado, realizando sempre testes a cada integração ao código e posteriormente dispondo para que todos do desenvolvimento tenham a modificação;
- ***Releases* curtos:** dispor imediatamente pequenas partes do software ao cliente com a implementação da funcionalidade conforme a priorização dos requisitos realizadas pelo mesmo. Assim, justificando o investimento até o momento e dispor da aplicação possibilitando uma avaliação mais rápida pela parte do cliente.

O XP em relação ao Scrum prescreve um conjunto de práticas mais robusto, isso faz com que exista uma utilização híbrida no processo de desenvolvimento, ou seja, muitas equipes absorvem também práticas do XP ao implementar o Scrum, como por exemplo, o desenvolvimento guiado a testes (SABBAGH, 2013). A proposta principal do XP é o desenvolvimento do software com alta qualidade, por isso há a valorização da automatização dos testes propondo que estes sejam criação antes, durante e depois da codificação do produto segundo (WILDT et al, 2015). A automatização dos testes deve ser uma prática planejada desde o início da implementação do XP, mas para os autores Wildt et al (2015) deve ser sempre considerada para qualquer padrão de projeto, pois afirmam que "Testar software manualmente, no século 21, é antiético, como um cirurgião operar sem lavar as mãos. Aprenda os conceitos, a essência da automação de teste. " (WILDT et al. 2015, p. 2).

5.4.3 DevOps

A adoção de práticas DevOps tem se mostrado cada vez maior como apontado pela pesquisa da VersionOne (2019) onde cerca de 73% dos entrevistados afirmaram utilizar iniciativas da prática. Por Sato (2014, n.p.), "DevOps é um movimento cultural e profissional […]", complementa ser focado "[…] em automação, colaboração, compartilhamento de ferramentas e de conhecimento […]". A prática DevOps incentiva que as diversas áreas de desenvolvimento de software sejam colaborativas, essa ideia é transportada no significado da palavra DevOps que compreende a união de "desenvolvimento" e "operação" (VERONA; DUFFY; SWARTOUT, 2016).

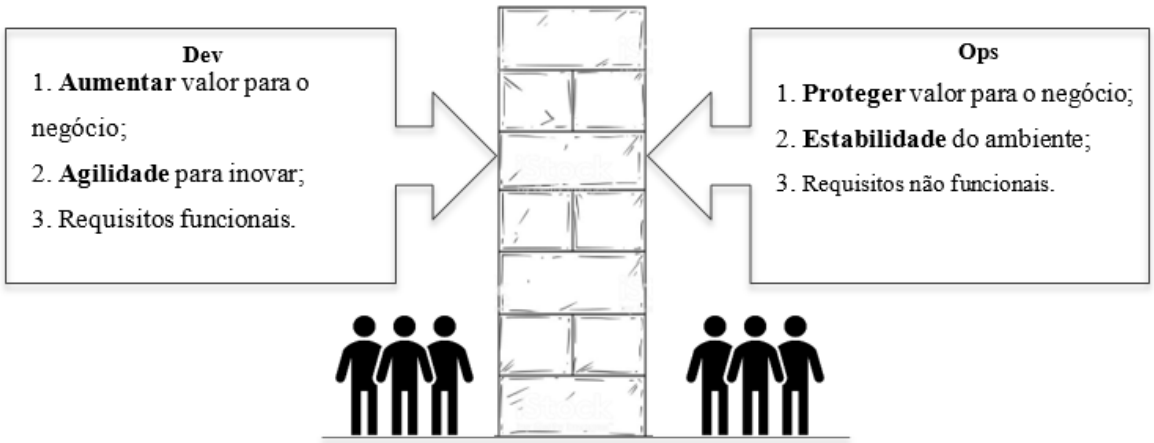
Desenvolvedor de software e consultor nas áreas da TI, Patrick Debois trouxe à tona toda essa ideia do DevOps pela sua frustração da divisão existente entre as áreas de desenvolvimento e operações. Debois et al. (2016, p. 9, tradução nossa) explicam que o

conflito dessas áreas se configura pela visão sobre as responsabilidades de cada uma, na qual "[…] o Desenvolvimento assumirá a responsabilidade de responder às mudanças no mercado, implementando recursos e mudanças na produção o mais rápido possível.". Já para operações a responsabilidade de "[…] fornecer aos clientes o serviço de TI estável, confiável e seguro, dificultando ou mesmo impossibilitando a introdução de mudanças na produção que possam comprometer a produção".

Isso se desenha em um espiral descendente ou uma barreira (Figura 11) gerando, tanto dentro da organização quanto fora, um bloqueio na execução dos resultados desejados do negócio comprometendo a entrega do produto e ou de novos recursos, impactando na qualidade com frequentes interrupções por falhas do software e, considerado o mais agravante, o aumento da dívida técnica (MUNIZ, et al., 2019).

De acordo com Debois et al. (2016), o termo "dívida técnica" é um análogo ao termo de "dívida financeira" por submeter as decisões tomadas ao aumento gradativo de problemas cada vez mais difíceis de serem resolvidos e reduzindo de maneira contínua a disponibilidade de opções solutivas ao problema. Também inferem os autores, que embora essas decisões sejam executadas com prudência, ainda assim, acarretam juros futuramente sob os problemas.

Figura 11 - Barreira entre objetivos e foco principal das equipes Dev e Ops



Fonte: Adaptado Muniz et al. (2019, p. 40).

Verona, Duffy e Swartout (2016) explicam que o movimento DevOps tem suas raízes no Manifesto Ágil e consolida-se ao primeiro valor do manifesto de "Indivíduos e interações mais que processos e ferramentas". Na mesma linha Muniz et al. (2019) agregam que esse movimento caracteriza uma continuação à jornada da cultura ágil, concentrando esforços para entregar mais valor ao cliente, focando na fácil adaptação e aprendizagem contínua. Citam ainda sobre ser um dos maiores benefícios à adoção dessa prática no contexto ágil

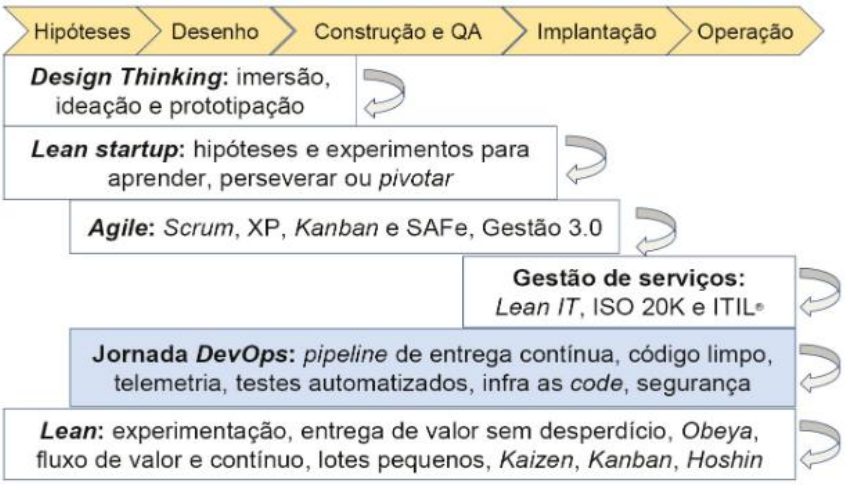
“[…] a mobilização de todas as equipes que participam do fluxo de valor em uma abordagem ponta a ponta, desde o levantamento de requisitos até a entrega do software no ambiente de produção. ” (MUNIZ et al, 2019, p.47).

DevOps tem seu embasamento quanto às práticas e princípios a partir da filosofia Lean, visando a redução de desperdícios e a entrega de valor da qual o cliente está propenso a pagar, sendo alguns dos conceitos adotados:

> **Fluxo de valor:** é o processo que concretiza uma necessidade de negócio em um produto ou serviço para entrega de valor ao cliente.
> **Mapeamento do fluxo de valor:** visa entender como o processo funciona com foco na entrega de valor ao cliente e identifica gargalos ou desperdícios.
> **Gemba:** é o local onde as coisas acontecem. Todos deveriam ir ao gemba com frequência para conhecer o “chão de fábrica” e evitar suposições sem dados e fatos.
> **Obeya**: também conhecida nas organizações como “sala de guerra”, o objetivo é facilitar a gestão visual e a coordenação para solução de problemas sem os entraves das estruturas clássicas das organizações.
> **Kankan:** quadro que permite visualizar onde o trabalho está fluindo bem e onde está na fila ou interrompido. Veja um exemplo desse quadro no decorrer deste capítulo, quando falarmos sobre WIP.
> **Corda de Andon**: dispositivo que existe nas fábricas da Toyota para interromper a linha de produção quando é encontrado algum defeito nos produtos. O objetivo da aglomeração (swarming) é conter os problemas antes que estes tenham a chance de se espalhar. As equipes até a alta administração.
> **Desperdício:** toda atividade que consome recursos sem adicionar valor na visão do cliente. Muniz et al. (2015, p. 46)

Diante desta estrutura conceitual do DevOps é comumente questionável como enquadrá-lo atualmente nos movimentos e *frameworks* já existentes. Para tal questão, Muniz et al. (2019) demonstram resumidamente na Figura 12, como é realizada a ligação do DevOps para esse contexto atual. Propondo a visualização no enquadramento de cada *framework* ou movimento no *pipeline* (linha de instruções) do DevOps.

Figura 12 - Enquadramento DevOps no *pipeline* com *frameworks* ágeis



Fonte: Muniz et al. (2019, p. 40).

Debois et al. (2016) relatam que na atualidade da indústria de software, a qual tem sido movida pela competitividade de uma entrega de valor com tempos cada vez mais curtos, as empresas que não conseguirem uma adaptação a esse modelo de mercado estarão fadadas a perder para competidores mais ágeis. Segundo os mesmos, o DevOps traz em sua cultura defender uso de práticas para garantir a entrega contínua desse produto com alto valor de negócio, a partir do aceleramento do fluxo para a implantação com qualidade do produto em produção, rápido *feedback* e experimentação contínua. Para tal, Muniz et al. (2019) elencam os três fundamentais requisitos do DevOps, sendo eles: a integração contínua (*continuous integrations*), a entrega contínua (*continuou delivery*) e implementação contínua (*continuous deployment*).

A integração contínua, já conceituada na subseção 5.4.1, é fortemente defendida dentro do DevOps em conjunto com a execução de testes automatizados a cada compilação do código. Essa prática é adotada em algumas empresas, inclusive, para implementações do código em produção, pois segundo Verona, Duffy e Swartout (2016) com a integração contínua o código em conjunto à automatização, estará sendo testado com frequência, o que, aumenta precocemente a detecção de falhas antes mesmo de ser integrado a qualquer ambiente. Para Debois et al. (2016, p. 190, tradução nossa), há ainda necessidade de atender alguns requisitos da integração contínua, tais como:

> Um conjunto abrangente e confiável de testes automatizados que validam que estamos em um estado implantável.
> Uma cultura que "interrompe toda a linha de produção" quando nossos testes de validação falham.
> Desenvolvedores trabalhando em pequenos lotes no tronco de desenvolvimento, em vez de ramos de recursos de vida longa.

Quanto à entrega contínua, Muniz et al. (2019, p. 53) a caracterizam como "[…] uma evolução natural quando existe o interesse de expandir os benefícios da automação dos testes e *feedback* [...]". Observam os mesmos, que a base da entrega contínua é a integração contínua, permanecendo para as equipes a ideia do rápido *feedback* em caso de falhas.

Já para a implementação contínua consiste, para o DevOps, em atingir uma etapa de *deploy* automático em produção, submetido anteriormente por testes automatizados com sucesso, atendendo às validações já previstas no escopo de desenvolvimento (MUNIZ et al, 2019). Prática esta, comumente utilizada para sistemas Web, com enfoque de disponibilizar pequenas alterações em produção com uma demanda frequente. As práticas de implementação contínua são de suma importância no universo digital em âmbito de empresas

inovadoras e direcionadas ao *time-to-market* (tempo para comercializar), que realizam a disponibilidade de versões automaticamente em produção Muniz et al. (2019).

5.5 CONSIDERAÇÕES AO REFERENCIAL TEÓRICO

Este capítulo finaliza o conteúdo proposto em seus objetivos específicos quanto ao referencial teórico deste trabalho, atendo também ao rigor do modelo de pesquisa empregado. Ademais, enriqueceu os conhecimentos para proposição do artefato com a busca do entendimento das práticas e conceitos que a literatura oferece ao respectivo assunto de pesquisa.

Assim, seguindo o planejamento estrutural deste trabalho, o capítulo subsequente apresentará o desenvolvimento do artefato, onde, primeiramente demonstra as premissas e limitações para o emprego do modelo, após sua descrição sob uma visão geral, seguida do detalhamento do mesmo quanto a cada etapa empregada no processo.

6 DESENVOLVIMENTO DO ARTEFATO

O presente capítulo aborda a proposta do modelo de processo para automatização de teste UI (MPATS – UI) voltado a equipes de desenvolvimento de software com metodologia ágil como solução à questão de pesquisa deste trabalho. O modelo é totalmente adaptado ao ciclo de gerenciamento Scrum e considera os princípios do Manifesto Ágil em seu desenvolvimento.

6.1 PREMISSAS E LIMITAÇÕES DO ARTEFATO PROPOSTO

Este modelo destina-se a equipes aderentes a metodologias ágeis e que utilizem para gerenciamento de seus processos o *framework* Scrum. O modelo aborda a condução das etapas para equipes que desejam implantar o processo de ATS para interface gráfica ou aderir a melhorias no processo corrente. Para tal, são válidas as seguintes premissas quanto à utilização do proposto modelo:

- ter o Scrum como *framework* padrão na gestão dos processos;
- a equipe deve conhecer e participar de todas as cerimônias do Scrum;
- deve existir na equipe alguém com experiência na área de teste de software e com habilidades para desenvolvimento de software;
- deve haver a conscientização por parte de todos os envolvidos quanto ao interesse de atribuir a ATS UI no projeto;
- ter o projeto de automatização de testes de interface em uma linguagem de desenvolvimento de domínio do time.

As limitações impostas a este modelo estão diretamente relacionadas a suas premissas conforme destacadas acima para ao seu uso em equipes de desenvolvimento de software que têm por objetivo desenvolver o projeto de automatização de testes funcionais voltado à interface gráfica e em uma linguagem de desenvolvimento aberta para todo o time.

Com relação a linguagem de desenvolvimento utilizada no projeto de ATS, embora se tenham tecnologias de mercado que ofereçam a automação de UI por meio de *record-and-play*, este tipo de ferramenta dispõe de *scripts* de testes padrão de difícil manutenibilidade. Assim, por consequência, não se adequa à proposta do presente modelo que visa integrar o desenvolvimento da automação e toda sua prática ao time de desenvolvimento Scrum.

Dito isto, o subcapítulo 6.2 aborda a visão geral do modelo, a evoluindo nas seções subsequentes para uma visão detalhada sobre cada fase do processo em conjunto às práticas necessárias de apoio à implementação do mesmo.

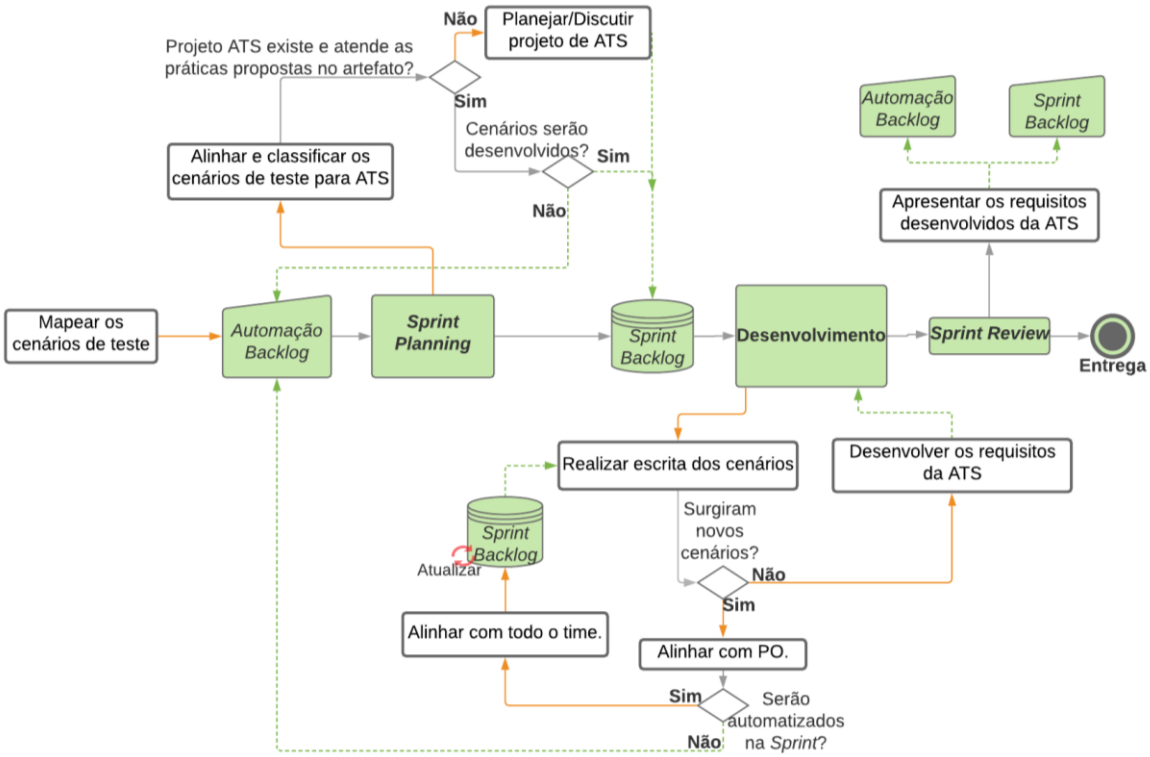
6.2 VISÃO GERAL DO ARTEFATO PROPOSTO

Embora a pirâmide ideal de Cohn (Figura 4) evidencie a automação voltada inicialmente para camadas de testes unitários e de serviços, o mesmo não desmotiva que seja realizada a automação para camada de interface gráfica. A literatura já provê um ferramental de boas práticas ligadas ao desenvolvimento da automação nas duas primeiras camadas, até mesmo por serem estas diretamente ligadas ao um processo de escrita do código.

Contudo, testes de interface, tanto para novas funcionalidades quanto para testes regressivos, promovem ainda uma alta demanda de atividade manual de testes para as equipes. E, conforme observou-se na evidenciação da problemática desta pesquisa quanto às dificuldades encontradas com o processo de implementação de ATS em equipes ágeis, o maior ônus à automatização de testes está concentrado nesta camada.

Diante disto, montou-se a proposta de um modelo de processo para fins de atender a implementação da ATS para UI (MPATS – UI), em equipes de desenvolvimento de software que empreguem de metodologias ágeis. A Figura 13 ilustra de forma geral a modelagem do referente artefato quanto ao fluxo do processo da automatização aplicado no Scrum. Neste as setas contínuas em laranja apontam as saídas para as atividades do processo e as setas pontilhadas na cor verde demonstram as entradas dos artefatos das atividades para as fases do processo.

Figura 13 - Visão geral do modelo



Fonte: elaborada pela autora.

A fim de manter a sincronia do processo de ATS com Scrum, as atividades são consolidadas nas cerimônias que seguem no conceito de fluxo do mesmo: Planejamento da *Sprint*, Desenvolvimento, Reunião Diária e Sprint *Review*. Logo, todas as cerimônias são devidamente executadas, englobando o processo de ATS e dando evidência como requisito principal de entrada para o processo de automatização, os cenários de teste a serem automatizados. Estes, devidamente classificados quanto ao risco e ao valor de negócio pelo PO.

O artefato de entrada para o processo é o *Backlog* composto por itens de um Produto *Backlog* e itens de uma Automação *Backlog*. Logo, como atividades do processo da ATS têm-se: **Planejar/Discutir o projeto de automatização**, **Mapear e Classificar os cenários de testes para ATS**, **Escrever os cenários de testes para automação**, **Desenvolver os requisitos da automação**, **Alinhar novos cenários com *Product Owner***, **Alinhar novos cenários com todo o time**.

E, das atividades citadas, são esperados os seguintes artefatos de saída para o processo: o Sprint *Backlog*, contendo as atividades de desenvolvimento do produto e da automação, os cenários de testes para atualização na Automação *Backlog*, os incrementos de melhorias para o processo e por fim, as atividades prontas compondo o produto como um

todo. Esta composição será apresentada no decorrer com o detalhamento de cada etapa do modelo e exibida de maneira completa na Figura 17.

6.3 DETALHAMENTO DO ARTEFATO

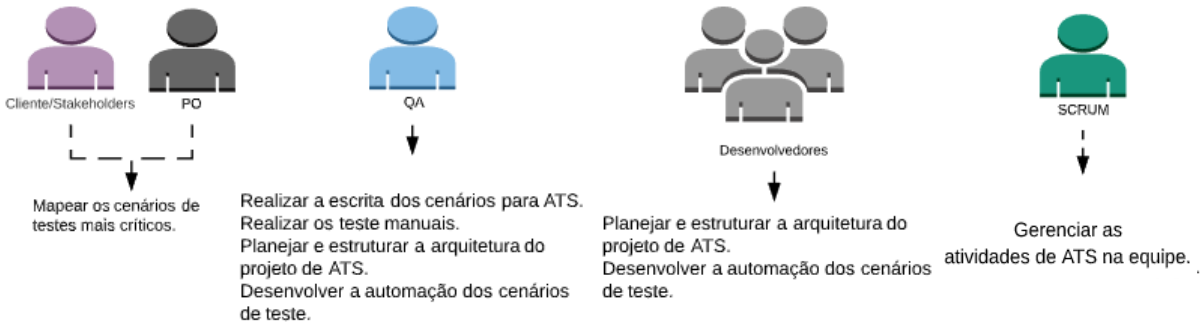
Nesta subseção é apresentado detalhadamente cada ciclo do processo com suas respectivas atividades, assim como, a abordagem do papel do QA na equipe e práticas adotadas para a sincronia do processo.

6.3.1 Papéis e responsabilidades

Para o Scrum há a definição de 3 papéis na equipe, dentro disto, para o modelo compreende-se a existência da figura do QA, como parte da equipe de desenvolvimento, está colocada em evidência no processo por suas responsabilidades estarem diretamente relacionada ao processo da ATS de UI. Explica-se esta abordagem quanto à visibilidade do QA no processo, assim como sua adoção na premissa para uso deste modelo, pela evidenciação dada na fase de conscientização deste trabalho quanto à existência deste papel na equipe, assim como nas leituras das publicações correlatas e em fontes da web, sendo, fóruns de discussões, artigos e revistas voltadas ao tema desta pesquisa. Compreendeu-se ainda no decorrer da pesquisa que este papel, além de ser a figura mais próxima do entendimento de requisitos para testes, está também diretamente consolida às atividades de testes manuais ou automatizados de UI, dentro das equipes.

Conjuntamente o proposto modelo assume então os seguintes personagens no desenho do processo: o Cliente/*Stakeholders*, o *Product Owner*, Scrum *Master* e o time de desenvolvimento agregando o QA (responsável pelos testes). Dito isto, as atribuições quanto às responsabilidades diante do modelo proposto estão evidênciadas na Figura 14, exceto responsabilidades das quais já são definidas pelo *framework* para os respectivos papéis. Optou-se pela definição das cores para cada papel para auxiliar na visualização quanto à participação de cada um no decorrer do processo.

Figura 14 - Papéis e responsabilidades no modelo



Fonte: elaborada pela autora.

Compreende-se para o modelo a utilização da Automação *Backlog* com a visão de um pacote que conterá os cenários mais críticos já mapeados pelo PO em conjunto com o cliente/*stakeholders* a partir dos requisitos da s novas funcionalidades para desenvolvimento ou qualquer requisito diretamente ligado à automatização de teste UI, como exemplo, construção ou arquitetura do projeto pela equipe, ou até mesmo, atividades de melhorias ligadas a este processo.

A atividade de mapear os cenários destinados à automação pelo PO, é proposta com o objetivo de trazer ao processo de ATS maior valor agregado ao negócio quanto à sua entrega ao final de cada *Sprint*, além também de proporcionar à equipe uma forma de conduzir um plano de testes coerente ao negócio. Porém, não limita que a equipe contribua com a observação de novos cenários durante o processo, mas caberá ao PO decidir a sua evolução para a entrega.

Com a observação da problemática sobre a ausência de documentação, importante ressaltar que o modelo de processo proposto para a documentação de requisitos apropria-se da adoção de práticas do ATDD, referenciado no Capítulo 5, tais como:

- Focar em um conjunto de testes de aceitação, garantindo a maior entrega de valor do negócio;
- Tornar os requisitos de uma funcionalidade em especificações executáveis através de uma linguagem de domínio do time.

As práticas do ATDD ajudam a prover requisitos mais maduros e diretamente ligados aos cenários de testes que serão automatizados. Dessa forma, o QA da equipe ganhará mais tempo para a execução do desenvolvimento dos requisitos ligados ao processo de ATS. Este irá transcrever ou criar o cenário de teste no projeto de ATS em uma linguagem de requisitos já definida desde o início do processo, assumindo-a como documentação viva para o projeto.

Considerando ainda a premissa de adotar um projeto de automação a partir de uma linguagem de código de desenvolvimento utilizada pela equipe e, por consequência, gerar um *framework* de teste independente, percebe-se então que este projeto de ATS provê equivalência quanto à sua estruturação a um projeto de desenvolvimento de software. A partir disto, provem a necessidade de o QA possuir habilidades com o desenvolvimento de código de software para poder, não só contribuir nas atividades de desenvolvimento da ATS, mas também, na estruturação do projeto quanto ao seu planejamento e evoluções no decorrer das *Sprints*. Contudo, antecipado a qualquer atividade de desenvolvimento da automatização do teste de UI, fica de responsabilidade do QA a escrita destes cenários no projeto em uma linguagem natural dentro do projeto de automação de teste.

Os desenvolvedores estarão comprometidos em auxiliar nas atividades de desenvolvimento de teste, assim como, na criação e estruturação do projeto de teste. Para tal, todos deverão participar de cada etapa do processo de ATS, provendo então a sua imersão no conhecimento sobre o mesmo. O Scrum *Master* passa a gerenciar as atividades de ATS em conjunto com as atividades de desenvolvimento do produto, potencializando maior envolvimento do time com o todo ao processo de automatização.

Para compreender cada etapa do modelo e suas respectivas atividades, as próximas subseções abordarão o detalhamento do processo de ATS de UI iniciando na *Sprint Planning* seguida dos artefatos necessários de entrada e saída da mesma.

6.3.2 *Sprint Planning*

Conforme descrito na subseção anterior, a Automação *Backlog* deve conter os itens que serão discutidos e compreendidos na cerimônia de *Sprint Planning* (Figura 15) diretamente ligados ao processo de ATS de UI. Aqui, além de discutir e planejar as tarefas de desenvolvimento do sistema, a equipe deve conhecer os cenários que serão testados e automatizados, também são livres para expor as opiniões sobre possíveis testes que não constam na Automação *Backlog*, mas cabe ao PO classificar os cenários para a o Sprint *Backlog*.

Ainda na *Sprint Planning*, haverá a atividade de planejamento do projeto de ATS de UI, caso a equipe ainda não tenha um projeto implementado, ou a discussão sobre o mesmo se houver a necessidade de atualizações quanto à estrutura, inclusive adequá-lo às premissas da proposta de modelo. Para tal, a equipe deverá ter os cuidados de impor a este projeto boas

práticas ligadas ao desenvolvimento de software, além dos pré-requisitos ligados a este modelo, sendo eles:

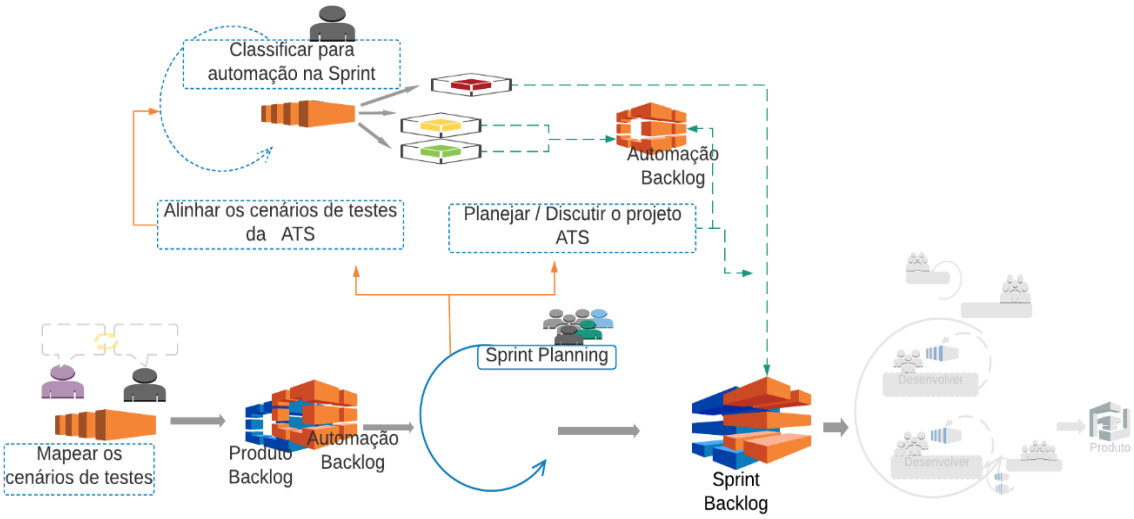
- Escolher uma linguagem de programação na qual toda a equipe seja capaz de contribuir e tenha domínio;
- Disponibilizar o código fonte em repositório compartilhado para toda a equipe de desenvolvimento adotando a prática de integração contínua;
- Prover a saída de relatório sobre execução da automação;
- Arquitetar o ambiente de testes;
- Integrar a execução do teste automatizado à *pipeline* de entrega;

É importante, em um primeiro momento, planejar a alocação de um desenvolvedor mais experiente junto ao QA para a montagem do projeto e desenvolvimento dos primeiros cenários, caso este tenha pouco conhecimento de programação. Contudo, a partir de referências de estudos lidos para esta pesquisa, foi possível inferir que quando há a necessidade da estruturação inicial do projeto de automatização de teste, as equipes tendem a utilizar em média de 1 a 2 ciclos do Scrum para o feito. Logo, estas atividades também deverão alimentar a Automação *Backlog* para que sejam evoluídas nas demais *Sprints*.

Embora ocorra um emprego maior de tempo nesta fase inicial, o ciclo de classificação e discussão dos cenários de testes deve continuar, pois, desta forma, assim que o projeto estiver pronto já se tem uma independência para realizar a automatização dos testes até que o processo sincronize totalmente com os requisitos correntes da *Sprint*. Além, também, de auxiliar a equipe quanto à visualização dos testes que serão executados no decorrer do projeto de automação, dando para a mesma uma base de informações usadas para planejar melhor as ferramentas necessárias no projeto de ATS de UI.

Logo, é preciso que, se houver já funcionalidades desenvolvidas no produto e que careçam de automatização de testes, estas deverão ter seus cenários classificados pelo PO e apresentados na *Sprint Planning*, dando a todos essa visibilidade. E, se os testes não entrarem para o desenvolvimento na *Sprint* corrente, deverão alimentar a Automação *Backlog* seguindo a ordem de priorização delegada pelo PO. Importante ressaltar que o Scrum não provê em seu processo dois *Backlogs* e que a utilização da denominação de Automação *Backlog* é para evidenciar o processo desta proposta à visualização destes itens dentro do *Backlogs* real de trabalho da equipe, assim como toda a contextualização no ciclo. Pois compreende-se ao final do processo uma única entrega contribuída por ambas atividades.

Figura 15 - Visão da *Sprint Planning*



Fonte: elaborada pela autora.

6.3.3 Desenvolvimento e Diária

Na fase de desenvolvimento para melhor ilustrar as atividades do QA na Figura 16, tem-se o Quadro QA. Este corresponde às atividades diretamente relacionadas a este papel no ciclo de desenvolvimento no espaço de tempo onde as primeiras funcionalidades do produto estão sendo desenvolvidas. Assim, adotando as práticas citadas do ATDD, aqui será desenvolvida a escrita dos cenários dentro do projeto de testes, que, para tal, utiliza-se de algum *framework* que permita essa integração de linguagem humana com o código, como por exemplo, o BDD citado na subseção 5.3.3. Com o cenário escrito, é gerada então a atividade de escrita de código da ATS de UI que retorna para o ciclo desenvolvimento, tornando-se disponível para que qualquer desenvolvedor da equipe realize a tarefa.

Observa-se que a partir da tarefa de escrita pelo QA, pode ocorrer o surgimento de novos cenários de teste não previstos anteriormente e, através da estrutura de decisão proposto no modelo, “Surgiram novos cenários?”, deve ser conduzida a próxima atividade. Sendo “sim”, segue para a atividade de alinhar com o PO, que por sua vez, realizará a classificação destes para automação, de forma que, quando crítico para entrega atualiza o *Sprint Backlog*, removendo outros itens quando necessário, ou, simplesmente atualiza a Automação *Backlog*. Ocorrendo a primeira, deve ser alinhado com todo o time de desenvolvimento sobre os novos cenários de testes a serem automatizados, pois dentro do modelo compreende-se que é de todos a responsabilidade de contribuir na atividade de automatização e então estes deverão estar sempre sincronizados com o objetivo da mesma.

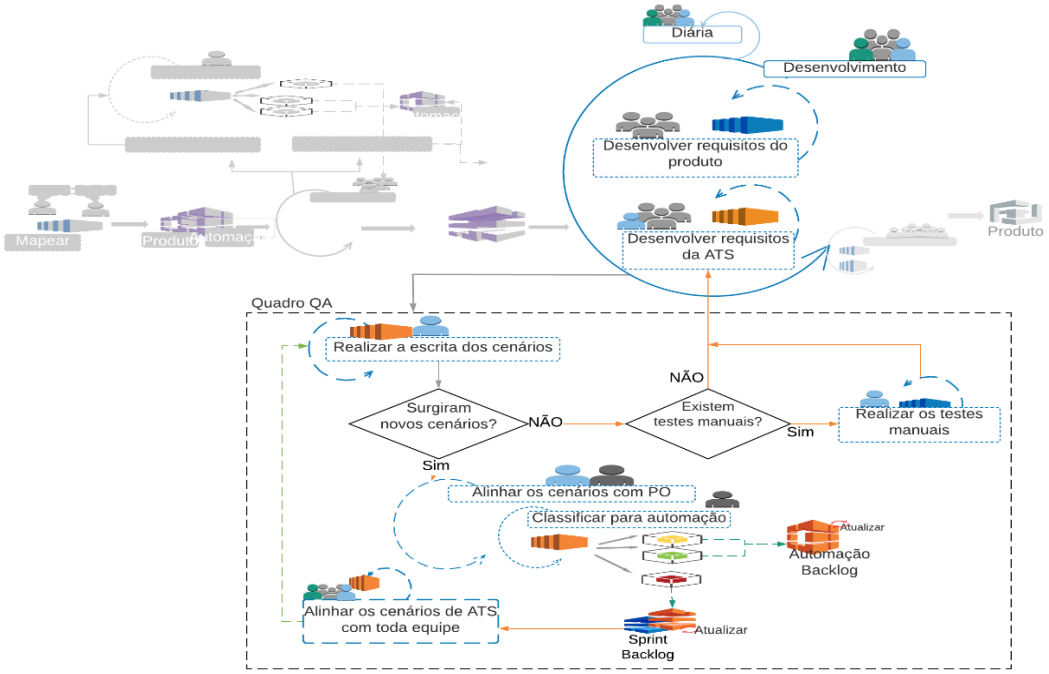
E, sendo “não”, apenas atualiza o *Backlog* para próximas demandas, seguindo uma classificação a cargo do PO.

Ainda no Quadro do QA e utilizando da estrutura de decisão “Existem testes manuais?”, se “sim” o QA realiza os testes os manuais do produto conforme a liberação da atividade, porém este tipo de teste deve ser minimizado a cada ciclo de *Sprint* com a colaboração de todo o time no desenvolvimento com o automatizado. Não havendo testes manuais o QA prossegue com a atividade de ATS dentro do ciclo de desenvolvimento. Essa tratativa pelo processo, objetiva também dispor ao QA mais tempo para se dedicar ao desenvolvimento da automatização. Por isso, a importância sobre as habilidades de desenvolvimento com linguagem de programação do QA, para que o mesmo consiga dar vazão às atividades de automatização junto ao time de desenvolvimento, sendo ele o ponto referência dos requisitos da automação.

Observa-se esta abordagem de contribuição no ciclo de desenvolvimento, a fim de consolidar a resolução pelo proposto modelo das problemáticas diretamente ligadas à ausência de uma equipe dedicada e treinada para automação, assim como, a dificuldade de manutenção do projeto de ATS ligadas às diversas alterações do produto, dispondo do auxílio de todos com a ATS de UI.

Contudo, para atender a essas problemáticas, é necessário também respeitar a premissa deste modelo quanto ao projeto estar na linguagem do time e aos cenários de testes serem bem consolidados pelo PO junto ao cliente/*stakeholders*.

Figura 16 - Visão do ciclo de Desenvolvimento



Fonte: elaborada pela autora.

A cerimônia diária do Scrum segue com o seu processo normal no decorrer do desenvolvimento. Porém, frisando que agora qualquer membro da equipe deve reportar na mesma sobre o processo de automatização e cabe ao Scrum *Master* gerenciar as atividades de ATS, assim como qualquer empecilho que a mesma possa ter no decorrer da *Sprint*.

6.3.4 *Sprint Review* e Entrega

Na cerimônia de Sprint *Review*, a abordagem da automação deve ser apresentada para todos do time, incluindo o cliente/stakeholders. Aqui podem ser gerados novos incrementos para o Automação *Backlog*, tais como, evoluções quanto a relatórios, sugestões de melhorias no projeto, ou ajustes para a entrega corrente, gerando nesta última a consequência retomando a execução do ciclo com um todo ou, ainda, diretamente ao desenvolvimento se os ajustes estiverem maduros o suficiente quanto a seus entendimentos para com todos da equipe.

A visibilidade da ATS de UI na *Sprint Review*, ajuda a compor o valor de senso comum quanto à responsabilidade da mesma para com toda a equipe, deixando de ser apenas uma etapa dentro do ciclo. Também é evidenciada para o cliente quanto à sua entrega de valor pela redução de riscos ao negócio com a clara percepção das atividades de automatização estarem consolidadas ao processo de desenvolvimento do produto e, consequentemente, agregadas de forma contínua e progressiva na entrega do mesmo.

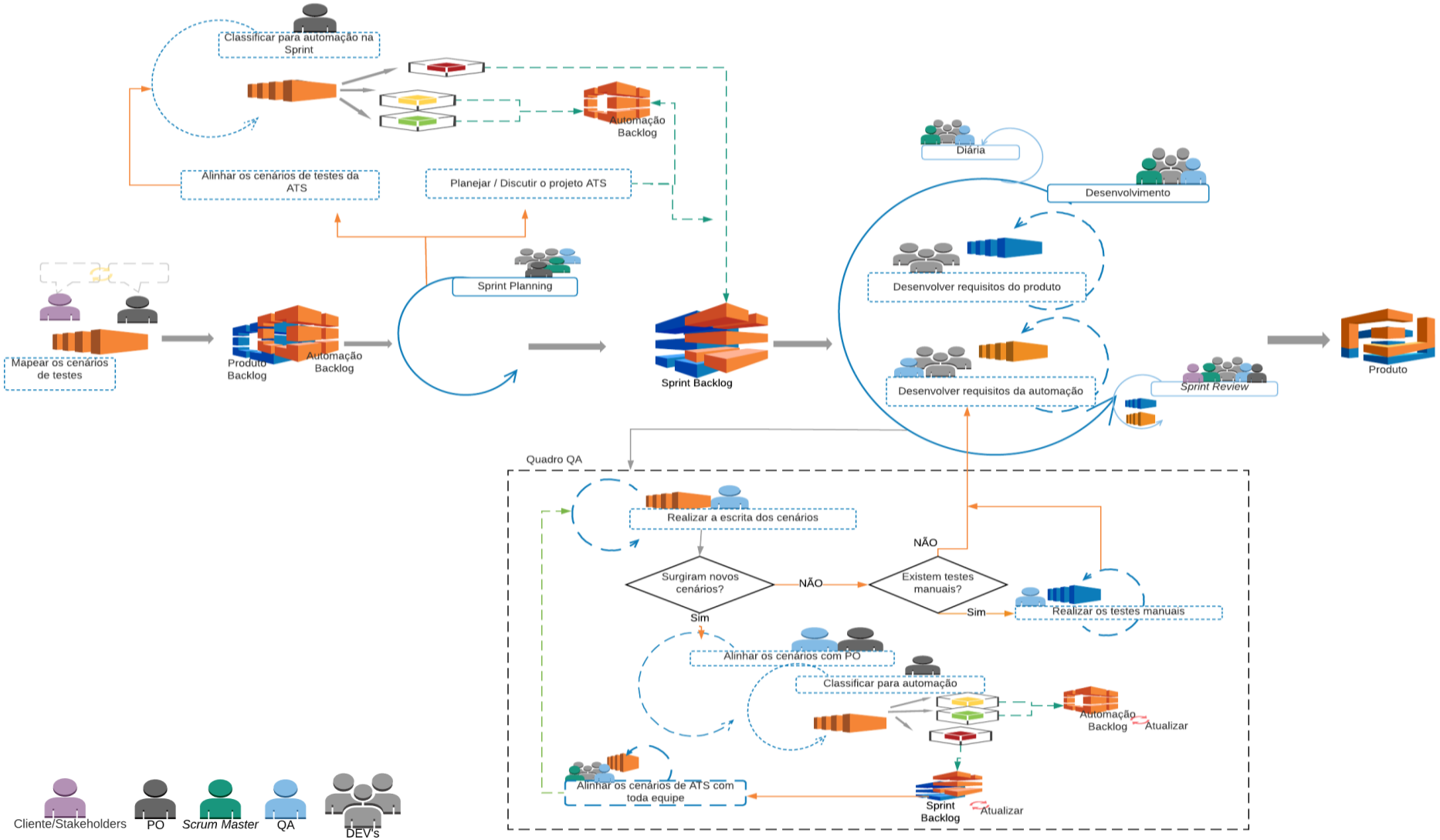
Contudo, a *Sprint Review* não gerando qualquer alteração na entrega corrente, por fim, tem-se definida a entrega do produto consolidada aos requisitos de entrada da automação e desenvolvidos por toda a equipe. A Figura 17 demonstra esse processo detalhado com os artefatos de entrada e de saída nas atividades e os papéis participantes íntegros ao processo.

A modelagem deste artefato de pesquisa teve por objetivo propor ao time de desenvolvimento ágil um processo para conduzir a ATS de UI em paralelo ao ciclo de desenvolvimento do produto, produzindo um sentimento de contribuição com esta tarefa a todos da equipe. Embora tenha a adoção de um personagem principal ligado a esse processo, a atividade de desenvolvimento da automação fica à disposição de todos da equipe, assim como, a contribuição de todos desde a concepção do projeto de automatização, melhorias e mudanças do mesmo a cada *Sprint*.

Importante salientar que o modelo proposto visa o aumento progressivo da cobertura de testes automatizados a cada ciclo que, por consequência, contribuirá com o ganho de tempo para as demais *Sprints* quanto à atividade de teste manual, e, sabendo que testes estão diretamente ligados à qualidade, contribui também na qualidade da entrega. Com isto, é dada

por completa a apresentação do artefato desta pesquisa. No próximo capítulo será apresentada a construção do instrumento de pesquisa e o roteiro de entrevista, ambos para verificar a viabilidade de adesão do artefato proposto.

Figura 17 - Visão do MPATS – UI detalhado



Fonte: elaborada pela autora.

7. AVALIAÇÃO DO ARTEFATO

A pesquisa apoiada pelo *Design Science Research* exige que o pesquisador exponha evidências de que o proposto artefato poderá ser utilizado no contexto externo real. Para tal, o presente trabalho utilizou do elemento da pesquisa-ação, que consiste a outro método de pesquisa, porém sugeridos por Dresch, Lacerda e Antunes Jr (2015) para esta etapa. Os autores explicam que este tipo de elemento objetiva propor soluções e explicações a problemas evidenciado em um certo tipo de sistema, produzindo assim, conhecimento. Contém nas suas etapas de condução de pesquisa a coleta de dados, compreendendo o contexto e o propósito quanto à pesquisa, e a análise dos mesmos, conforme exibidos na Figura 18.

Figura 18 - Técnicas de coleta e análise de dados

Objetivo	Técnicas
Coleta de dados	Documental Bibliográfica Entrevistas Grupo focal Questionários Observação direta
Análise dos dados	Análise de conteúdo Análise do discurso Estatística multivariada

Fonte: Dresch, Lacerda e Antunes Jr (2015, p.33).

Das técnicas elencadas, esta pesquisa utilizou para validação do artefato, o questionário e a entrevista para a coleta de dados. A primeira é composta por uma série de perguntas definidas pelo pesquisador de acordo com o objetivo da pesquisa e, cabe ao mesmo, definir a estratégia de aplicação de forma que compreenda o público externo diretamente relacionado ao interesse da pesquisa (DRESCH; LACERDA; ANTUNES JR, 2015).

A segunda pode ser caracterizada em dois tipos, conforme Dicicco-Bloom e Crabtree (2006), citados por Dresch, Lacerda e Antunes Jr (2015, p.34). Sendo um tipo como entrevista padronizada/estruturada, onde o pesquisador define um roteiro para orientação prévia da entrevista, sem sofrer quaisquer tipos de modificações para adequação a diferentes situações. O outro, como entrevista despadronizada/não estruturada, na qual o entrevistado fica livre a propor situações conforme julga adequado, explorando os assuntos de maneira menos formal.

Por compreender que o contexto desta pesquisa é guiado a um tipo específico de ambiente externo para aplicação do artefato proposto, o presente trabalho aplicou a entrevista padronizada/estruturada, porém o entrevistado teve total liberdade para expor qualquer tipo opinião sobre o artefato no decorrer da entrevista, mas mantendo o padrão de passar por todas as questões elencadas às características do processo.

Dito isto, as seções 7.1 e 7.3 apresentarão a proposição da entrevista utilizada com os especialistas, e do questionário aplicado ao viés de pesquisa de campo aberto ao público externo.

7.1 PROPOSIÇÃO DO ROTEIRO DA ENTREVISTA

Com o objetivo de verificar a viabilidade quanto ao uso do artefato produzido a âmbito qualitativo, foram realizadas entrevistas com especialistas conduzidas por um roteiro elaborado a partir das características do artefato e das práticas nele adotadas. As questões do roteiro e os propostos objetivos requeridos a cada uma, são apresentadas no Quadro 3.

Quadro 3 - Questões da entrevista e objetivos

(continua)

Questões	Objetivo
Seção: Perfil do entrevistado	
Identificação do participante	Elencar o nível de conhecimento do especialista de acordo com os assuntos pertinentes à pesquisa.
Nome:	
E-mail:	
Escolaridade:	
Em projetos de automatização de testes, qual(s) a(s) atuação(ões): (podendo marcar mais de uma)	
Tempo de experiência com metodologias ágeis:	
Tempo de experiência trabalhando com equipe de desenvolvimento de software:	
Nível de experiência em automação de testes:	
Nível de experiência com Scrum:	
Fale um pouco sobre suas especializações e certificações:	
Seção: Contextualização	
Esta seção tratou da contextualização do entrevistado quanto às problemáticas: (1) Pouco tempo para desenvolver a automação de testes. (4) Diversas alterações no produto. (2) Ausência de uma equipe dedicada. (5) Ausência de documentação. (3) Ausência de uma equipe treinada a automação de testes. (6) Processo de testes não definido.	Dispor ao especialista a problemática da pesquisa v.s. a solução proposta para a condução das etapas seguintes da entrevista.
E a apresentação do artefato, explicando o processo detalhamento partir da Figura 17.	
Seção: Papéis e documentação	
Qual sua opinião sobre o papel do QA (ter habilidades de desenvolvimentos) e das responsabilidades da equipe dentro do modelo proposto.	Validar a visão sobre os papéis e suas responsabilidades para com o processo.

Quadro 4 - Questões da entrevista e objetivos

(conclusão)

Questões	Objetivo
Qual sua opinião sobre os itens abaixo quanto a documentação para o processo de ATS de UI no ciclo? -Focar em um conjunto de testes de aceitação garantindo a maior entrega de valor do negócio; -Tornar os requisitos de uma funcionalidade em especificações executáveis através de uma linguagem de domínio do time	Verificar o impacto em aderir técnicas do ATDD para o processo de ATS de UI.
Seção: Automação *Backlog* e *Sprint Planning*	
Qual sua opinião sobre a visão Backlog Automação com atividades de automatização no processo.	Validar pela percepção do especialista a possibilidade de adesão desta parte do processo em contexto real e assim como a consciência de uma Automação *Backlog* para com o processo.
Questão aberta para o entrevistado, exibindo a imagem contendo a parte do processo focado na Automação *Backlog* e *Sprint Backlog*	
Sobre os fatores para o projeto de ATS no processo. -Escolher uma linguagem de programação na qual toda a equipe seja capaz de contribuir e tenha domínio; - Disponibilizar o código fonte em repositório compartilhado para toda a equipe de desenvolvimento adotando a prática de integração contínua; - Prover a saída de relatório sobre execução da automação; - Arquitetar o ambiente de testes; - Integrar a execução do teste automatizado a pipeline de entrega;	Verificar se os requisitos para planejamento do projeto de ATS de UI colaboram com o processo, assim como, evidenciar a visão sobre este ser tratado com equivalência a projeto de software.
O processo sugere a alocação de um desenvolvedor mais experiente junto ao QA, caso este tenha conhecimentos muito básicos, para a montagem do projeto e desenvolvimento dos primeiros cenários. Qual sua visão sobre essa abordagem?	Compreender se esta iniciativa contribui ao processo de ATS de UI ou poderia onerar em algum ponto do ciclo com a alocação do dev.
Seção: Quadro QA e Desenvolvimento	
Questão aberta para o entrevistado, exibindo a imagem contendo a parte do processo focado no Quadro QA: Quais suas observações quanto às abordagens para esta parte do processo?	Validar pela percepção do especialista a possibilidade adesão desta parte do processo em contexto real.
Com os artefatos da atividade de escrita de cenários prontos e o desenvolvedor contextualizado sobre a mesma desde o início do processo, pode contribuir realizando o desenvolvimento dos requisitos da ATS de UI. Qual sua opinião sobre esta afirmação quanto ao processo?	Compreender se esta iniciativa é viável e se proporciona ganho quanto ao tempo com o desenvolvimento das atividades de ATS de UI.
Seção: *Sprint Review* e Entrega	
Quanto à visibilidade da ATS na Sprint Review: sobre estas afirmações, qual sua opinião? * Cria valor de senso comum quanto à responsabilidade da mesma para com toda a equipe, deixando assim de ser apenas uma etapa dentro do ciclo. * Proporciona evidências para o cliente/stakeholders sobre a entrega de valor quanto à redução de riscos para o negócio, por estarem as atividades de automatização de testes de UI consolidadas ao processo de desenvolvimento do produto. * Proporciona também contextualizar a todos sobre o ganho de tempo nas demais Sprints na execução de testes, gerando entrega mais rápidas e aumentando a cada ciclo a cobertura de testes automatizados.	Validar pela percepção do especialista se diante do modelo estas afirmações podem se consolidar nesta cerimônia.
A Sprint Review não gerando alteração na entrega corrente, por fim, tem-se definida a entrega do produto consolidada aos requisitos de entrada da automação e desenvolvidos por toda a equipe. Qual sua opinião dobre esta visão na entrega final?	Validar pela percepção do especialista se há ganhos na entrega com a utilização do processo.

Fonte: elaborada pela autora.

Como o modelo proposto visa ser utilizado para implementação da ATS de UI em ambiente ágil consolidado ao *framework* Scrum, buscou-se pela opinião de profissionais com experiência e conhecimento sobre ATS de UI, Metodologias Ágeis, Scrum e diretamente relacionados à área de desenvolvimento de software. A descrição completa do perfil de cada especialista está constituída na seção 7.2, a qual também evidencia a análise sobre esta etapa da validação relacionada às entrevistas.

7.2 RESULTADO E ANÁLISE DAS ENTREVISTAS

Para manter o respeito quanto ao anonimato dos dados dos participantes, os mesmos serão denominados como especialista 1, especialista 2, especialista 3 e especialista 4. O Quadro 4 elenca o perfil dos mesmo a cada denominação utilizada.

Quadro 5 - Perfil dos especialistas

Especialista	Especializações e certificações	Perfil profissional	Considera seu nível de experiência com:	
			Automação de testes	Scrum
Especialista 1	Mestre em Computação Aplicada, Bacharel em Ciência da Computação. Palestrante e coordenador nas trilhas de teste do TDC (Florianópolis, São Paulo e Porto Alegre).	Trabalha entre 9 a 10 anos na área de qualidade de software e atua como engenheiro de testes automatizados.	Alto	Médio
Especialista 2	Bacharel em Administração de Empresas, MBA em gerenciamento de projetos. Certificado em PMP Project Management Professional(PMP), Profissional Scrum Master PSMI I e PSMI II e Lean Inception Facilitador (CLF).	Trabalha entre 6 a 10 anos com equipes de desenvolvimento de software e 5 anos diretamente com ágil. Atualmente atua como Coach Agile.	Médio	Muito Alto
Especialista 3	Bacharel em Sistema de Informação, Pós Graduação em Segurança Ofensiva e Inteligência Cibernética (em andamento).	Trabalha entre 3 a 5 anos com automatização de testes e atualmente atua como Líder de Qualidade de software diretamente com testes automatizados, prestando suporte e gerenciando outras equipes.	Alto	Médio
Especialista 4	Doutorado em Ciência da Computação, Mestrado em Ciência da Computação, Especialização em Qualidade de Software, Graduação em Análise e Desenvolvimento de Sistemas.	Trabalha entre 9 a 10 anos na área de qualidade de software. Atualmente leciona na área de tecnologia e atua também como Analista de Testes.	Muito Alto	Médio

Fonte: elaborada pela autora.

Com a condução da entrevista aplicada através de um padrão por roteiro em seções, foi possível avaliar a percepção dos mesmos por partes ao processo, a mesma se aplica para a condução da análise, porém considerando após a seção de contextualização.

7.2.1 Papéis e Documentação

Evidenciou-se para a primeira seção a concordância de todos quanto às responsabilidades do QA no processo proposto, concordando o especialista 2 que para o Scrum "o QA deve integrar o time de desenvolvimento e não ser uma figura separada". Já os especialistas 1 e 3 enfatizaram sobre a necessidade de o QA possuir o conhecimento de linguagem de programação para o processo de ATS de UI, sendo na visão dos mesmos, uma característica indispensável para a figura ligada a automatização de testes. O especialista 4 ainda declarou sobre sua atividade enquanto QA "eu como QA hoje não posso pensar apenas no meu teste manual, embora ele ainda seja o prioritário dentro do meu atual modelo de trabalho, sempre que tenho tempo realizo a automatização dos cenários".

Quanto à conscientização da documentação, todos concordaram que o formato proposto traz ao processo maior ganho, salientou o especialista 1 que "É importante que os requisitos sejam sempre os mais claros possíveis, garantindo que não ocorram problemas de comunicação entre os envolvidos (problema mais comum em ambientes de desenvolvimento de software) ". E, complementou o especialista 3 dizendo que " Do ponto de vista da automação sempre tento implementar o BDD para documentar os testes mesmo sem haver a dependência de uma documentação boa de requisitos…". Para o especialista 3 são itens fundamentais e dão sentido ao processo.

7.2.2 Automação *Backlog* e *Sprint Planning*

Quando expostos a visão da Automação *Backlog* o especialista 2 chamou a atenção para a visão sobre o Scrum não ter a percepção de dois *Backlogs* na *Sprint*, então foi explicado ao mesmo que este tratamento é realizado a nível de visualização do processo, mas que a proposta respeita um único *Backlog* no processo. Dito isto, o mesmo complementou entendendo que "teria a visão de apenas um *Backlog*, sendo importante englobar as ambas atividades".

Ainda, ao questionar o especialista 2 sobre a atividade de o PO mapear os cenários de teste, este observou que isto pode gerar uma sobrecarga ao PO podendo fazer com que ele não contribua com esta atividade como previsto, enfatiza o especialista 2 quando comparado ao seu contexto atual de trabalho, mas concorda que "ao pensar em qualidade de entrega e melhoria contínua, ter esta organização prévia quanto aos cenários impacta na entrega de valor do produto.". O especialista 1 observou que "A inclusão das atividades de automação no *Backlog* proporciona maior visibilidade para o time. ", o especialista 3 concorda com a

entrada dos itens no *Backlog* e salientou “Vejo ser ideal ter essa percepção no processo.”. O especialista 4 achou apta esta visibilidade e percebeu valor para o processo de ATS de UI, porém complementou explicando que “ atualmente não temos esta definição do PO e sim pelo analista de sistemas e por mim, até mesmo pela complexidade da aplicação, minha dúvida quanto a este mapeamento seria se o PO teria o total conhecimento para poder mapear os cenários mais críticos”. Contudo o especialista 4, enfatizou que não há a adoção de Scrum no seu contexto atual de trabalho e ainda empregam uma metodologia cascata.

Sobre tratar do processo de ATS de UI na *Sprint Planning* para os especialistas 1 e 2 caracteriza um cenário positivo para o processo, declarou ainda o primeiro “Entendo que é indispensável o alinhamento constante entre o time quanto às atividades a serem realizadas”. Porém o especialista 3, embora concorde com esta tratativa afirmando que “O ideal seria o conhecimento dos testes sempre ficar disseminado a todos da equipe”, chama a atenção para a percepção quanto à questão cultural de os desenvolvedores não quererem se envolver com as demandas da qualidade. O mesmo demonstrou também preocupação com o fato de não ter requisitos da automação alinhados com as funcionalidades atuais da *Sprint*, e complementou dizendo ver este fato “…como agravante para a demanda atual da funcionalidade sendo diferente da de testes automatizados” e explicou por estar “dividindo o foco entre a demanda atual com a demanda passada.”. O especialista 4 contribuiu dizendo que em complemento a esta abordagem poderia haver ainda um processo de refinamento que antecedesse a *Sprint Planning*, pois ressalta novamente que pela complexidade de alguns softwares, por vezes, poderia tomar muito tempo da *Planning* ou até mesmo sair sem definição, mas apoiou a proposta do tratamento para com todos sobre a ATS de UI nesta cerimônia.

Abordada também nesta seção, a questão relacionada à atividade de planejamento do projeto de ATS de UI elencando aos especialistas os itens chaves a serem tratados do mesmo com a equipe, conforme apresentado na subseção 6.3.2, diante disso todos assentiram quanto aos itens. O especialista 1 salientou que “É um diferencial para o projeto que todos os envolvidos sejam capazes de apoiar na automação dos testes.” e complementa sobre um dos itens que tratava da entrega de relatórios pela automação, “É essencial que se dê visibilidade de qual o valor real de negócio que os testes de regressão automatizados entregam, seja através de relatórios de cobertura de código ou de cobertura de funcionalidades”. O especialista 3 declarou que “Na forma mais enxuta este é o mínimo a ser planejado para o projeto, principalmente se o foco for em que a automação ocorra em paralelo com o desenvolvimento do produto”. Os demais concordaram e entenderam que em vista da

proposta, seriam essenciais estes itens para o projeto, assim como, o conhecimento destes por todos os envolvidos.

A última questão abordada nesta seção, tratou da sugestão em que o artefato traz sobre um desenvolvedor mais experiente alocado junto ao QA, para o especialista 1 é "um diferencial ter uma referência técnica apoiando o QA, mas acredito que ter um profissional com um conhecimento técnico bom sobre a linguagem utilizada como apoio já traz grandes vantagens para a construção dos testes, quando ambos os profissionais trabalham em par". O especialista 3 salientou que não deve ser uma premissa ao processo ter esta alocação, o que de fato não é, mas ao ser contextualizado sobre ser apenas a colocação como uma sugestão, o mesmo demonstrou afetividade com a ideia, porém salientou dizendo "… entendo ser útil para acompanhar o QA quando o mesmo não tenha uma *skill* (habilidade) de desenvolvimento muito aprimorada.".

Ainda pelo mesmo, trouxe uma contribuição para o processo enfatizando que "Mais do que acompanhar um QA no desenvolvimento, seria a participação do desenvolvedor com *code review*, para ter cada vez mais maturidade/qualidade no código da automação. ", o que demonstra alinhamento quanto à observação do especialista 2 "Acho importante esse apoio, porque vai gerar maturidade no desenvolvimento da automação.". O especialista 4 assemelhou sua percepção aos alheios.

Embora com algumas observações elencadas nesta seção pelos especialistas, em um modo geral, foi possível evidenciar que todos aprovaram a forma de condução da cerimônia *Sprint Planning*, a visibilidade com a Automação *Backlog* e as demais abordagens citadas, o que demostra a coerências desta etapa do modelo para a utilização em contexto real.

7.2.3 Quadro do QA e o ciclo de Desenvolvimento

Sobre a visibilidade do Quadro do QA no modelo, os especialistas concordaram quanto à sua estrutura de decisão sugerida e o fato de ainda tratar o teste manual no processo. O especialista 1 chamou a atenção de forma positiva quanto o envolvimento do PO dizendo "Outro ponto positivo é a visibilidade que o modelo traz para o PO" e complementou que dessa forma o mesmo "tem à sua mão mais algumas maneiras de medir a evolução do produto ao longo do desenvolvimento". O especialista 4 demonstrou empatia a esta etapa e destacou ser essencial manter o tratamento do teste manual e complementou "vejo que dentro de um ciclo mais maduro a tendência é de reduzir o manual, porém não acredito que ele possa vir a ser uma atividade descartada por completo.".

Já o especialista 3, novamente chamou a atenção para o quesito cultural e afirmou “a questão de alinhar os cenários com toda a equipe pode impactar na questão cultural em relação ao desinteresse dos desenvolvedores quanto à qualidade. ”, porém complementou que no ciclo de desenvolvimento “É importante que para o desenvolvedor possa interagir com as atividades de automação o mesmo também tenha *skills* de qualidade também.”.

Quanto a esta última observação, o especialista 1 já compreende que há a possibilidade de o desenvolvedor realizar as atividades de ATS de UI apoiado ao fato de perceber “… que o modelo de inclusão de atividades de análise de testes desde a fase de concepção do projeto é um ponto diferencial para a melhora, tanto na comunicação entre os envolvidos (que estarão cientes desde a definição das funcionalidades de boa parte de como a funcionalidade será), quanto para a garantia de que não só os itens priorizados serão automatizados.”.

O especialista 2 aprovou esta iteração e salientou “A participação do desenvolvimento na automação atende a filosofia do Scrum. ”, porém acredita que depende muito da visão do cliente quanto ao valor da entrega para que se tenha essa sincronia, explicando sobre o equilíbrio entre “… visar mais a entrega dos requisitos do produto do que da automação. ”, mas enfatizou quanto ao modelo proposto que “Esse é o modelo ágil a ser seguido…” .

Ainda para mesma questão, o especialista 4 se opôs em um primeiro momento ao tentar adequar a visão do modelo em seu cenário em seu atual contexto de trabalho e complementou “ Entendo que o desenvolvedor deva se preocupar com o teste unitário, tenho um pouco de receio que eles possam forçar a passar algum teste no desenvolvimento, mas pode ser uma questão de amadurecimento sobre também, porém olhando para uma questão técnica eles teriam a total capacidade em colaborar com a atividade de automatização de UI.”.

Foi possível perceber quanto à proposta de elencar o desenvolvimento da automação diretamente com o desenvolvedor, existiu uma certa hesitação à ideia pelo especialista 3. Porém o mesmo esclareceu, no decorrer da entrevista, por haver o contraponto de uma cultura alimentada pelo medo de o desenvolvedor carregar de vícios de linguagem e fazer com que o teste seja forçado a positivo.

Diante disso, o mesmo novamente enfatizou o uso da atividade de *code review*, e, assim como os demais especialistas, concordou que esta etapa do processo é aderente quanto à sua usabilidade em um contexto real.

7.2.4 *Sprint Review* e Entrega

Na última etapa da entrevista foi possível perceber, principalmente dos especialistas diretamente ligados ao processo de automatização, um sentimento comum de satisfação ao notarem que o modelo propunha trazer a automatização para a fase da *Sprint Review.* A justificativa percebida se deu pelo fato de que neste ponto a ATS de UI se faria visível também para o cliente/*stakeholders*, o que segundo o especialista 3 "…isso daria a automação uma maior visibilidade na hora de mostrar ao cliente o que está sendo entregue, mostrando também as garantias em impactos com o sistema." e sobre um tom de euforia complementou "É necessário que os profissionais do projeto evidenciem a automação para o cliente como agregação de valor ao produto.".

Para o especialista 2 é essencial a visão da ATS na *Sprint Review* e justifica que "mostra ganho de qualidade no produto para todos e agrega valor ao uso da ATS de UI. A entrega contendo os requisitos da automação garante a qualidade da entrega do produto, e mostra a qualidade do processo em si. ". O especialista 4 compartilhou do mesmo sentimento quanto à evidenciação da automação na *Sprint Review* e a qualidade de entrega agregada pelo processo. Não houve observações ou críticas opostas para nesta parte da entrevista, mostrando-se também esta etapa apta quanto sua utilização no contexto externo.

Ainda ao final de cada entrevista, foram novamente elencados os problemas com os especialistas com o desígnio de compreender se as problemáticas elencadas no início poderiam ser solucionadas a partir do proposto artefato. Diante da percepção de todos em coerência com as premissas do processo, compreendeu-se que o modelo atende como solução para o propósito de implementação de ATS de UI em contexto ágil baseado em Scrum.

Assim, dada como finalizada esta etapa de verificação com as entrevistas, constatando que seus objetivos foram alcançados. A seção seguinte apresentará a construção do instrumento de pesquisa aplicado ao público externo de interesse, seguida da posterior análise dos dados coletados através do mesmo.

7.3 PREPOSIÇÃO DO QUESTIONÁRIO

Com o intuito de verificar a adesão do artefato por um âmbito numericamente maior, foi elaborado um questionário com perguntas fechadas que apresentou alternativas restringidas ao propósito de facilitar a posterior análise em relação à objetividade da pesquisa. As questões eram de cunho obrigatório e o respondente poderia marcar apenas uma

alternativa. Ao final do questionário foi mantida uma questão livre e descritiva, caso o respondente sentisse a vontade de contribuir com alguma observação.

No Quadro 5 estão apresentadas as questões em conjunto aos seus objetivos para com a pesquisa e o referente questionário aplicado encontra-se no Apêndice D deste trabalho.

Quadro 6 - Questões do questionário e objetivos

Questões	Objetivo
Seção: Perfil do respondente	
Trabalha em qual estado?	Verificar o nível regional de alcance do público alvo.
Trabalha em uma equipe de desenvolvimento de software?	Validar o público respondente ao ambiente externo ao qual o artefato foi proposto, bem como o conhecimento dos mesmo sobre área de pesquisa.
Como você avalia sua experiência com automação de testes de interface:	
Você possui conhecimento sobre metodologias ágeis?	
Você possui conhecimento sobre o *framework* Scrum?	
Seção: Responda ao questionário considerando sua última experiência com automatização de testes de interface, em relação à equipe:	
Qual seu papel?	Objetiva consolidar a validação sob a visão papéis definidos pelo Scrum.
Emprega metodologia ágil?	Buscar o alinhamento do ambiente da proposta v.s. ATS UI
Utiliza algum *framework* ágil para gerência de processos?	Parear a informação com o *framework* utilizado na proposta
Realiza testes de interface automatizados?	identificar o peso das respostas do respondente v.s. o contato com a ATS de UI
Seção: Na sua opinião sobre o processo de Automatização de Testes de Software (ATS) para interface gráfica	
Ter os cenários de testes a serem automatizados já mapeados pelo PO/Cliente impacta no processo de ATS?	Cada pergunta desta seção está diretamente ligada às principais características do artefato utilizadas para a resolução das problemáticas elencadas na conscientização. Facilitando a posterior análise para verificar a adesão do modelo.
Ter os cenários de testes a serem automatizados já classificados quanto à sua prioridade pelo PO/Cliente impacta no processo de ATS?	
Ter a documentação de requisitos, escrita em especificações executáveis através de uma linguagem de domínio do time exemplo, BDD, impacta no processo de ATS?	
Utilizar da *Sprint Planning* para planejar as atividades do processo de automatização impacta no processo de ATS?	
Concorda que um projeto de automatização de testes, quando criado em uma linguagem de desenvolvimento de domínio do time, facilita o desenvolvimento das atividades de ATS e a evolução do mesmo?	
No ciclo de Desenvolvimento, com o projeto de ATS disponível para todos da equipe, concorda que é possível um desenvolvedor atuar na atividade de desenvolvimento de testes automatizados?	

O responsável de testes da equipe, ao escrever os cenários de testes diretamente no projeto de ATS através de um *framework* para BDD, por exemplo, impacta o processo de ATS?	
Você acredita que a entrega consiste tanto de características/*features* de produto quanto da automatização?	
Seção: Questão de livre contribuição	
Sinta-se à vontade, caso queira deixar alguma observação sobre sua percepção em relação às características do processo de ATS abordadas no questionário.	Buscar a partir de uma visão livre do participante críticas ou observações quanto às características citadas nas questões, elencando posteriormente na análise com o modelo.

Fonte: elaborada pela autora.

A estratégia adotada para aplicação do questionário se deu através da divulgação na rede social *LinkedIn*, adotando por premissa buscar pessoas que pertenciam a área de desenvolvimento de software e elencando com perfis específicos quanto à profissão que estivessem compreendidas no formato Scrum. Com o intuito de dotar de uma maior solidez na consolidação da viabilidade do modelo quanto às suas problemáticas.

Dessa forma, o primeiro contato para a divulgação do questionário foi direcionado aos participantes que contribuíram com a construção da problemática do trabalho na etapa de conscientização do mesmo, feito este possível por ter optado pela padronizado quanto à forma de divulgação em ambas as etapas. Porém, foram ainda realizadas divulgações pela mesma rede como conteúdo público, mantendo em aberta a participação de qualquer interessado, e ainda, a divulgação em canais da comunidade de software, tais como GUTS-RS e QALadies de SP.

O questionário ficou disponível para admissão de respostas no período de 20/10/2019 a 25/10/2019, contemplando a participação de 294 respondentes. Isto posto, a seção 7.4 apresenta o resultado em conjunto à análise sobre a viabilidade de adesão do artefato constituída pela pesquisa de campo.

7.4 RESULTADO E ANÁLISE DO QUESTIONÁRIO

A pesquisa atingiu 14 estados do Brasil, concentrando o maior número de participantes no Rio Grande do Sul com 169, São Paulo com 69, Rondônia com 18 e Santa Catarina com 12, os demais ficaram entre 1 e 8. A visibilidade desta em conjunto ao total de 294 participantes, demostrou a percepção da dimensão quanto ao interesse do assunto do trabalho pelo contexto externo. Porém, para uma validação coesa aos objetivos do artefato, a

partir das respostas foram realizados filtros para denotar melhor o conhecimento do respondente quanto ao assunto de pesquisa, o resultado é apresentado na Tabela 4.

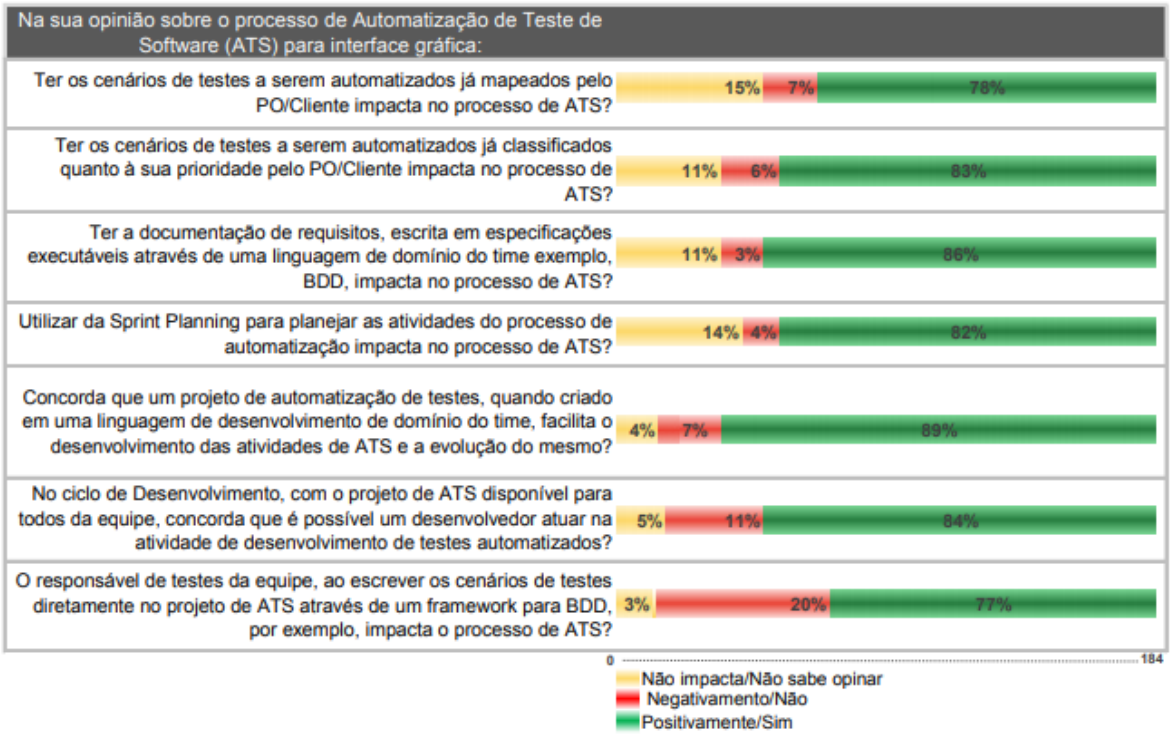
Tabela 4 - Quantidade de respondentes filtrados para análise

Questão	Requisitos atendidos	Total da amostra
Como você avalia sua experiência com automação de testes de interface:	Muito alta, trabalho diretamente e apoio outros profissionais. Alta, trabalho diretamente. Média, já atuei em equipes que implementaram, mas não trabalhei desenvolvendo diretamente.	184
Você possui conhecimento sobre metodologias ágeis?	Sim	
Você possui conhecimento sobre o *framework* Scrum?	Sim	
Emprega metodologia ágil?	Sim	

Fonte: elaborada pela autora.

Com a amostra adequada para a análise, foi possível verificar, a partir das questões diretamente relacionadas às características do artefato, a aprovação quanto à sua viabilidade de uso pelo público de interesse, os resultados são exibidos na Figura 19.

Figura 19 - Resultados da pesquisa de viabilidade do artefato



Fonte: elaborada pela autora.

A última pergunta do questionário de cunho livre ao participante, teve dentro da amostra filtrada 28 contribuições. E em declarações como: “Defendo muito que os times precisam iniciar o trabalho da *Sprint* pensando nos testes funcionais automatizados.”, “O

processo de ATS é contínuo, em todos os eventos deve ser discutido, planejado e escrito.” e “O processo de desenvolvimento de testes é, muitas das vezes, algo que pode e deve ser compartilhado com o time, de forma que todos possam colaborar com o conhecimento que possuem. Ao meu ver, isso diminui os conflitos que costumam existir entre time de desenvolvimento e QA. O time precisa ser coeso. Precisa entender que todas as funções são importantes e parte de um ecossistema”, possibilitaram evidenciar a sincronia do entendimento do participante quanto à proposta de pesquisa.

Embora o instrumento de pesquisa não tenha elencado questões quanto às práticas DevOps adotada no modelo pela integração contínua e a execução em *pipeline*, por terem sido estas validadas junto aos especialistas, houve a expressividade a respeito na última questão, onde o participante declara que “Além da importância da automação em si, considero ainda mais necessário que a ATS de UI esteja rodando em um plano de integração contínua e entrega contínua, pois de nada adiantaria todo esforço se a mesma não for executada nos momentos mais críticos.”, percepção, esta, em total coerência ao artefato e as práticas para um ambiente ágil.

Ainda que em sua maioria as contribuições tenham sido a favor do modelo, houve observações sobre a etapa de desenvolvimento da atividade de desenvolvimento da ATS de UI, como “…desenvolvedores pode SIM automatizar também, mas creio que o tempo deles é melhor gasto criando testes unitários, sem contar que precisam ser desenvolvedores de mente aberta em relação a testes de interface (tanto no sentido de achar bugs não esperados, quanto no sentido de saber que o que é importante pro cliente é o comportamento geral da aplicação e não as linhas de código que ele está criando);”. Ao correlacionar esta declaração às observações feitas pelo especialista 3, nota-se que há uma preocupação sobre a questão cultural das habilidades do desenvolvedor quanto à qualidade do produto. Porém entende-se que, além da adoção de práticas como o *code review*, assume-se como consequência o amadurecimento da equipe a cada ciclo de execução do processo, já que todos estarão se relacionando diretamente com o processo de ATS de UI.

Além da percepção positiva através da análise dos dados da pesquisa de campo sobre ao uso de linguagem de exemplos para a documentação do time, houve ainda a contribuição por parte de um participante enfatizando que “O BDD é voltado e escrito pelo time ágil, principalmente nas figuras do testador, desenvolvedor e PO, o BDD escrito apenas pelo time de QA “empobrece” a técnica e pode gerar desentendimentos e retrabalhos”, o que alinha com a ideia do artefato em ter uma única linguagem na equipe tratada por todos desde os requisitos aos cenários de teste.

7.5 CONSIDERAÇÕES SOBRE A VALIDAÇÃO DO ARTEFATO

Para verificar a validade programática, assim como, a viabilidade de contribuição a um ambiente externo da resposta de pesquisa deste trabalho, utilizou-se da etapa de validação, sendo esta fundamental ao método adotado e ao atendimento do rigor de pesquisa. A partir dos meios de validação utilizados, houve a contribuição para o ganho de conhecimento através da troca de experiência com os especialistas, proporcionada com as entrevistas, assim como, o enriquecimento com a troca de informações pela estratégia adotada na divulgação do questionário ao contatar, na maioria das vezes, diretamente os profissionais da área via bate-papo (*chat*), gerando, inclusive, em algumas situações debates sobre o assunto e indicações de materiais para o estudo.

Ainda durante esta abordagem, houve pedidos para a divulgação do trabalho na rede, bem como, a análise quanto aos resultados obtidos pela pesquisa. Alguns participantes utilizaram-se da questão livre para realizar esta solicitação, “Tenho interesse em saber o resultado da pesquisa depois! ” e também traçar considerações à mesma, tais como, “Muito útil”, “Excelentes perguntas.” e “Ótimo trabalho. Por favor, espalhe a mensagem de que não existe desenvolvimento ágil sem automação de testes”.

Embora a análise do questionário tenha sido realizada sob a redução de uma amostra mais próxima ao contexto do trabalho, cabe realçar quanto à percepção sobre a vontade de contribuição para com o assunto pelo público participante, esta, dada tanto pelos totais dos respondentes na conscientização da problemática, quanto aos totais na validação.

8. CONCLUSÃO

As empresas têm enfrentado muitas dificuldades com a implementação da automação de testes de UI e, a busca por melhores alternativas vêm chamando a atenção da comunidade de desenvolvimento de software, assim como, a comunidade acadêmica. Este trabalho propôs um modelo de processo para a condução da ATS de UI voltado ao ambiente ágil de desenvolvimento de software, com o propósito de atender a algumas das dificuldades mais evidentes e elencadas no decorrer desta pesquisa, além de trazer maior visibilidade do processo de ATS de UI dentro de todo o ciclo de desenvolvimento de software consolidado ao Scrum.

No ambiente ágil de desenvolvimento software a equipe deve ser capaz de contribuir com a melhoria contínua das entregas e responder às mudanças de forma rápida. Em vista disto, o modelo proposto abordou o processo de ATS de UI em todas as cerimônias do Scrum visando a conscientização do mesmo para com todo o time. Ainda, indicou filtrar as atividades de ATS de UI classificadas pelo PO em conjunto com o cliente/*stakeholders*, objetivando maior valor na entrega.

E partir disto, propôs integrar ao ciclo de desenvolvimento estas atividades orientando e elencando boas práticas para com o projeto de ATS de UI espelhado a um projeto de software, bem como, atribuições do DevOps quanto à integração contínua na *pipeline*. Com isso, facilitar a contribuição dos desenvolvedores com às atividades e, consequentemente, aumentando a cobertura de testes automatizados a cada ciclo de desenvolvimento.

A proposta de integração constante de todos da equipe com o processo de ATS de UI, exposta pelo artefato, não só auxilia a provir com as atividades na etapa de desenvolvimento, mas também com a evolução do mesmo por parte de cada um. Além disso, conforme proposto, abordar o processo de ATS de UI na cerimônia de *Sprint Review* com o envolvimento do cliente/*stakeholder*, contribui na visibilidade deste no que se refere à entrega e todo o seu tratamento no ciclo de desenvolvimento. E, consequentemente, ajuda a atribuir aos envolvidos no desenvolvimento o senso de responsabilidade pela entrega.

Dito isto, referenciando o problema de pesquisa conscientizado no capítulo 3 deste trabalho, em conjunto com a etapa de avaliação e análise apresentadas no capítulo 7 do mesmo, possibilitaram afirmar que a presente pesquisa atingiu seu objetivo ao contribuir com um artefato válido que auxilie em um contexto real seu público de interesse.

Em natureza pessoal, esta pesquisa viabilizou diversas oportunidades de absorver conhecimento sobre o assunto em suas diferentes etapas aplicadas. A partir da etapa de

conscientização da problemática, para construção do artefato de pesquisa, foram necessárias leituras e busca por conteúdos que levassem à compreensão da mesma com uma visão diretamente relacionada à realidade.

Seguida da aplicação do instrumento diante da forma utilizada na condução do mesmo, foi gerada uma massa de dados que, quando tratada, implicou em informações para a construção dos requisitos de pesquisa. Partindo então ao seu referencial teórico e leitura de trabalhos correlatos, os quais trouxeram a percepção de contribuição científica ao trabalho e provendo o conhecimento de apoio para elaboração satisfatória da resposta de solução.

Com respeito à etapa de validação por meio de entrevistas, esta oportunizou o contato com diferentes especialistas da área que geraram, além de observações e críticas ao modelo, o incremento de conhecimento a cunho profissional. Vale ressaltar que esta etapa não foi abordada na conscientização da problemática, mas ao fim desta pesquisa percebeu-se que seria de grande valia tê-la agregada em seu início, justamente pelas percepções mais sólidas que os especialistas trazem à pesquisa com uma visão madura do ambiente real. Contudo, a ausência desta não impactou nos resultados da pesquisa, mas atribui a futuros trabalhos esta conscientização.

Em relação à definição do estudo de viabilidade do artefato, cabe ressaltar o resultado satisfatório à sua adesão, como exibido na Figura 19, evidenciando também a coerência quanto à objetividade do método DSR para com as relações em propor um artefato de caráter científico voltado a um contexto prático quanto à sua percepção real. Destaca-se ainda ao método utilizado, a contribuição da abordagem científica como aporte à pesquisa da área, pois promove ao pesquisador a nítida visão de contribuição além do caráter científico.

Por fim, é possível propor futuras linhas de pesquisas a serem exploradas a partir do trabalho apresentado, tais como, a adequação do modelo para equipes que utilizam de outros *frameworks* de gerenciamento de processos, a aplicação do modelo para um estudo de caso e a proposição e aplicação de métricas de qualidade diretamente relacionadas à ATS de UI diante do proposto artefato.

Com base no que foi apresentado, conclui-se que o presente trabalho consolidou seu objetivo geral alcançando seus objetivos específicos, dispondo também da contribuição do conhecimento científico através do processo de concepção da proposta de solução. Ainda, a contribuição ao contexto real quanto à utilização do artefato gerado pela mesma.

REFERÊNCIAS BIBLIOGRÁFICAS

ANDRADE, Mayb. **Qualidade de software**. 2015, 1 ed. Rio de Janeiro, RJ: SESES, 2015.

ANICHE, Maurício. **Testes automatizados de software:** um guia prático. 2017. São Paulo, SP: Casa do código, 2017. ISBN: 9788555190285.

BASTOS, Anderson S. et al. **Base de conhecimento em teste de software**. 2.ed. São Paulo, SP: Martins, 2007.

BECK, Kent. **TDD desenvolvimento guiado por testes**. Porto Alegre, RS: Bookman, 2010.

BECK et al. **Manifesto Ágil**. 2001.Disponível em: <https://www.manifestoagil.com.br/> Acesso em: 28 maio de 2019.

CHELIMSKY, David et al. **The RSpec Book:** Behaviour Driven Development with RSpec, Cucumber, and Friends (Facets of Ruby). 2010, vP 2.1, EUA: September, 2012.

COHN, Mike. **Desenvolvimento de software com Scrum:** Aplicando métodos ágeis com sucesso. Porto Alegre, RS: Bookman, 2011.

COLLINS, Eliane Figueiredo. **Modelo de automação de testes funcionais para desenvolvimento ágil de software**. 2013. 100 f. Dissertação (Pós-Graduação em Engenharia Elétrica) – Universidade do Amazonas, Manaus, Brasil, 2013.

COLLINS, Eliane Figueiredo et al. **An Industrial Experience on the Application of Distributed Testing in an Agile Software Development Environment**. In: Global Software Engineering (ICGSE), 2012 IEEE 7th International Conference, 2012.

COLLINS, Eliane Figueiredo; LUCENA JR., Vicente Ferreira de. **Software test automation practices in agile development environment: an industry experience report**. In: Proceedings of the 7th International Workshop on Automation of Software Test (AST), 2012.

COLLINS, Eliane Figueiredo; DIAS-NETO, Arilo; LUCENA JR., Vicente Ferreira de. **Strategies for Agile Software Testing Automation: An Industrial Experience**. In: IEEE 36th Annual Computer Software and Applications Conference Workshops, IEEE, 2012.

COLLINS, Eliane Figueiredo; LOBÃO, Luana M. de A**. Experiência em Automação do Processo de Testes em Ambiente ágil com Scrum e Ferramenta OpenSource.** In: SIMPÓSIO BRASILEIRO DE QUALIDADE DE SOFTWARE, 9., 2010.

COSTA, Mozart Guerra. **Estratégia de Automação em Testes:** Requisitos, Arquitetura e Acompanhamento de sua Implementação. 2004. 102 f. Trabalho de Conclusão de Curso (Mestrado) – Engenharia de Computação, Instituto de Computação Universidade Estadual de Campinas (UNICAMP), Campinas, SP, 2004.

CRISPIN, Lisa; GREGORY, Janet. **Agile Testing:** A Practical Guide for Testers and Agile Teams. Boston, MA: Pearson Education Inc, 2009.

CRUZ, Fábio. Scrum e Agile em projetos.2018, 1ed. Rio de Janeiro, RJ: BRASPORT, 2018.

DEBOIS, Patrick et al. **The DevOps Handbook:** How to Create World-Class Agility, Reliability, and Security in Technology Organizations. 1 ed. EUA: IT REVOLUTION, Press, 2016.

DELAMARO, Márcio Eduardo; MALDONADO, José Carlos; JINO, Mario. **Introdução ao teste de software**. 2 ed. São Paulo, SP: ELSEVIER, 2016.

DRESCH, Aline; LACERDA, Daniel Pacheco; ANTUNES, José A. V. **Design science research:** método de pesquisa para avanço da ciência e tecnologia. Porto Alegre, RS: Bookman, 2015.

GARTNER, Markus. **ATDD by Example A Practical Guide to Acceptance Test-Driven Development**. Boston, MA: Pearson Education Inc, 2013.

GOMES, André Faria. **Agile:** Desenvolvimento de software com entregas frequentes e foco no valor de negócio. 2. ed. Novo Hamburgo, RS: Feevale, 2013.

HUMBLE, Jez; FARLEY David. **Entrega contínua:** Como entregar software de forma rápida. Porto Alegre: Bookman, 2014.

MARTIN, Robert C.; MARTIN, Micah. **Princípios, padrões e práticas ágeis em C#**. 2011. Porto Alegre, RS: Bookman, 2011.

MARTINS, Marcos Danilo Chiodi. **Testes de Software**. 1 ed. Rio de Janeiro, RJ: SESES, 2016.

MASSARI, Vitor L. **Gerenciamento ágil de projetos**. 2018, 2ed. Rio de Janeiro, RJ: BRASPORT, 2018. ISBN: 9788574528939.

MOLINARI, Leonardo. **Inovação e Automação de Testes de Software**. 1 ed. São Paulo, SP: Érica, 2010.

MULCHANDANI, Bharat. Software Test Conference. **Why test automation fails**. 2013. Disponível em: < http://conference.qaiglobalservices.com/stc2013/PDFs/Bharat_K_Mulchandani.pdf >. Acesso em: 30 março de 2019.

MUNIZ, Antônio et al. **Jornada DevOps: unindo cultura ágil, Lean e tecnologia para entrega de software com qualidade**. 1 ed. São Paulo, SP: Brasport, 2019.

MYERS, Glenford J.; BADGETT, Tom; SANDLER Corey. **The art of software testing**. 3 ed. Hoboken, NJ: Wiley, 2012.

NAIK, Kshirasagar; TRIPATHY, Priyadarshi. **Software Testing and Quality Assurance:** Theory and Practices. Hoboken, Nova Jersey: Wiley, 2008.

PÁDUA FILHO, Wilson de Paula. **Engenharia de Software:** fundamentos, métodos e padrões. 2010, 3d. Rio de Janeiro: LTC, 2009. ISBN: 9788521616504

PERES, Hugo. **Automatizando Testes de Software Com Selenium**. 1. ed. Porto Alegre, RS: SIMPLÍSSIMO, 2016.

PIMENTEL, Mariano. **Design Science Research e Pesquisas com os Cotidianos Escolares para fazer pensar as pesquisas em Informática na Educação**. In: VI Congresso Brasileiro de Informática na Educação. 6. 2017, Rio de Janeiro. Anais... Rio de Janeiro: UNIRIO, 2017.

PRESSMAN, Roger S.; MAXIM, Bruce R. **Engenharia de software:** uma abordagem profissional. 8. ed. Porto Alegre, RS: AMGH, 2016.

PRODANOV, Cleber Cristiano; FREITAS, Ernani Cesar de. **Metodologia do trabalho científico:** métodos e técnicas da pesquisa e do trabalho acadêmico. 2. ed. Novo Hamburgo, RS: Feevale, 2013.

PRIKLADNICKI, Rafael; MILANI, Fabiano; WILLI, Renato. **Métodos ágeis para desenv olvimento de software**. 2014.Porto Alegre, RS: Livroman, 2014.

QASYMPHONY; TECHWELL Survey Corp. The Evolution of Test Automation. 2018 Disponível em: <https://www.qasymphony.com/landing-pages/report-the-evolution-of-test-automation/ > Acesso em: 4 de junho de 2019.

RAFI, Dudekula Mohammad et al. **Benefits and limitations of automated software testing:** Systematic literature review and practitioner survey. 2012 7th International Workshop on Automation of Software Test (ATS), pp. 36 - 42. DOI = http://dx.doi.org/10. 1109/IWATS.2012.6228988.

RIOS, Emerson. **Análise de riscos em projetos de teste de software**. Rio de Janeiro, RJ: ALTA BOOKS, 2005.

RODRIGUES, Anderson Clayton Barreto. **Um arcabouço conceitual para diagnóstico organizacional a respeito da utilização da automação de teste de software**. 2018. 117 f. Tese (Doutorado em Informática) - Universidade Federal do Amazonas. Manaus, 2018.

SABBAGH, Rafael. **Scrum: Gestão ágil para projetos de sucesso**. 2014, 465 p. Casa do código. ISBN: 9788566250954.

SATO, Danilo. **DevOps na prática:** entrega de software confiável e automatizada. 2014, 320 p. Casa do código: 2014. ISBN: 9788566250664.

SOMMERVILLE, Ian. **Engenharia de Software**. 9. ed. São Paulo, SP: PERSON, 2007.

SBROCCO, José Henrique T.C.; MACEDO, Paulo Cesar de. **Metodologias ágeis:** engenharia de software sob medida. 2010, 1 ed. São Paulo: Erica, 2012.

STEPHENS, Matt; ROSENBERG, Doug. **Design Driven Testing:** Test Smarter, Not Harder. 2010. Nova York, NY: Springer, 2010.

VERSION ONE. **The 13th Annual State of Agile Report**. 2019. Disponível em: < https:// www.stateofagile.com/#ufh-i-521251909-13th-annual-state-of-agile-report/473508>. Acesso em: 4 de junho de 2019.

VERONA, Joakim; DUFFY, Michael; SWARTOUT, Paul. **Learning DevOps**: Continuous ly Deliver Better Software. 1ed. 2016. Birmingham, UK: Packt Publishing.

VINCENZI, Auri Marcelo Rizzo et al. **Automatização de Teste de Software com Ferramentas de Software Livre**. 2018. Rio de Janeiro, RJ: Elsevier, 2018.

WALMYR, Filho. **Protractor:** Lições sobre testes end-to-end automatizados. 2016, 224 p. Casa do código, 2016 ISBN: 9788555192296.

WILDT; Daniel et al. **eXtreme Programming:** Práticas para o dia a dia no desenvolvimento ágil de software. 2015, 148 p. Casa do Código: 2015. ISBN: 9788555191060

YE, Wayne. **Instant Cucumber BDD How-to**. 1. ed. Birmingham, UK: PACKT, 2013.

APÊNDICE A – TERMO DE CONSENTIMENTO

Automatização de testes em ambiente ágil

Este questionário faz parte de um trabalho de conclusão de curso, na área de Sistema de Informação, desenvolvido na Universidade Feevale (Novo Hamburgo/RS) e tem por objetivo identificar aspectos relevantes quanto às dificuldades no processo de automação de testes de software em um contexto ágil.

Desde já agradeço sua colaboração!

Termo de Consentimento Livre e Esclarecido

O trabalho de conclusão de curso está sendo desenvolvido pela acadêmica Maristela Oliveira de Paula, do curso de Sistemas de Informação da Universidade Feevale.

Sua participação nesta pesquisa é voluntária e consistirá em responder este questionário online. Não há riscos ou despesas para os participantes. Os dados obtidos a partir desta pesquisa não serão usados para outros fins além do já mencionado.

Ao avançar neste questionário você estará aceitando este Termo de Consentimento Livre e Esclarecido.

PRÓXIMA

APÊNDICE B – PERGUNTAS

Responda ao questionário considerando sua última experiência com automatização de testes.

A equipe emprega metodologias ágeis? *

- ◯ Sim
- ◯ Não

VOLTAR PRÓXIMA

A equipe utiliza algum framework ágil ou alguma técnica de desenvolvimento de software? *

- ☐ Desenvolvimento Orientado a Testes de Aceitação (ATDD)
- ☐ Desenvolvimento Orientado por Testes (TDD)
- ☐ Desenvolvimento Guiado por Comportamento (BDD)
- ☐ Kanban
- ☐ SCRUM
- ☐ XP (eXtreme Programming)
- ☐ Nenhuma
- ☐ Outro: _____

Sobre os testes automatizados: *

- ◯ Houve implementação de automatização de testes.
- ◯ Houve a tentativa de implementação, mas falhou.
- ◯ Não houve implementação de automatização de testes, mas existiu o interesse.
- ◯ Nunca foi utilizada, nem existiu o interesse.

Na percepção da equipe, sobre o grau de relevância dos testes quanto à entrega do produto: *

- ○ 0 - Não tem relevância na entrega do produto: pode ser entregue sem teste.
- ○ 1 - Parcialmente relevante na entrega do produto: entregue mesmo sem todos os testes, porém são obrigatórios testes que garantam a funcionalidade mínima do produto.
- ○ 2 - Rigorosamente relevante na entrega do produto: não é liberado até que todos os testes sejam realizados com sucesso.

Qual seu papel na equipe? *

- ○ Analista de Automação de Testes de Software
- ○ Analista de Qualidade de Software
- ○ Analista de Negócios
- ○ Analista de Sistemas
- ○ Analista de Testes de Software
- ○ Desenvolvedor de Software
- ○ Desenvolvedor de Testes Automatizados
- ○ Engenheiro de Software
- ○ Engenheiro de Qualidade de Software
- ○ Gerente de Projetos
- ○ Testador de Software
- ○ Outro: _____

A empresa é considerada como: *

- ○ Microempresa (Até 9 funcionários)
- ○ Empresa de Pequeno porte (De 10 a 49 funcionários)
- ○ Empresa de Médio porte (De 50 a 99 funcionários)
- ○ Empresa de Grande porte (Acima de 99 funcionários)

VOLTAR PRÓXIMA

Quais as motivações para implementação dos testes automatizados? *

- [ ] Suprir a ausência de documentação.
- [ ] Necessidade de adequação ao modelo de entrega contínua.
- [ ] Pouco tempo para execução dos testes regressivos funcionais.
- [ ] Outro: _____

Quais testes foram automatizados ? *

- [ ] Testes Unitários
- [ ] Testes de Serviços
- [ ] Testes de interface do usuário (funcional)
- [ ] Outro: _____

Houve dificuldades no processo de implementação de testes automatizados? *

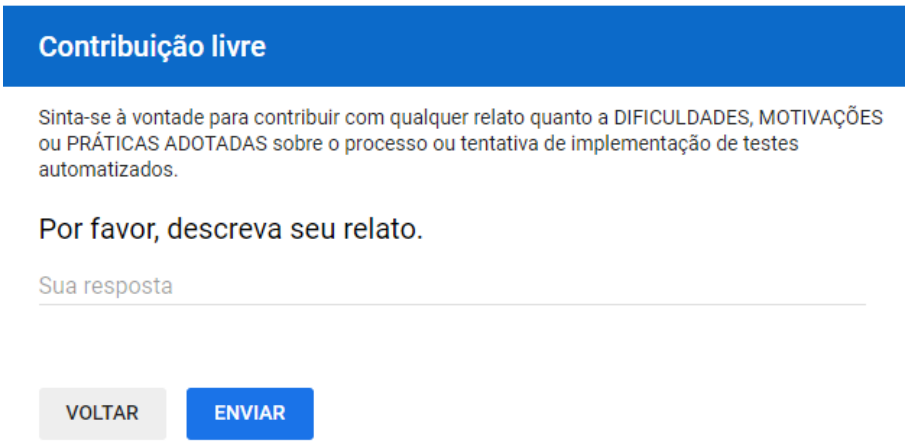
- ( ) Sim
- ( ) Não

VOLTAR PRÓXIMA

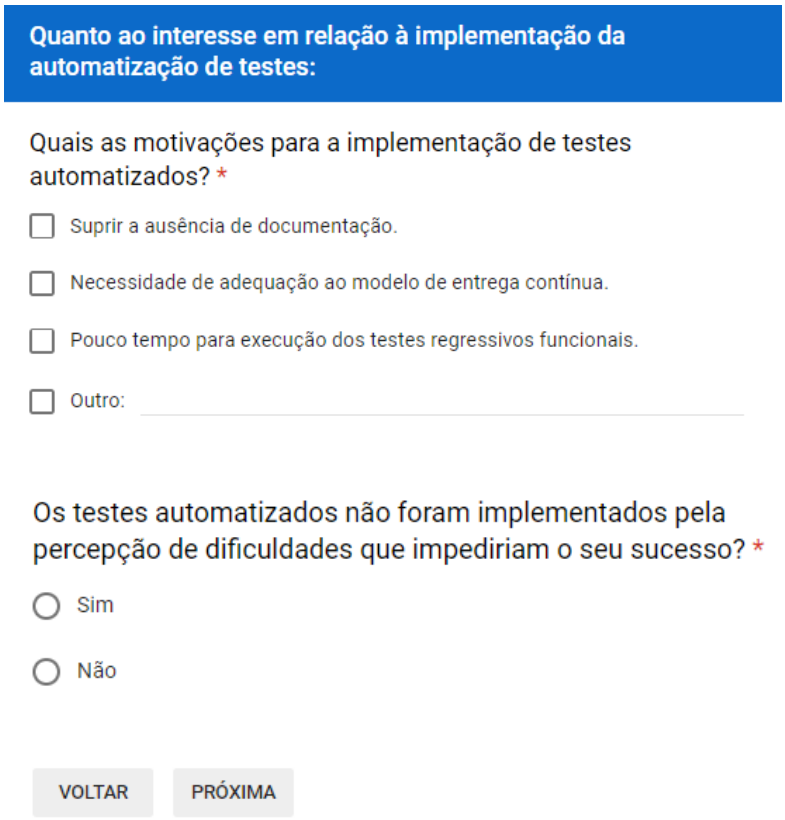
Indique quais as dificuldades: *

- [ ] Ausência de documentação
- [ ] Ausência de uma equipe dedicada a automação de testes
- [ ] Ausência de uma equipe treinada a automação de testes
- [ ] Diversas alterações do produto (difícil manutenção do projeto)
- [ ] Falta de viabilidade técnica (tecnologia não passível de automatização)
- [ ] Falta de viabilidade econômica (benefícios podem não superar o investimento)
- [ ] Pouco tempo para desenvolver a automação
- [ ] Pouca/nenhuma utilização de métricas de qualidade (usada para medir o sucesso do resultado esperado)
- [ ] Processo de testes não definido
- [ ] Outro: _____

VOLTAR PRÓXIMA



APÊNDICE C – FLUXO NÃO IMPLEMENTOU ATS, MAS EXISTIU INTERESSE



Indique quais as dificuldades: *

- [ ] Ausência de documentação
- [ ] Ausência de uma equipe dedicada a automação de testes
- [ ] Ausência de uma equipe treinada a automação de testes
- [ ] Diversas alterações do produto (difícil manutenção do projeto)
- [ ] Falta de viabilidade técnica (tecnologia não passível de automatização)
- [ ] Falta de viabilidade econômica (benefícios podem não superar o investimento)
- [ ] Pouco tempo para desenvolver a automação
- [ ] Pouca/nenhuma utilização de métricas de qualidade (usada para medir o sucesso do resultado esperado)
- [ ] Processo de testes não definido
- [ ] Outro: _____

VOLTAR PRÓXIMA

Contribuição livre

Sinta-se à vontade para contribuir com qualquer relato quanto a DIFICULDADES, MOTIVAÇÕES ou PRÁTICAS ADOTADAS sobre o processo ou tentativa de implementação de testes automatizados.

Por favor, descreva seu relato.

Sua resposta

VOLTAR ENVIAR

APÊNDICE D - QUESTIONÁRIO DE VERIFICAÇÃO DE ADESÃO DO ARTEFATO

Automatização de testes em ambiente ágil

Este questionário faz parte de um trabalho de conclusão de curso, na área de Sistema de Informação, desenvolvido na Universidade Feevale (Novo Hamburgo/RS) e tem como objetivo verificar a viabilidade de uso de um modelo de processo para implementação de automatização de teste para interface gráfica em ambiente ágil.

Desde já agradeço sua colaboração!

*Obrigatório

Termo de Consentimento Livre e Esclarecido

O trabalho de conclusão de curso está sendo desenvolvido pela acadêmica Maristela Oliveira de Paula, do curso de Sistemas de Informação da Universidade Feevale.

Sua participação nesta pesquisa é voluntária e consistirá em responder este questionário online. Não há riscos ou despesas para os participantes. Os dados obtidos a partir desta pesquisa não serão usados para outros fins além do já mencionado.

Ao avançar neste questionário você estará aceitando este Termo de Consentimento Livre e Esclarecido.

Perfil

1. **Trabalha em qual estado?** *

Marcar apenas uma oval.

- ◯ Acre (AC)
- ◯ Alagoas (AL)
- ◯ Amapá (AP)
- ◯ Amazonas (AM)
- ◯ Bahia (BA)
- ◯ Ceará (CE)
- ◯ Distrito Federal (DF)
- ◯ Espírito Santo (ES)
- ◯ Goiás (GO)
- ◯ Maranhão (MA)
- ◯ Mato Grosso (MT)
- ◯ Mato Grosso do Sul (MS)
- ◯ Minas Gerais (MG)
- ◯ Pará (PA)
- ◯ Paraíba (PB)
- ◯ Paraná (PR)
- ◯ Pernambuco (PE)
- ◯ Piauí (PI)
- ◯ Rio de Janeiro (RJ)
- ◯ Rio Grande do Norte (RN)
- ◯ Rio Grande do Sul (RS)
- ◯ Rondônia (RO)
- ◯ Roraima (RR)
- ◯ Santa Catarina (SC)
- ◯ São Paulo (SP)
- ◯ Sergipe (SE)
- ◯ Tocantins (TO)

2. **Trabalha em uma equipe de desenvolvimento de software?** *
Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

3. **Como você avalia sua experiência com automação de testes de interface:** *
Marcar apenas uma oval.

- ◯ Muito alta, trabalho diretamente e apoio outros profissionais.
- ◯ Alta, trabalho diretamente.
- ◯ Média, já atuei em equipes que implementaram, mas não trabalhei desenvolvendo diretamente.
- ◯ Pouca, só por leitura ou de ouvir falar.
- ◯ Nenhuma.

4. **Você possui conhecimento sobre metodologias ágeis?** *
Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

5. **Você possui conhecimento sobre o framework SCRUM?** *
Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

Responda ao questionário considerando sua última experiência com automatização de testes de interface, em relação à equipe:

6. **Qual seu papel?** *
Marcar apenas uma oval.

- ◯ QA: testador / analista de testes / desenvolvedor de testes automatizados.
- ◯ Desenvolvedor de software.
- ◯ Scrum Master.
- ◯ Product Owner.
- ◯ Outro: _____

7. **Emprega metodologia ágil?** *
Marcar apenas uma oval.

- ◯ Sim.
- ◯ Não.

8. **Utiliza algum framework ágil para gerência de processos?** *
Marcar apenas uma oval.

- ◯ SCRUM.
- ◯ XP (eXtreming Programing).
- ◯ CRYSTAL.
- ◯ Outro: _____

9. **Realiza testes de interface automatizados?** *
Marcar apenas uma oval.

- ◯ Sim.
- ◯ Não.

Na sua opinião sobre o processo de Automatização de Testes de Software (ATS) para interface gráfica:

10. **Ter os cenários de testes a serem automatizados já mapeados pelo PO/Cliente impacta no processo de ATS?** *
Marcar apenas uma oval.

- ◯ Positivamente.
- ◯ Negativamente.
- ◯ Não impacta.

11. **Ter os cenários de testes a serem automatizados já classificados quanto à sua prioridade pelo PO/Cliente impacta no processo de ATS?** *
Marcar apenas uma oval.

- ◯ Positivamente.
- ◯ Negativamente.
- ◯ Não impacta.

12. **Ter a documentação de requisitos, escrita em especificações executáveis através de uma linguagem de domínio do time exemplo, BDD, impacta no processo de ATS?** *
Marcar apenas uma oval.

- ◯ Positivamente.
- ◯ Negativamente.
- ◯ Não impacta.

13. **Utilizar da Sprint Planning para planejar as atividades do processo de automatização impacta no processo de ATS?** *
Marcar apenas uma oval.

- ◯ Positivamente.
- ◯ Negativamente.
- ◯ Não impacta.

14. **Concorda que um projeto de automatização de testes, quando criado em uma linguagem de desenvolvimento de domínio do time, facilita o desenvolvimento das atividades de ATS e a evolução do mesmo?** *
Marcar apenas uma oval.

- ◯ Sim.
- ◯ Não.
- ◯ Não sei opinar.

15. **No ciclo de Desenvolvimento, com o projeto de ATS disponível para todos da equipe, concorda que é possível um desenvolvedor atuar na atividade de desenvolvimento de testes automatizados?** *
Marcar apenas uma oval.

- ◯ Sim.
- ◯ Não.
- ◯ Não sei opinar.

16. **O responsável de testes da equipe, ao escrever os cenários de testes diretamente no projeto de ATS através de um framework para BDD, por exemplo, impacta o processo de ATS?** *
Marcar apenas uma oval.

- ◯ Positivamente.
- ◯ Negativamente.
- ◯ Não impacta.

17. **Você acredita que a entrega consiste tanto de características/features de produto quanto da automatização ?** *
Marcar apenas uma oval.

- ◯ Sim.
- ◯ Não
- ◯ Não sei opinar.

Obrigada pela contribuição!

18. **Sinta-se à vontade, caso queira deixar alguma observação sobre sua percepção em relação às características do processo de ATS abordadas no questionário.**

_____

_____