CENTRO UNIVERSITÁRIO DE ANÁPOLIS – UNIEVANGÉLICA
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

BRENO BORGES FRANCA
JEAN SANTOS FLORES

MELHORIA DO PROCESSO DE TESTE DE *SOFTWARE* BASEADO NO MPT.BR

ANÁPOLIS
2019

BRENO BORGES FRANCA
JEAN SANTOS FLORES

MELHORIA DO PROCESSO DE TESTE DE *SOFTWARE* BASEADO NO MPT.BR

Trabalho de Conclusão de Curso II apresentado como requisito parcial para a conclusão da disciplina de Trabalho de Conclusão de Curso II do curso de Bacharelado em Engenharia de Computação do Centro Universitário de Anápolis – UniEVANGÉLICA.

Orientadora: Prof.ª Ms.Walquíria Fernandes Marins.

ANÁPOLIS
2019

BRENO BORGES FRANCA
JEAN SANTOS FLORES

MELHORIA DO PROCESSO DE TESTE DE SOFTWARE BASEADO NO MPT.BR

Trabalho de Conclusão de Curso II apresentado como requisito parcial para a obtenção de grau do curso de Bacharelado em Engenharia de Computação do Centro Universitário de Anápolis – UniEVANGÉLICA.

Aprovado(a) pela banca examinadora em 11 de junho de 2019, composta por:

Walquiria F. Marins

Walquiria Fernandes Marins
Presidente da Banca

Audir da Costa Oliveira Filho
Prof(a). Convidado(a)

Luciana Nishi
Prof(a). Convidado(a)

Agradecimentos

Agradecemos a Deus por nos ter dado saúde e força para superar as dificuldades, agradecemos também está universidade e seu corpo docente, juntamente com a direção, que nos deu a oportunidade de vislumbrar um horizonte superior para nossa vida.

Muita gratidão a nossa orientadora Walquíria Fernandes Maríns, pelo suporte no pouco tempo que lhe coube, pela sus correções e incentivos ao longo do ano, ao nossos pais pelo incentivo e apoio incondicional e a todos que fizeram parte da nossa formação, nosso obrigado.

“Ninguém ignora tudo. Ninguém sabe tudo. Todos nós sabemos alguma coisa. Todos nós ignoramos alguma coisa. Por isso aprendemos sempre”.

Paulo Freire

Resumo

Com a constante renovação tecnológica, os sistemas de *software* passaram a fazer parte das tarefas rotineiras da atual sociedade. Nesse sentido, não atender às expectativas do usuário pode ocasionar uma série de problemas, correndo desde o âmbito financeiro até a integridade das informações dos usuários. Dessa forma, a execução de testes se tornou uma atividade crucial para garantir a qualidade e a eficiência do produto final. Entretanto, os processos de teste utilizados nem sempre são os mais adequados ao contexto do projeto ou realizam as atividades de forma ineficiente. Logo, o objetivo deste trabalho é avaliar um processo de teste de *software* tendo como parâmetro o primeiro nível de um modelo de maturidade MPT.BR, para identificar pontos de melhoria bem como propor uma nova abordagem de processo mais completa, eficiente e eficaz.

Palavras-chave: Melhoria do Processo de Teste de *Software*, Engenharia de *Software*, Teste de *Software*.

Abstract

With constant technological renewal, software systems have become part of the routine tasks of today's society. In this sense, not meeting the user's expectations can cause a series of problems, ranging from the financial scope to the integrity of users. In this way, testing has become a crucial activity to ensure the quality and efficiency of the final product. However, the test processes used are not always the most appropriate to the project context or perform the testing activities inefficiently. Therefore, the objective of this work is to evaluate a software testing process having as parameter the first level of a MPT.BR maturity model to identify improvement points as well as to propose a new, more complete, efficient and effective process approach.

Keywords: Improvement of the Software Testing Process, Software Engineering, Software Testing.

Lista de Ilustrações

ABNT	Associação Brasileira de Normas Técnicas
BID	Banco Interamericano de Desenvolvimento
CMMI	*Capability Maturity Model Integration*
FINEP	Financiadora de Estudos e Projetos
FUMIN	Fundo Multilateral de Investimentos
GPT	Gerência de Projetos de Teste de *Software*
IEC	*International Electrotechnical Commission*
IEEE	*Institute of Electricaland Electronic Engineers*
ISO	*International Organization for Standardization*
MCT	Ministério da Ciência e Tecnologia
MPS.BR	Melhoria de Processo do *Software* Brasileiro
PET	Projeto e Execução de Teste
PG	Práticas Genéricas
SEBRAE	Serviço Brasileiro de Apoio às Micro e Pequenas Empresas
SINFIC	Sistemas de Informação Industriais e Consultoria
SOFTEX	Associação para Promoção da Excelência do *Software* Brasileiro
TMMI	*Test Maturity Model Integrated*

Lista de Ilustrações

Lista de Tabelas

Sumário

1. INTRODUÇÃO

Com a constante renovação tecnológica, os sistemas de *software* passaram a fazer parte das tarefas rotineiras da atual sociedade, onde, as fábricas de *software* tornaram-se centros necessários para a evolução constante dos programas dada a grande velocidade que as mudanças se dão e se fazem cada vez mais necessárias (Souza, 2017).

Segundo Pressman (2009), engenharia de *software* é o que emprega os princípios de engenharia para obter *software* de maneira econômica, confiável e que funcione em máquinas reais, no qual deve estar fundamentada em um comprometimento organizacional com a qualidade, pois a gestão da qualidade promove uma cultura de aperfeiçoamento continuo nos processos, e esta cultura é o que importa, dado que a engenharia de *software* é focada na qualidade desse produto.

Ainda de acordo com Pressman (2016) testar *software*, é uma das atividades cruciais para garantir a qualidade do projeto, porém para testar *software* envolve um processo organizacional de atividades, que se não forem realizadas de maneira correta podem ocasionar impactos e custos desnecessários no desenvolvimento dos projetos.

A qualidade de um *software* para Koscianski e Soares (2007) tem o objetivo de satisfazer as necessidades do cliente, no qual possui aspectos envolvidos, como: orçamento para realização do projeto; prevenção e eliminação de defeitos; ferramentas; técnicas organizacionais; validação de requisitos entre outros.

Segundo Pressman (2009) o processo de desenvolvimento de *software* engloba uma grande quantidade de atividades, no qual a chance de ocorrer falhas humanas é enorme, logo a atividade de teste de *software* torna-se uma atividade crítica para garantia de qualidade de *software*.

Para verificar e validar requisitos, prevenir ou eliminar defeitos nos produtos, utiliza-se de métodos de testes. Segundo Hetzel (1987), testes são atividades que avaliam características ou recurso de um sistema, sendo um indicador da qualidade de *software*. Já Pressman (2009), enfatiza que a atividade de teste tem sido uma das atividades com maior custo no desenvolvimento de *software*, por este motivo não é difícil encontrar empresas gastando entre 30 a 40% do esforço total do projeto em testes de *software*, dado que, quanto mais cedo defeitos forem encontrados no *software*, menor será o custo de sua correção.

A empresa Sistemas de Informação Industriais e Consultoria (SINFIC) explica que as grandes organizações empresariais são totalmente adeptas da ideia do incentivo

financeiro e técnico ao departamento onde são planejadas as inovações, como melhor forma de criar um diferencial em relação à concorrência. E nesse sentido, enfatiza-se o apoio ao pessoal destacado ao trabalho de criar e testar os *softwares*, mas alertam que quanto mais avançado é o produto mais vulnerável se torna a ameaças, surgimento de erros e baixa qualidade (Sinfic, 2006).

De acordo com Souza (2017), para desenvolver qualidade de *software* é necessário possuir pessoas especializadas e aperfeiçoar o processo de teste, uma vez que o processo possibilita analisar a escalabilidade de um produto, facilita a introdução das melhores práticas e novas tecnologias no ambiente de desenvolvimento. Logicamente acentuando aqui que mesmo diante de tantos avanços na informática, ainda existem processos considerados como indefinidos e imaturos.

Nas últimas décadas modelos de referência de melhoria do processo foram desenvolvidos, estes modelos são utilizados como ferramentas para conseguir mapear e melhorar o processo de desenvolvimento dos *softwares*. Entre os modelos encontra-se o MPT.BR (Melhoria de Processo do *Software* Brasileiro), que tem como objetivo maximizar os resultados encontrando as melhores técnicas de testes que se adequem a um tipo de *software* em específico ao longo de seu ciclo de vida.

Esse tipo de modelo de maturidade de teste de *software* possui características que o diferenciam, pois ele é capaz de diminuir a tacha de retrabalho através de aplicação de práticas aos processos de teste dentro do ciclo de vida do produto, dessa forma é possível encontrar com maior facilidade erros, defeito ou falha, bem como, a omissão de interface; ambiguidade; inconsistência; fato incorreto e informações consideradas estranhas ao ambiente em que o produto deve atuar.

Possuir times focados em testes de *software* é importante para a execução das atividades de testes, entretanto para os times alcançarem êxito em suas atividades é necessário mapear o processo. A partir do mapeamento é possível identificar falhas e realizar a correção das mesmas. De acordo com Souza (2017), o mapeamento de processo permite visualizar as diferentes etapas e tarefas em uma sequência cronológica, o que permitirá que todas as etapas sejam descritas de maneira sucinta, trazendo uma melhoria na análise do processo em questão e, se possível, identificar as melhorias a serem implementadas.

Muitos times de teste de *software* realizam suas atividades de maneira incorreta, e erram na estimativa de suas atividades uma vez que os mesmos, não possuem dados

precisos nem confiáveis, o que acaba sendo um dos fatores que impactam diretamente no tempo e custo para a entrega do projeto (Souza, 2017).

Assim, entende-se que os testes pela própria dinâmica evolutiva dos sistemas de informação, também deve ser constantemente avaliado, no sentido de estarem sempre preparados para enfrentar situações cada vez mais sofisticadas. Os testes devem ser utilizados em todas as camadas produtivas do *software*, pois desta forma os erros e incompatibilidades são encontradas de maneira muito mais veloz e desonera o produto final.

Mas de que maneira é possível evoluir processos de testes dos *software*s dentro de um modelo de maturidade compatível com as necessidades de uma organização e agregando qualidade ao produto e/ou serviço prestado?

Ao coletar informações de um processo de teste de *software* já existente e aplicado em projetos de uma organização, evidenciará a situação atual do mesmo, possibilitando conhecer e avaliar o fluxograma de execução de suas atividades, desta forma, permitindo a identificação de possíveis adequações neste processo, com base nas convenções estipuladas ao primeiro nível do modelo de maturidade do MPT.BR.

Durante a execução deste processo de intervenção, será possível abordar alguns tópicos como: Selecionar e analisar um processo de teste de uma fabrica de *software*; Propor evoluções há processo de teste; Otimizar o gerenciamento dos testes de *softwares* quantificando os resultados obtidos através de artefatos de controle.

Ao avaliar e sugerir melhorias em um processo de *software* este trabalho pode ser caracterizado como uma pesquisa-ação, segundo Tripp (2005), a pesquisa-ação favorece as discussões e a produção cooperativa de conhecimento específica sobre a realidade vivida a partir de perspectivas do esmorecimento das estruturas hierárquicas que fragmentam o cotidiano de um grupo, além de possuir definições de uma pesquisa descritiva de natureza exploratória, segundo Gil (2002), no qual se tem o objetivo descrever características de determinadas populações ou fenômenos.

Nos próximos capítulos desta pesquisa, serão apresentados os conceitos por trás de um processo de teste de *software* e sua importância como parte complementar de uma gestão de projeto, também será abordado sobre boas práticas conforme um modelo de maturidade e como uma intervenção a um processo de teste pode ser realizada a fim de identificar melhorias no mesmo, a partir dos resultados obtidos durante a realização deste

projeto, ao final será possível expor considerações e conclusões sobre a pesquisa e sua aplicação.

2. FUNDAMENTAÇÃO TEÓRICA

2.1 Engenharia de *Software*

Sommerville (2007) salienta sobre a importância da conceituação de *software*, em seu entendimento é comum observar que a maioria vê esse produto como um mero programa de computação, para o autor em destaque, elementos característicos como dados de documentação e configuração associada são peças fundamentais para o bom funcionamento de toda a programação.

Engenharia de *Software* é uma área de estudo que contempla todas as etapas produtivas para o desenvolvimento de um novo *software*. Aborda desde os estágios fundamentais onde as especificações do sistema são identificadas e registradas até a fase de manutenção. Segundo Pressman (2016), engenharia de *software* é o conjunto de métodos, procedimentos e ferramentas que auxiliam no gerenciamento do processo de produção de *software* que opere com a qualidade que se espera dele.

2.1.1 Processo de Engenharia de *Software*

Conforme Barcaui (2012), para o *Institute of Electricaland Electronic Engineers* (IEEE) o processo de engenharia em *software* é composto por um conjunto de etapas a serem seguidas visando um objetivo, por outro lado, no *Capability Maturity Model Integration* (CMMI) a produção de *software* se dá através de ações e métodos especificados e aplicados de maneira inter-relacionados na intenção de obter-se ao final, um produto ou serviço de qualidade.

Sommerville (2007) propõe a subdivisão da Engenharia de *Software* em quatro etapas fundamentais e comuns a todos os processos de construção de *software*, conforme descrito na Tabela 1:

Tabela 1 – Processo de produção de *software*

Etapa	Descrição
Especificação	Definir as funcionalidades e restrições do *software* a ser desenvolvido.
Desenvolvimento	Produzir o *software*, de modo que atenda às suas especificações.
Validação	Validar o *software* para garantir que atende ao que o cliente deseja.
Evolução	Evoluir e realizar manutenções para atender às necessidades mutáveis do cliente

Fonte: Sommerville (2007)

Pressman (2016), diz que em relação à engenharia de *software* os processos são determinados por um conjunto de atividades primárias, citando o desenvolvimento, a manutenção, as aquisições e contratações de *software*, posteriormente vem a fase dos subprocessos tais como: a análise, desenho, implementação e testes.

Ainda Pressman (2016) afirma que para a construção e desenvolvimento de *software*, primeiramente é preciso atentar com cuidado para a escolha do ciclo de vida que se adapte ao modelo de produto a ser produzido, e reafirma o que foi exposto na Tabela 1.

Este conjunto de atividades especifica definem um modelo de processo, onde possui métodos específicos para uma solução de desenvolvimento de *software*, pois segundo Sommerville (2007), o modelo de processo de *software* é uma forma de representar abstratamente o processo de *software*.

2.2 Métodos Ágeis

Os métodos ágeis são comumente utilizados no âmbito de gerencia de projetos, ainda mais quando se trata de desenvolvimento de *software*, devido ao fato de possuir abordagens simplificadas.

No início de 2001, houve uma reunião entre 17 representantes de técnicas e metodologias, o intuito desta reunião era estabelecer um padrão de desenvolvimento de projeto dentre as existentes e por fim o Manifesto para o Desenvolvimento Ágil de *Software*, que definiu um framework comum para processos ágeis. O entendimento e a aplicação deste framework envolvem boas práticas e melhorias de processos, gestão de projetos de forma ágil, técnicas e métodos de desenvolvimento (Beck, 2017).

Segundo o Pressman (2016), recomenda-se fortemente a utilização de métodos ágeis, pelos seus benefícios agregados: o aumento da integração dos envolvidos do projeto; aumento da qualidade do projeto; diminuição dos custos e riscos de desenvolvimento; aumento da satisfação dos usuários finais (clientes).

Um dos métodos mais aderidos na gestão de projetos de TI é o *Scrum* e devido ao fato da empresa envolvida neste projeto ser adepta, será abordado na próxima sessão as atividades utilizadas na mesma.

2.2.1 *Scrum*

Fortemente utilizada para gestão e planejamento de projetos de *software* o *Scrum* é uma metodologia ágil que carrega os princípios do Manifesto para o Desenvolvimento Ágil de *Software* e seus criadores foram: Mike Beedle, Ken Schwaber e Jeff Sutherland, sendo os mesmos autores do Manifesto.

O foco desta metodologia é no gerenciamento da equipe, preocupada com a organização de processos, no modo como as atividades serão executadas, deixando a responsabilidade dos participantes do projeto escolher a forma de concluir as etapas do desenvolvimento Schwaber (2004).

Conforme o *Scrum* (2014) as equipes geralmente são pequenas e nelas há a presença de três papeis, sendo eles:

- *ProductOwner* (Dono do Produto): representa o cliente e seus interesses no projeto, ele é responsável por agregar valor ao produto conforme as necessidades do cliente.
- *Scrum Master* (Mestre *Scrum*): responsável pela execução e aplicação do *Scrum* e responsável por remover quaisquer impedimentos encontrados pela equipe;
- *Team Scrum* (Time): é a equipe responsável pelo desenvolvimento do projeto, normalmente entre 6 a 10 pessoas.

No *Scrum*, artefatos são desenvolvidos a medida que o processo de desenvolvimento é executado, com o objetivo aumentar a transparência das informações para os envolvidos, os principais conforme Schwabere Sutherland (2013) são:

- *Product Backlog*: é uma lista com todas as funcionalidades a serem desenvolvidas de acordo com as necessidades do cliente, este conteúdo é desenvolvido pelo *ProductOwner*.
- *Sprint Backlog*: é uma lista baseada no *Product Backlog*, mas os itens desta lista são tarefas definidas pelo *Team Scrum*à serem desenvolvidas em uma Sprint, uma ou mais tarefas podem contemplar uma funcionalidade do *Product Backlog*.
- *ProductIncrement*: representa um pacote com um conjunto de itens desenvolvidos na Sprint, de forma que esteja em condições de uso para o cliente, podendo ou não ser liberadas conforme a decisão do *ProductOwner*.

Durante a execução do *Scrum* há eventos que ocorrem regularmente com duração máxima definida, esses eventos tem o intuito de permitir a transparência e a inspeção do

processo de desenvolvimento, segundo as definições de Schwabere Sutherland (2013), os eventos são:

- *Sprint*: é o principal evento geralmente costumam ocorrer mensalmente, basicamente uma Sprint pode ser chamado de ciclos onde possui um conjunto de atividades a serem executas. Essas atividades são definidas pelo *Product Owner* e negociadas com o *Team Scrum* assim definindo um objetivo para a Sprint, para haver o cancelamento de uma Sprint, deve se levar em consideração ao objetivo estipulado inicialmente, caso se torne obsoleto, não há motivos para se continuar a *sprint*.
- *Sprint Planning Meeting* (Reunião de Planejamento da Sprint): nesta reunião o *Product Owner* descreve as funcionalidades de alta prioridade do projeto para o *Team Scrum*. O *Team Scrum* tem o objetivo durante a reunião de entender as atividades proposta afim de que não fique nenhuma dúvida e quebrar as funcionalidades em tarefas técnicas, com as tarefas definidas se tornará a *Sprint Backlog*.
- *Daily Scrum* (Reunião Diária): todos os membros da equipe realizam esta reunião geralmente no mesmo horário do dia, durante a reunião cada membro deve responder as seguintes questões: O que você fez ontem?; O que você fará hoje?; há algum impedimento? Neste momento a equipe envolvida no *Scrum* adquiriu conhecimento sobre o que foi feito e que ainda precisa ser feito. Caso haja impedimentos, os mesmos devem ser reportados para o *Scrum Master*, é aconselhado que esta reunião possa ocorrer no tempo máximo de 15 minutos.
- *Sprint Review Meeting* (Reunião de Revisão do Sprint): Esta reunião deve ocorrer no fim da *Sprint*, todos os envolvidos do projeto devem participar, o *ProductOwner* avalia os objetivos da *Sprint* em questão, verificado se todos os itens definidos no *Sprint Backlog* foram realizados ou não, como resultado desta reunião é a atualização do *Product Backlog* para a próxima *Sprint* podendo ser alterado ou não conforme os resultados obtidos na Sprint anterior, esta reunião deverá ter um prazo máximo de quatro horas.
- *Sprint Retrospective* (Reunião de Retrospectiva do Sprint): Esta reunião ocorre no final de cada *Sprint*. Ela é direcionada pelo *Scrum Master* a fim de auxiliar o *Team Scrum* identificar o que foi bem-sucedido, o que deve ser melhorado e se tiver que ser melhorado algo, quais serão as devidas providências, esta reunião deve

acontecer no período de três horas para Sprint mensal, ou um período proporcional a *Sprint*.

2.2.1.1Execução do Scrum

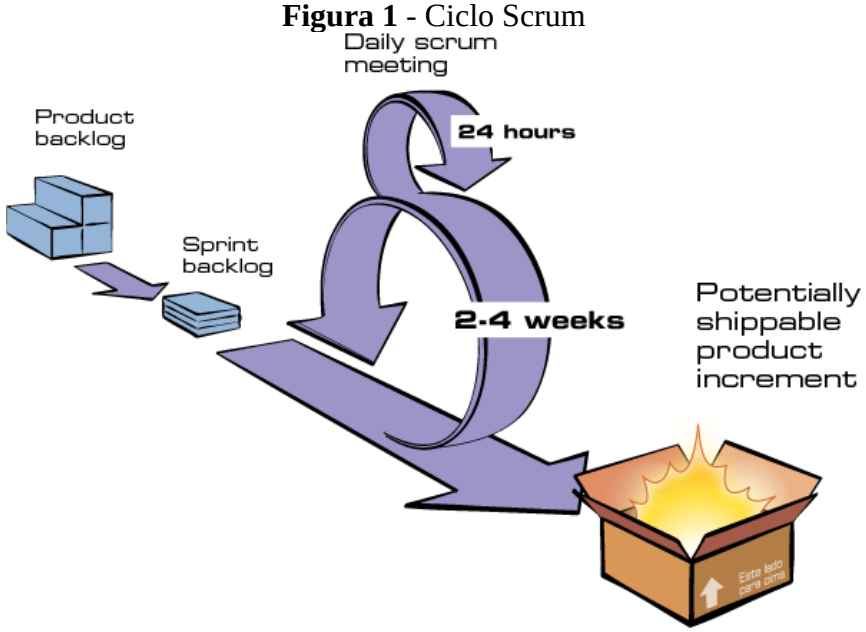
O *Scrum* tem início quando a visão do que deve ser realizado é criada, desta forma as funcionalidades serão definidas e alocadas por prioridade no *Product Backlog*.

Segundo Schwaber (2004), antes de iniciar uma *Sprint* será realizada uma *Sprint Planning Meeting* e obterá como artefato inicial a *Sprint Backlog* definidos pelo *ProductOwner* e o *Team Scrum* onde decidem o conjunto de tarefas que deverá ser desenvolvido durante a *Sprint*.

A cada dia da *Sprint* são realizadas as *Dailies Scrum*para acompanhar o progresso da *Sprint*. No final da *Sprint*, uma *Sprint Review Meeting* deve ser realizada para a apresentação dos resultados e logo após a *Sprint Retrospective*.

Neste momento ao se obter os resultados satisfatórios e itens do *Product Backlog*, se obtém uma pequena parte do *software* onde contém as funcionalidades desenvolvidas. Durante a execução de mais *Sprints* será possível incrementar os resultados, até obter o resultado final desejado pelo cliente. A Figura 1 representa o ciclo do *Scrum*.

Figura 1 - Ciclo Scrum



Fonte: Scrum (2019)

Todo esse processo visa ao final construir um produto que atenda às especificações ao qual foi projetado, oferecendo ao cliente a certeza de bom funcionamento e com a qualidade que se espera.

2.3 Garantia da Qualidade de *Software*

Batista (2014) conceitua qualidade como sendo um conjunto de características positivas que visam atender à necessidade dos clientes e dentro dos padrões determinados pelo produto que a empresa produz. Fala ainda que tais características podem ser morais, qualidade intrínseca, a entrega, o custo final e a segurança.

Pressman (2016) ressalta que a qualidade em um *software* é conceituada como a observância aos requisitos de funcionamento e de desempenho plenamente especificados, obedecendo a padrões legais de garantia de que o produto seja eficaz e com a qualidade que o cliente espera encontrar. Sendo que esta definição deve destacar três pontos importantes: i) requisitos é a base a partir da qual a qualidade é medida; ii) padrões especificados definem um conjunto de critérios de desenvolvimento a serem seguidos para obtenção da qualidade; iii) existência de um conjunto de requisitos implícitos que frequentemente não são mencionados na especificação.

É importante frisar que a definição acima vai de encontro ao que é defendido na proposta da ISO 9000:2005 (ABNT, 2005), que trata a qualidade como um patamar do produto dentro de um conjunto de características pertinentes a ele mesmo e que satisfaz aos requisitos que lhe são esperados. Assim, se torna importante dedicar o tempo necessário à realização dos testes e dessa forma ter a possibilidade maior de entregar um produto de qualidade ao cliente (Cérgoli, 2017).

Para que, em organizações maduras, possam conseguir atingir seus objetivos de qualidade, prazos e custos de forma consistente e eficiente é necessário ter conhecimento modelo de maturidade a ser empregado, tema esse que será melhor conceituado e discriminado a seguir.

2.4 Modelos de Maturidade

Segundo Vasconcelos (2006) existem diversas maneiras, normas, modelos e padrões, para se conseguir a qualidade de *software*. A evolução desses artefatos tem evoluído nos últimos anos graças ao empenho mundial em seu estudo, o que une todos

esses modelos é o fato de que em todas as possiblidades o que se apresenta como objetivo é minimizar o tempo gasto e os impactos do custo no preço final do *software*.

Villas Boas (2007) esclarece que essas normas podem ser nacionais, regionais, organizacionais ou internacionais. Em instância internacional destaca-se a *International Organization for Standardization* (ISO) e a *International Electrotechnica l Commission*(IEC), enquanto que as normas Brasileiras são editadas pela a Associação Brasileira de Normas Técnicas (ABNT).

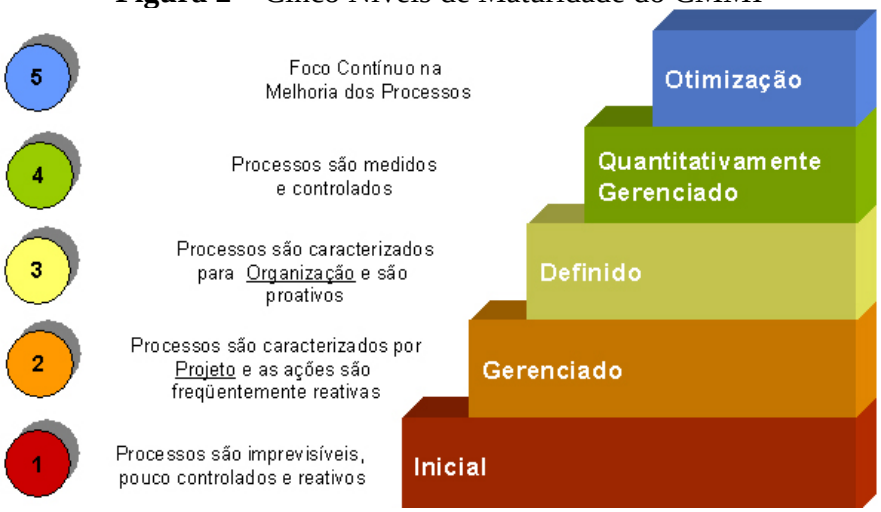
Na sessão a seguir serão apresentados alguns dos modelos de maturidade e suas definições, os mesmos são voltados à processo.

2.4.1 Capability Maturity Model Integration (CMMI)

O Modelo Integrado de Maturidade da Capacidade de Processo de *Software* (CMMI) foi criado no ano de 2002 e, segundo Quintellaet al. (2005), seu objetivo é a integração dos modelos existentes provendo uma estrutura coerente e alinhada.

Este modelo é estruturado em 5 níveis referentes aos estágios de maturidade que a organização possui. Cada nível consiste em relacionar práticas específicas e genéricas para uma área de processo, seu objetivo é a redução do custo da implementação da melhoria de processo, eliminando inconsistências e estabelecendo diretrizes que auxiliem as organizações nos vários estágios de um projeto de *software*.

Figura 2 – Cinco Níveis de Maturidade do CMMI



Fonte: ISD (2011)

Segundo Presmam (2016), sua arquitetura é composta pela definição de um conjunto de 22 áreas de processo, organizadas em duas representações: uma por estágio, na qual as áreas estão agrupadas em 5 níveis de maturidade, sendo: Inicial, Gerenciado, Definido, Gerenciado Quantitativamente e Otimização. Suas informações detalhadas são apresentadas por estágio a seguir na Tabela 2.

Tabela 2– Discriminação dos trabalhos por estágio no CMMI

Nível	Nome	Capacidade
1	Inicial	Desenvolvimento de produtos de *software* de alta qualidade, seu desempenho depende da competência das pessoas, não existe nenhuma área de processo definida.
2	Gerenciado	Os métodos de gerenciamento de *software* são documentados e acompanhados. Políticas organizacionais orientam os projetos estabelecendo processos de gerenciamento. São ao todo 7 áreas de processo: Monitoramento e Controle de Projeto, Planejamento de Projeto, Garantia da Qualidade de Processo e Produto, Medição e Análise, Gestão de Configuração, Gestão de Requisitos e Gestão de Contrato com Fornecedores.
3	Definido	O processo de engenharia é bem definido, nele possui uma preocupação com um processo padronizado para a organização, customizado para cada projeto. O projeto é definido, documentado e compreendido.
4	Quantitativo gerenciado	A gerência já possui seus objetivos a serem alcançados, o processo é medido e gerenciado quantitativamente. Aqui se tem noção do desempenho dentro de limites quantificados.
5	Otimização	Foco na melhoria contínua do processo, a mudança de tecnologia e as mudanças no próprio processo são gerenciadas visando não trazer impacto na qualidade do produto final. Aqui tem-se duas áreas de processo: Implantação de Inovações na Organização, Análise e Resolução de Causas.

Fonte: Adaptado de Presmam (2016)

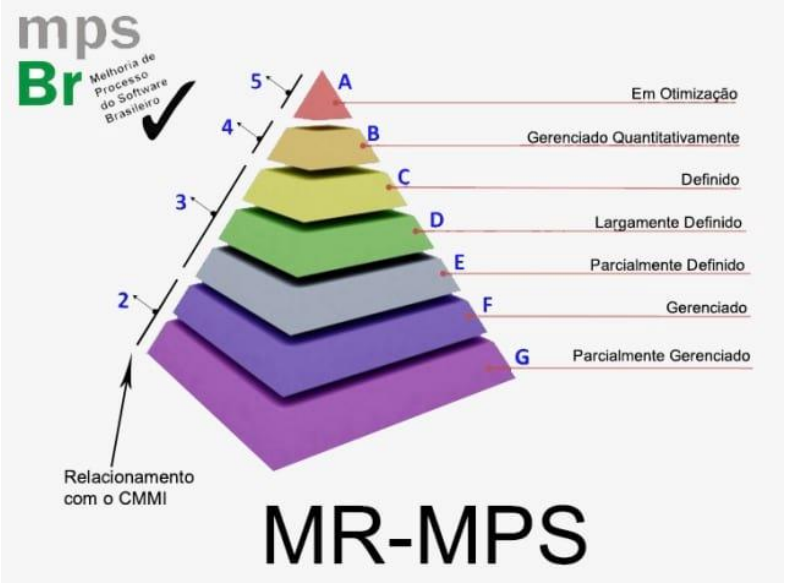
Observa-se que no conjunto todos os níveis de maturidade possuem suas metas e objetivos a serem alcançados no processo de construção do *software*, ao se atingir essas metas e objetivos níveis seus resultados são utilizados nas avaliações dentro de cada uma das áreas de maturidade. A avaliação pode ser considerada como satisfeita quando todas as metas sejam elas genéricas ou específicas tenham sido satisfeitas em sua totalidade, caso exista uma só discordância todo o processo é considerado como não satisfeito (Quintella et al, 2005).

2.4.2 Melhoria de Processo do *Software* Brasileiro (MPS.BR)

De acordo com a Associação para Promoção da Excelência do *Software* Brasileiro(SOFTEX), a Melhoria de Processo do *Software* Brasileiro (MPS.BR), foi instituída no ano de 2003, com o gerenciamento da SOFTEX e apoiado por várias organizações brasileiras, entre elas: Ministério da Ciência e Tecnologia (MCT); Banco Interamericano de Desenvolvimento / Fundo Multilateral de Investimentos (BID/FUMIN);Serviço Brasileiro de Apoio às Micro e Pequenas Empresas (SEBRAE) e a Financiadora de Estudos e Projetos (FINEP) SOFTEX (2012).

Se trata de um modelo que se adequa aos cinco níveis do CMMI, mas, possui características próprias em relação aos trabalhos exercidos em cada um deles, já que nesse modelo sua escala vai de “A” até “G”, conforme apresenta a Figura 3.

Figura 3 –Modelo Melhoria de Processo do *Software* Brasileiro (MPS.BR)



Fonte: SOFTEX (2012)

Este modelo tem como principal objetivo traçar um caminho com um custo viável e compatível com a realidade financeira brasileira, dessa forma as pequenas e médias empresas do país sejam beneficiados e possam adquirir produtos de *software* com a qualidade necessária. Em relação aos processos, a Tabela 3apresenta as devidas discriminações de cada trabalho realizado por esse modelo.

Tabela 3 - Discriminação dos trabalhos por estágio no MPS/BR

Nível	Nome	Capacidade
G	Parcial Gerenciado	Composto pelos processos Gerência de Projetos e Gerência de Requisitos
F	Gerenciado	Composto pelos processos do nível de maturidade (G) acrescidos dos processos Aquisição, Garantia da Qualidade, Gerência de Configuração, Gerência de Portfólio de Projetos e Medição.
E	Parcial Definido	Composto pelos processos dos níveis de maturidade anteriores (G e F), acrescidos dos processos Avaliação e Melhoria do Processo Organizacional, Definição do Processo Organizacional, Gerência de Recursos Humanos e Gerência de Reutilização
D	Largamente definido	Composto pelos processos dos níveis de maturidade (G ao E), acrescidos dos processos Desenvolvimento de Requisitos, Integração do Produto, Projeto e Construção do Produto, Validação e Verificação.
C	Definido	Composto pelos processos dos níveis de maturidade (G ao D), acrescidos dos processos Desenvolvimento para Reutilização, Gerência de Decisões e Gerência de Riscos.
B	Gerenciado quantitativamente	Composto pelos processos dos níveis de maturidade (G ao C). Neste nível o processo de Gerência da Operação do Serviço sofre sua segunda evolução, sendo acrescentados novos resultados para atender aos objetivos de gerenciamento quantitativo.
A	Otimizado	Composto pelos processos dos níveis de maturidade anteriores (G ao B). Este nível não possui processos específicos.

Fonte: Adaptado de SOFTEX (2012)

É preciso ressaltar que o modelo MPS.BR obedece aos parâmetros exigidos pelas principais abordagens internacionais relativas à definição, avaliação e a melhoria do desempenho de *software*, tornando compatível e de fácil utilização. Os diferentes níveis de composição deste modelo servem como indicativo para que as pequenas e médias empresas possam medir o grau de qualidade em que a organização esteja.

2.5 TESTES DE *SOFTWARE*

Os testes de *softwares* são de extrema importância para se obter a garantia de qualidade de um sistema, pois o mesmo, deve garantir que o produto desenvolvido atenda todos os requisitos solicitados pelo cliente (Myers, 2004). Os testes são processos, sendo parte fundamental de um projeto de desenvolvimento de um *software*, com o objetivo de

descobrir falhas, erros e verificar se os mesmos foram corrigidos, maximizando a qualidade da entrega do produto.

2.5.1 Processo de Teste de *Software*

Para Eliza e Lagares (2012), em um processo de teste de *Software* se objetiva a estruturação das atividades, os artefatos, os papéis e responsabilidade do teste dentro do conjunto de construção do *software*. Entre os artefatos mais importantes se pode citar: os diagramas de classes, os requisitos específicos do *software* e documentos do projeto, são esses elementos que auxiliam na construção, estrutura e funcionalidade do produto.

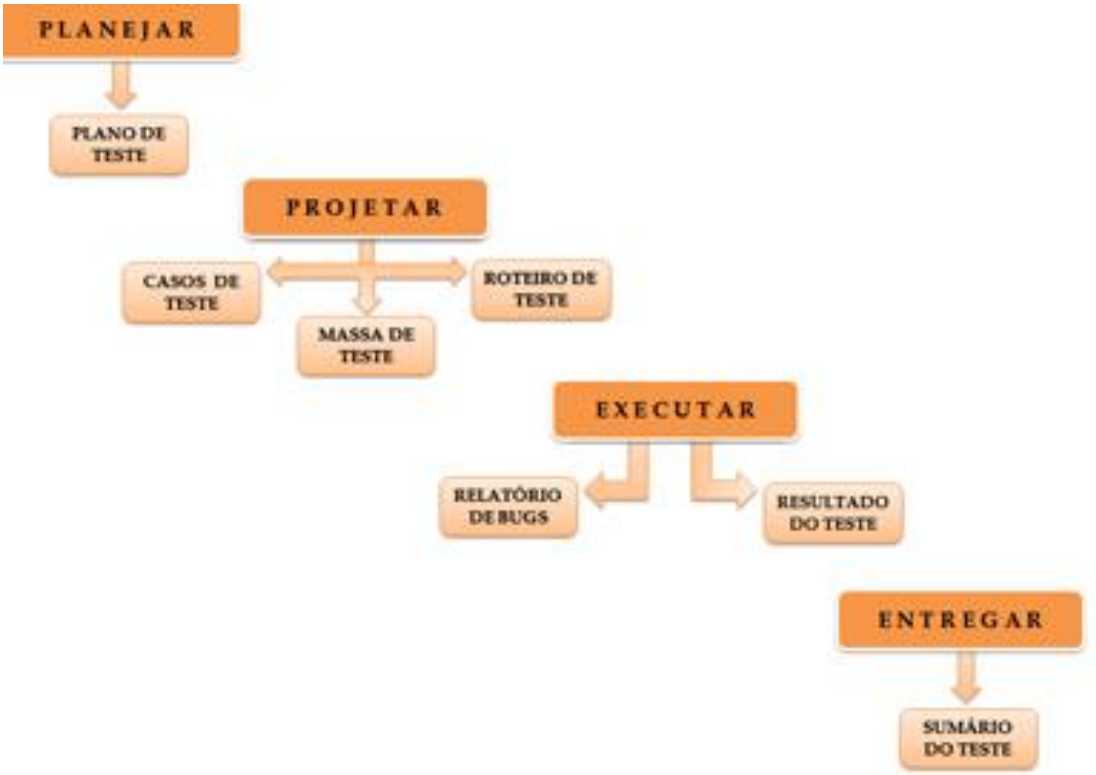
Villas Boas (2007), o principal objetivo do processo de teste de *software* é o de padronizar o uso desse tipo de ferramenta. Com o uso dos testes de *software* a equipe de desenvolvimento pode ir corrigindo os erros que possam surgir de maneira concomitante com o desenvolvimento do projeto e ao final tem-se maior probabilidade de oferecer a necessária qualidade que se espera do produto. E para que isso aconteça o processo é dividido em quatro etapas, como demonstra a Tabela 4 e na representação destas etapas na Figura 4:

Tabela 4 – Etapas do teste de *software*

Etapa	Descriminação
Planejamento e controle	Tem o objetivo de elaborar os métodos de testes e o plano de teste a ser utilizado, a fim de minimizar os riscos do negócio e orientar os testadores de *software* para as próximas etapas dos testes.
Projeto	Aqui se fazem as escolhas dos critérios do *software* a serem testados, especificando os métodos empregados e os testes que deverão ser feitos.
Implementação	Fase de especificação dos itens a serem testados, descrevendo as entradas e saídas para comparar se realmente é o resultado esperado com a finalidade de avaliar um conjunto de características, gerando o documento de caso de teste e procedimento de teste.
Execução	Hora da parte manual do teste, ou seja, é a hora em que se efetiva a tarefa de teste propriamente dita, nesta fase o procedimento de teste será seguido e os casos de testes serão associados a uns testados para que o mesmo execute os testes.

Fonte: Villas Boas (2007)

Figura 4 – Esquema das etapas dos testes de *software*



Fonte: Adaptado de Vilela Jr. (2012)

Souza (2017) ressalta a importância do teste e reafirma a validade do processo levantado na Tabela 4 acrescentado que para uma maior segurança os testes devem ser aplicados em vários níveis. O primeiro seria o teste de unidade que Wazlawick (2012) discrimina como sendo o mais básico e serve primordialmente para verificar se alguma unidade do *software* foi realmente implantada de maneira correta pela sua característica a execução pode ser feita apenas pelo programador.

O segundo é o teste de aceitação que nas palavras de Myers (2004), trata-se do processo de teste conhecido como caixa preta sendo realizado em um sistema e deve ser aplicado antes de ser disponibilizado o seu acesso aos interessados. O principal objetivo desse tipo de teste é rever se o dispositivo atende às especificações e o que a clientela espera de seu funcionamento.

O terceiro é o teste de integração que Humble e Fearley (2014) descrevem como sendo aplicado na eventualidade da ligação de vários sistemas externos via protocolos diferenciados. Tal modalidade de teste é indicado quando o usuário fará uso do *software* para acoplar vários módulos que trabalhem com baixo acoplamento e interações complexas.

Por fim o teste de sistema, nesta fase é necessário o uso de ferramentas específicas, pois o produto já se encontra em fase final de concepção. Com a utilização destas ferramentas específicas o produto pode ser totalmente testado e sua qualidade de funcionamento em várias situações devidamente comprovada. São verificadas situações cotidianas de uso e como o *software* reage a cada uma delas (Bartié, 2002).

2.6 Modelos de Maturidade de Teste

Assim como os processos de desenvolvimento de *software*, os processos de testes possuem modelos de maturidade.

Para Myers (2004), existe uma lista de regras que podem ser aplicadas a teste de *software*. O autor defende a possibilidade de verificar quais os limites e habilidades do produto com total clareza para que na ocorrência de erros eles possam ser prontamente corrigidos. Quanto mais testado for o produto, então, menor a probabilidade de que ao final não se consiga obter um *software* de qualidade.

Cérgoli (2017) diz que o teste de *software*s trata de uma ação de máxima importância na busca pela qualidade desse produto. Relembra o autor que os seres humanos são passíveis de erros, por isso, testar *software* é uma das maneiras mais confiáveis se tratada de maneira sistemática se torna essencial para que os possíveis erros sejam encontrados e corrigidos concedendo desta forma a garantia da qualidade.

Nesse sentido, de acordo com a Softex (2012), podem ser adotados em conjunto dos modelos de maturidade CMMI e MPS.BR, modelos de maturidade de teste pois possuem semelhanças em suas áreas de processos, conforme serão discriminados os modelos TMMI e MPT.BR, a seguir:

2.6.1 Test Maturity Model Integrated (TMMI)

Desenvolvido pela TMMI Foundation essa ferramenta se trata de um modelo detalhado para a melhoria do processo de teste, sendo complementar ao CMMI, seu desenvolvimento utiliza o modelo TMMI, com o pressuposto inicial de que todas as organizações iniciam no nível 1 da escala de maturidade (TMMI, 2009).

E a Tabela 5 discrimina os cinco níveis da seguinte maneira:

Tabela 5 – Discriminação dos trabalhos no modelo de maturidade TMMI

Níveis	Discriminação
1 – Inicial	Aqui o teste é um processo indefinido em um ambiente instável em que as atividades são testadas de qualquer maneira afim de encontrar *bugs* no sistema.
2 - Gerenciado	A abordagem de teste fundamental é estabelecida e gerenciada, que pode variar de projeto para projeto dentro de uma organização com política de teste e estratégia sendo uma área de foco, este nível fornece orientação para planejamento e controle de testes com técnicas de projeto de teste com cada projeto tendo controle sobre a execução do teste.
3 – Definido	Estabelece que todos os projetos sigam os mesmos padrões e procedimentos em toda a organização ou unidade organizacional.
4 – Gestão e medição	A medição de atividades e resultados é aplicada exaustivamente no início de todos os projetos para estabelecer, em cada estágio de desenvolvimento, que os projetos são tão livres de defeitos quanto possível.
5 – Otimização	Todas as atividades e resultados são avaliados e as atividades otimizadas são implementadas para garantir a melhoria contínua em relação à prevenção de defeitos e à qualidade otimizada.

Fonte: Adaptado de Costa (2016)

Conforme a fundação TMMI (2009), esse modelo é estruturado com uma arquitetura dividida em etapas que visam a melhoria de processos, com níveis no qual uma organização passa à medida que seu processo de teste evolui, elevando a qualidade do *software* com menos defeitos e maior qualidade em sua produção.

A passagem pelos diferentes níveis de maturidade aumenta a capacidade do gerenciamento de qualidade de teste e *software* para se alinhar às necessidades do negócio ou projeto (Costa, 2016).

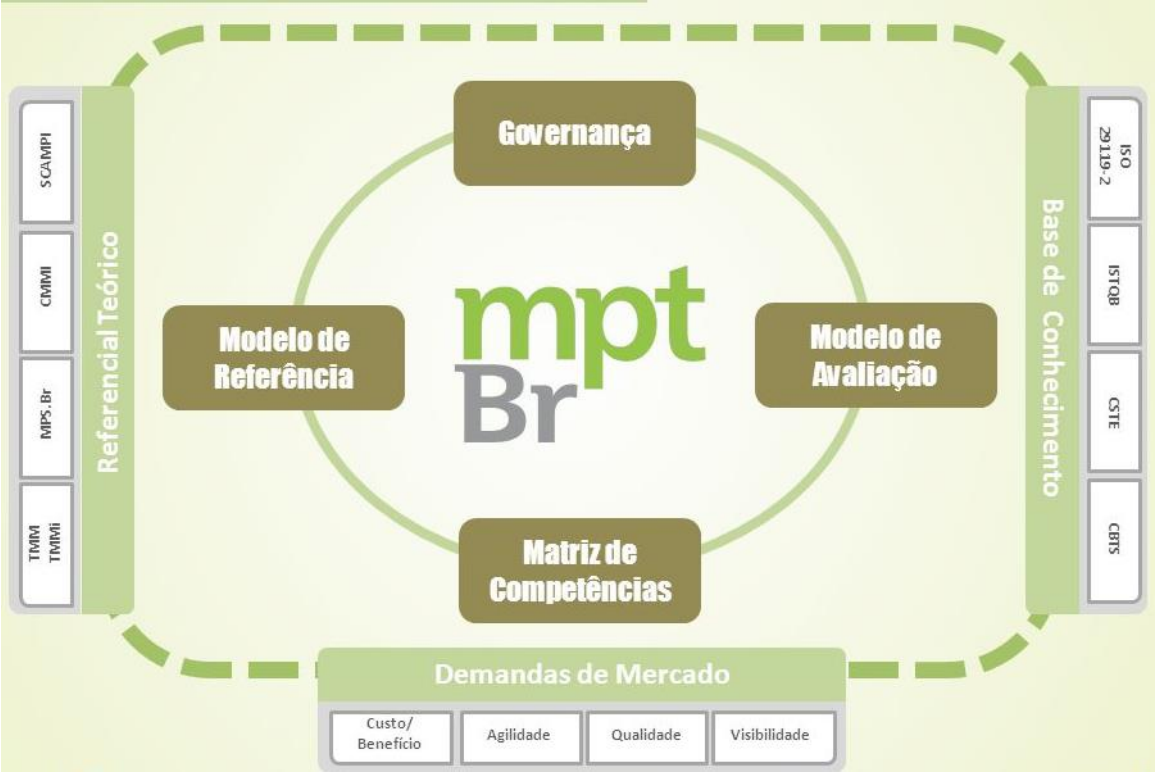
2.6.2 Melhoria do Processo de Teste Brasileiro (MPT.BR)

Conforme informações da organização que gerencia o MPT.BR, esse modelo de maturidade do processo de teste é composto por dois elementos: o modelo de referência no qual se apresenta sua estrutura, as áreas do processo e as práticas desenvolvidas; o guia de

avaliação, que é o documento em que está descrito as instruções para uso e análise dos resultados obtidos pelo MPT.BR, conforme o (MPT.BR, 2013).

O modelo em estudo é composto por cinco níveis de maturidade, sendo que cada um deles apresenta um conjunto de áreas de processo. As áreas de processos são conjuntas de práticas inter-relacionadas que trabalham coletivamente na busca de atingir um objetivo comum, conforme se pode verificar na Figura 5.

Figura 5 – Estrutura do modelo de maturidade de teste MPT.BR



Fonte: MPT.BR (2013)

Para Narcizo (2015) o modelo MPT.BR é uma referência nos processos de teste de *software*, abordando dentro dos objetivos das organizações a busca pela melhoria nos processos. Através de seu uso pode-se alcançar uma maior base de dados para avaliação e identificação da maturidade atual, além de reunir práticas mais evoluídas e estruturadas de acordo com o grau de complexidade em contraponto com o nível de maturidade que a mesma esteja relacionada.

O gerenciamento do MPT.BR é dividido em cinco níveis de trabalhos distintos, mas correlacionados entre si, sendo que a tarefa cumprida se passa a outra de maneira concomitante e conjunta. Sendo que estes níveis estão dispostos e apresentados conforme descrito na Tabela 6 da seguinte forma:

Tabela 6 - Discriminação dos trabalhos por níveis no MPT.BR

Nível	Descrição
Parcialmente gerenciado	Primeiro patamar de maturidade de uma organização. É composto pela premissa necessária em que se demonstra a disciplina de teste e sua aplicação é planejada e controlada.
Gerenciado	A partir desse momento a visibilidade se torna maior. O escopo do projeto passa a ser controlado pelo processo de gestão de mudanças, padrões são definidos e os processos são monitorados e controlados.
Definido	Nível organizacional nível no qual se dá o início do processo de busca pela qualidade do produto, as orientações da responsabilidade são adotadas através do sistema de medição. Aqui o ciclo de vida é adotado com o teste estático e de aceitação sendo formalizado e os procedimentos sistemáticos são aplicados para o fechamento do teste.
Prevenção de defeitos	Voltado para a detecção de defeitos e melhoria sistemática da qualidade do produto acompanhada e ações proativas para evitar que novos defeitos originados pelas mesmas causas raiz ocorram. Também é realizada uma análise para determinar a eficácia do teste e a determinação com dados objetivos do nível de qualidade do produto.
Automação e otimização	Seu objetivo é estabelecer um processo de melhoria contínua e automação do teste. Suas características principais são: abordagem sistemática de automação e execução do teste, seleção e escolha das ferramentas e utilização da CASE, todo processo é vistoriado de forma estatística e visando a melhoria evolutiva.

Fonte: MPT.BR (2016)

Conforme a elevação dos níveis, maior é a aderência das práticas específicas e as práticas genéricas definidas para cada nível, geralmente para a comprovação de que a unidade de teste conseguiu aderir há um nível é aplicado um exame de evidências (diretas, indiretas e afirmações), que comprovam que a empresa implantou com sucesso as práticas definidas pelo nível específico.

Na próxima sessão será abordado com ênfase no detalhamento do nível 1, tendo em vista que será o utilizado para realizar a avaliação.

2.6.2.1 Nível 1 – Parcialmente Gerenciado

O MPT.BR possui um nível exclusivo para a Gerência de Projeto de Teste, por sua vez tem como exigência pré-requisitos para alcançar um nível de maturidade, estes requisitos estão voltados para a organização seja capaz de gerir seus projetos de teste, possibilitando como base para que ela possa elevar sua maturidade entre os níveis a diante.

No primeiro nível é de extrema importância que a organização faça uso de projetos de teste paralelamente com o desenvolvimento do *software*, de forma que preferencialmente os mesmos possam ser inicializados juntos, havendo assim um paralelismo e integração de forma gerenciável.

Todos os níveis instituídos no MPT.BR possuem práticas genéricas e como exigência das áreas de processo, no caso do primeiro nível, possui as seguintes áreas de processo: Projeto e Execução de Teste e Gerência de Projetos de Teste de *Software*.

2.6.2.1.1 Gerência de Projetos de Teste de *Software* (GPT)

> O objetivo da área de processo Gerência de Projetos de Teste de Software é estabelecer e manter planos para gerenciar, monitorar e controlar as atividades até o encerramento do projeto(MPT.BR, 2013).

Um planejamento para um projeto de teste possui como um dos seus objetivos minimizar os riscos associados e durante o mesmo deve ser abordado alguns itens, de extrema preocupação como:

- Definir os objetivos;
- Levantar os riscos do *software*;
- Definir um plano de execução para os testes;
- Estimar os atributos de projeto e suas tarefas;
- Elaborar cronograma de entregas.

Conforme MPT.BR (2013), todas estas informações levantadas em um planejamento devem estar presentes no Plano de Teste, documento que descreve os artefatos desenvolvidos no planejamento, servindo como um guia. Os parâmetros deste planejamento servem como uma linha de progresso, onde o gerenciamento do projeto possa ser realizado, de forma que caso seja encontrado alguma inconsistência ações corretivas possam ser aplicadas o mais breve possível.

Nesta área possui 20 práticas específicas que são utilizadas para a verificação e avaliação de quanto a organização é aderente ao primeiro nível. Cada prática possui uma especificação conforme abaixo conforme o Guia de Referência do MPT.BR (2013):

GPT1 – Realizar análise de risco: Nesta prática possui como propósito, realizar uma profunda análise no produto, com o intuito de identificar áreas críticas e assim estipular testes específicos para minimizar falhas.

GPT2 – Estabelecer objetivos do teste: Nesta prática possui como propósito, definir os objetivos de teste. Por exemplo, testes com objetivos de verificar volume de utilização, onde, o sistema deverá suportar uma carga máxima de 3500 usuários acessando simultaneamente com degradação de desempenho máximo de 10% em qualquer operação.

GPT3 – Definir estratégia de teste: Nesta prática possui como propósito, definir a execução do projeto conforme objetivos e riscos levantados.

GPT4 – Definir escopo do projeto de teste: Esta prática possui como propósito, elaborar o escopo do projeto conforme as necessidades a partir disto podem ser feito as estimativas do projeto como o tempo que será empregado.

GPT5 – Estabelecer estimativas de tamanho: Esta prática possui como propósito, estipulara amplitude das atividades e os artefatos que serão desenvolvidas.

GPT6 – Definir o ciclo de vida: Nesta prática é escolhido o ciclo de vida do projeto de teste que servirá de base para planejar o projeto.

GPT7 – Estimar o esforço e o custo: Com base nos métodos e histórico do projeto, nesta prática é possível realizar uma estimativa de custo e esforço para a realização do mesmo.

GPT8 – Estabelecer e manter o orçamento e o cronograma do projeto: Com os marcos e pontos de gerenciamento do projeto é possível ter o controle dos gastos e tempos do projeto nesta prática.

GPT9 – Identificar riscos: A prática de análise e planejamento de acordo com os feedbacks obtidos do projeto é possível gerenciar os riscos, controlando a probabilidade de que ocorra e seus tratamentos.

GPT10 – Planejar os recursos: Esta prática leva em consideração os recursos humanos disponíveis, levando em conta os perfis e conhecimentos para a eficiência no projeto.

GPT11 – Planejar o ambiente de teste para o projeto: Nesta prática levam-se em consideração todos os elementos necessários para que o ambiente de teste seja provido.

GPT12 – Planejar os artefatos e dados do projeto: Nesta prática estipular e planejar os artefatos a serem entregues com base nas informações relevantes do projeto, levando em consideração os métodos de obtenção, armazenagem e distribuição.

GPT13 – Estabelecer indicadores: Com esta prática visa proporcionar a gestão do projeto de teste, indicadores que irão facilitar a visualizações de informações específicas sobre o projeto.

GPT14 – Estabelecer o Plano de Teste: Ao definir um plano de ação para a execução de teste deve se levar em consideração aspectos relevantes ao projeto, sendo documentado.

GPT15 – Revisar e obter compromisso com o Plano de Teste: Nesta prática todos os interessados devem se comprometer a realizar a execução do projeto conforme o planejado, de acordo com o escopo definido.

GPT16 – Monitorar o projeto: Nesta etapa todos os atributos referentes ao projeto serão colhidos, como forma de feedback para que assim possa ser analisado o progresso e desempenho do mesmo.

GPT17 – Gerenciar o envolvimento dos *stakeholders*: É interessante que com essa prática, possa se ter encontros durante a execução do projeto em determinados momentos para que haja uma comunicação e interação entre as partes interessadas.

GPT18 – Executar revisões em marcos do projeto: Conforme o cronograma definido o objetivo desta prática é que possa haver revisões e análises em fases importantes do projeto.

GPT19 – Analisar e registrar os problemas identificados: Podem ocorrer falhas durante a execução do projeto e com o gerenciamento constante é possível identificar estes problemas, reportando-os o mais breve possível.

GPT20 – Estabelecer e acompanhar ações corretivas até a sua conclusão: Nesta prática visa-se minimizar as possibilidades de que falhas venham ocorrer novamente, assim aumentando a qualidade do produto desenvolvido.

2.6.2.1.2 Projeto e Execução de Teste (PET)

> O objetivo da área de processo Projeto e Execução de Teste é identificar, elaborar e executar casos de teste, registrando a execução do teste e as divergências entre os resultados atuais e esperados na forma de incidentes (MPT.BR, 2013).

Nesta área de processo os casos testes devem ser identificados e elaborados, de forma que os resultados que o sistema reproduz possam ser comparados com os resultados esperados, caso haja divergência os mesmos devem ser reportados na forma de incidentes. Envolvendo as seguintes etapas abaixo:

- Indicar os casos de teste
- Executar os casos de teste

- Reportar e acompanhar os incidentes

De acordo com o Guia de Referência do MPT.BR, em empresas que aplicam o modelo de processo de teste, mas que possuem uma baixa maturidade em teste, visam somente garantir que os testes estão sendo executados de forma correta baseadas nas práticas desta área, mas especificamente para o Nível 1 de maturidade não há preocupação para a existência de documentação adequada, mas que as atividades estão sendo executadas MPT.BR(2013). Abaixo as práticas específicas desta área, sendo elas:

As práticas específicas são utilizadas para a verificação e avaliação de quanto a organização é aderente ao primeiro nível. Cada prática possui uma especificação conforme abaixo:

PET1 – Identificar casos de teste: Esta prática tem como objetivo identificar, priorizar e documentar os casos de teste do sistema.

PET2 – Executar casos de teste: Esta prática tem como objetivo executar os casos de teste identificados e registrar as informações da execução no log do teste.

PET3 – Reportar incidentes: Esta prática tem como objetivo garantir que as divergências de comportamento apresentadas pela aplicação sejam reportadas na forma de incidentes.

PET4 – Acompanhar incidentes: Esta prática tem como objetivo garantir que todos os incidentes sejam analisados e acompanhados até o seu fechamento.

2.6.2.1.3 Práticas Genéricas (PG)

As Práticas genéricas são atividades que devem ser seguidas por todas as áreas de processos, independentemente do nível de maturidade. Porque as mesmas devem ser seguidas por todas as áreas de processo (MPT.BR, 2013).

PG1 – Atingir os resultados definidos

> O objetivo desta prática genérica é gerar os produtos de trabalho e fornece os serviços que são esperados a partir da execução do processo (MPT.BR, 2013).

Para que esta prática possa ser atendida a organização que está aplicando-a deve executar todas as práticas específicas de das áreas de processo do nível de maturidade que organização almeja.

Produtos típicos:

- Produtos típicos de cada prática específica da área de processo.

PG2 – Estabelecer uma política organizacional

> O objetivo desta prática é estabelecer e manter uma política organizacional para o processo. (MPT.BR, 2013).

A política do processo tem como objetivo principal, informar as expectativas da organização sobre o processo aderido, tornando assim essas expectativas visíveis a todos que são afetados por ele.

Produtos típicos:

- Política organizacional;
- Atualizações da política organizacional.

PG3 – Planejar a execução do processo

> Esta prática objetiva a definição de como será executado um determinado processo (MPT.BR, 2013).

Ao realizar o planejamento devem-se levar em consideração os recursos disponíveis, as responsabilidades e o tempo a ser empregado. Assim estabelecendo um documento como um plano de ação do projeto.

Produtos típicos:

- Descrição do processo;
- Planejamento da execução do processo.

PG4 – Identificar e disponibilizar recursos

> O objetivo desta prática é garantir que os recursos indispensáveis para executar o processo serão identificados previamente e estarão disponíveis quando forem necessários. (MPT.BR, 2013).

Basicamente os recursos para a execução do processo de teste incluem-se: recursos financeiros, condições físicas, pessoal e ferramentas.

Produtos típicos:

- Recursos identificados e disponibilizados para execução do processo;

PG5 – Definir responsabilidade e autoridade

> O objetivo desta prática é definir, atribuir e comunicar as responsabilidades para executar o processo, definindo também a autoridade. (MPT.BR, 2013).

Nesta prática define que os colaboradores e envolvidos no projeto precisam ser responsáveis pela execução do mesmo, desta forma definir responsabilidades e autoridades é necessária e as mesmas podem ser descritas no plano de execução do processo.

Produtos típicos:

- Definição de responsabilidades e autoridades para os executores do processo;

PG6 – Prover treinamento

> O objetivo desta prática é garantir que as pessoas que executam o processo são competentes em termos de formação, treinamento e experiência (MPT.BR,2013).

Todos os colaboradores devem estar aptos e possuir habilidades para a execução dos processos conforme suas responsabilidades, desta a forma a organização deve garantir através de treinamentos e preparação dos mesmos sobre o domínio das técnicas específicas e aplicação do processo.

Finalizando as pesquisas bibliográficas aqui apresentadas, pode-se observar a importância referente aos testes de *software*, entre suas características principais tem-se que é através da aplicação destas metodologias é que se pode oferecer aos clientes a qualidade esperada no produto a ser oferecido. Um dos grandes entraves é quanto aos custos, o que foi largamente rebatido pelos autores.

Por fim, cabe ressaltar que todas as metodologias apresentadas são uteis, devendo ao responsável técnico fazer a melhor escolha diante das configurações que o *software* a ser produzido venha a fornecer e o funcionamento esperado pelo mesmo.

3.ABORDAGEM DO PROCESSO PROPOSTO

3.1 Seleção de um Processo de Teste de *Software*

Para Cervo e Bervian (2002) as técnicas de pesquisas são os passos que devem ser realizados para conclusão do trabalho. Neste sentido, na primeira etapa será escolhido um processo de teste de *software* para realizar uma observação em um ambiente de fábrica de *software* com o apoio de um questionário para coletar dados acerca da situação atual em relação ao nível de maturidade do processo de teste de *software* existente.

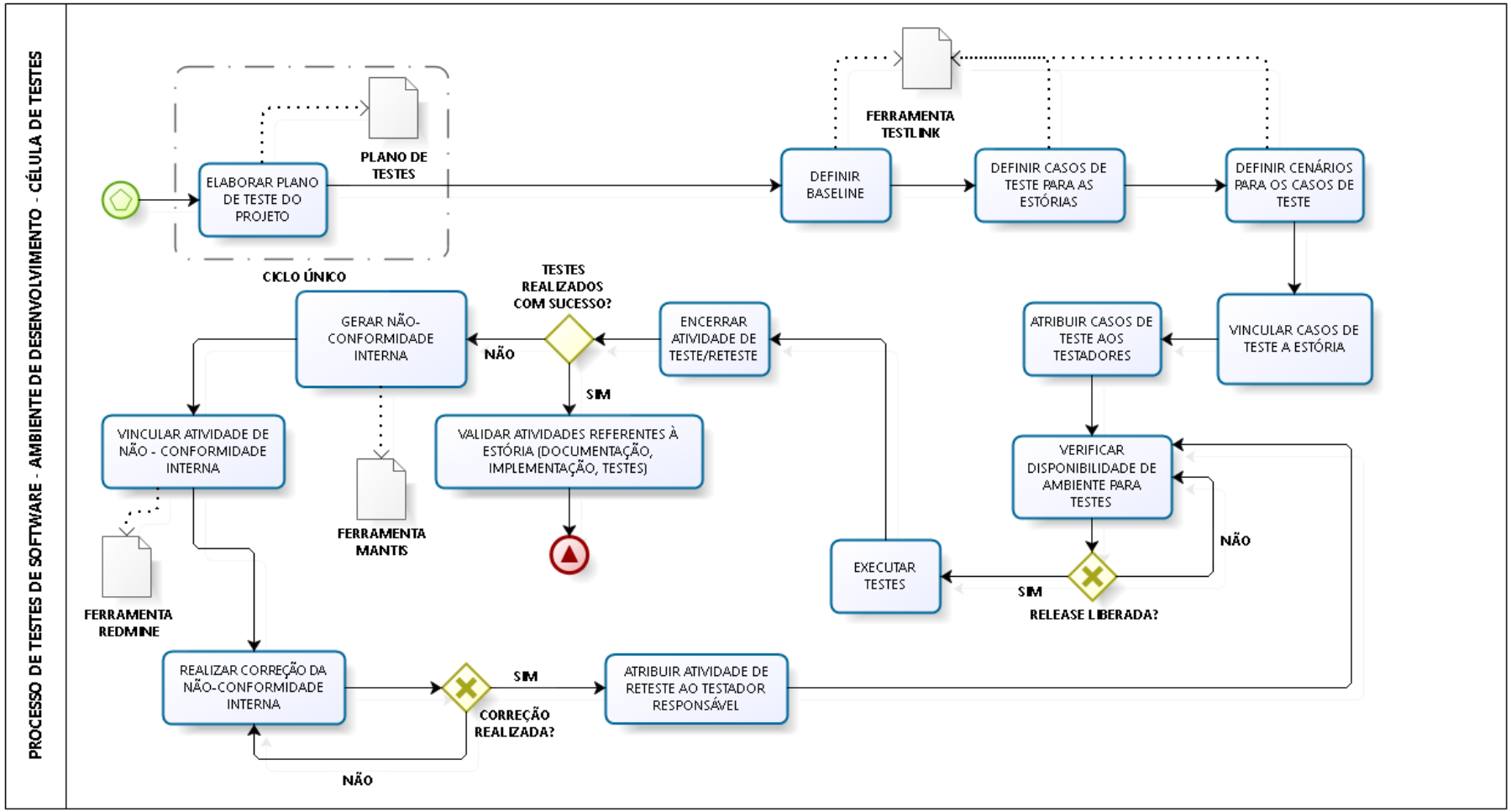
O centro desta pesquisa são as práticas realizadas em uma organização de grande porte, administradora de recursos de Telecom, fundada em 1993 logo após o início da telefonia celular no Brasil, a sede da empresa está situada na cidade de Anápolis – GO, a mesma é voltada para geração de negócios entre entidades e associados, de todo Brasil, integrada com as maiores operadoras de telefonia do Brasil.

A empresa utiliza de recursos que possibilita entidades associativas total segurança na relação com seus associados devido ao alto controle de informação, gestão e integração, através de suas tecnologias de informação. Atualmente a empresa conta em seu quadro com mais de trinta colaboradores e cinco departamentos, e um destes departamentos é o departamento de tecnologia da informação, onde são alocadas as equipes de análise de sistemas, desenvolvedores de *software*, testadores de *software*, analistas de infraestrutura, e gerente de projeto.

A organização possui uma equipe de teste que realiza testes de *softwares* na etapa de desenvolvimento ou em sistemas já em produção.

A Figura 6 apresenta o processo atual que é divido em dois ciclos: **Planejamento e Execução.**

Figura 6 - Processo de teste de *software* atual da organização



1. **Planejamento**

No ciclo de Planejamento é elaborado o plano de teste, por qualquer integrante da equipe de teste. Está etapa ocorre no momento em que o time de especificação de requisitos repassa as histórias de usuários para a equipe de teste e informa-os verbalmente que os requisitos descritos nas histórias de usuários serão desenvolvidos e integrados ao sistema.

2. **Execução**

Neste ciclo, são elaborados os casos de testes, utilizando a ferramenta *Testlink*. Cada caso de teste é elaborado com base nas histórias de usuários desenvolvidas pelos analistas de requisitos, e segue o modelo de um *checklist* do que dever ser testado após a entrega de um requisito implementado por um desenvolvedor.

Em um segundo momento, após a implementação de um requisito, o mesmo é disponibilizado em um ambiente de teste idêntico ao ambiente de produção onde são realizados os testes funcionais ou também conhecidos como os testes de caixa preta. Neste estágio o analista de teste verifica se existem, erros, defeitos ou falhas com base nos casos de testes elaborados, caso algum caso de teste não seja executado com sucesso, o erro é reportado imediatamente para o desenvolvedor através da ferramenta *Mantis Bug Tracker* para que os mesmo sejam tratados e os analista de teste possam realizar um Reteste.

Logo, se espera que os casos de testes sejam executados com sucesso, para que então este novo requisito seja integrado com outras partes do sistema, e seja disponibilizado a atualização para os usuários.

3.2Problema

Para a realização desta pesquisa utilizou-se como base o projeto SGC2, que é um sistema para gerenciamento de celulares, que contém módulos operacionais, financeiros, cadastro de associados e o processo de teste de *software* citado na sessão anterior.

Este projeto é um sistema web desenvolvido na linguagem *Ruby* com *framerwork Ruby on Rails*, e tem como banco de dados o *Postegre SQL*, o mesmo é um sistema focado em serviços Telecom responsável por cuidar de todas as atividades diárias da empresa, desde o administrativo ao operacional. Atualmente o projeto está em constante

desenvolvimento e conta com um time de nove integrantes sendo dois deles analista de requisitos, dois analistas de testes, dois analistas de infraestrutura três, desenvolvedores e um Gerente de Projetos.

Para avaliar o nível de maturidade do processo atual, e identificar os problemas foram necessários realizar dois passos, o primeiro foi identificar elaborar um questionário para nível de maturidade do processo de teste atual, através das práticas de testes aplicadas no projeto, o segundo passo foi avaliar quais as práticas do primeiro nível que não são aplicadas, para então ser levantado os problemas, e logo propor um processo contemplando as práticas especificas.

- **Primeiro Passo – Elaboração de questionário para avaliar o nível de maturidade de um Processo de Teste de software em relação ao primeiro nível do modelo MPT.BR**

O modelo MPT.BR não possui um modelo de avaliação particular, porém o método proposto neste trabalho, utilizou- se de práticas utilizada no modelo MPS.BR no qual possui graus de implementação, porém para esta pesquisa foram adotados apenas três graus de implementação, sendo: Implementado, Parcialmente Implementado e não implementado. No qual espera-se que todas as práticas especificas sejam implementadas para que então o processo de teste de *software* comtemple o primeiro nível de maturidade do modelo MPT.BR.

Logo para avaliação do nível de maturidade foi aplicado um questionário com os integrantes do time de teste de *software* apresentado no **apêndice A** ele é um documento baseado nas vinte e quatro práticas específicas do primeiro nível do modelo de maturidade MPT.BR, que auxilia as empresas avaliarem o seu nível de maturidade em relação ao primeiro nível do modelo MPT.BR.

O questionário conta com 24 perguntas de respostas: implementado, parcialmente implementado e não implementado, respondendo as 24 perguntas como implementado, a empresa é aderente ao primeiro nível de maturidade MPT.BR pois contempla as práticas especificas do primeiro nível do modelo, e já está sujeita a melhorias no segundo nível, caso contrário o processo de teste atual da mesma está sujeito a melhorias no primeiro nível, pois segundo o guia de implantação MPT.BR (2013) representa o patamar de maturidade de uma organização. Este nível contém o mínimo que uma organização precisa

para demonstrar que a disciplina de teste é aplicada nos projetos e que esta aplicação ocorre de forma planejada e controlada.

- **Segundo passo – Resultado da Avaliação do Processo de Teste de *Software* Atual da Organização e Identificação dos Problemas**

Os resultados foram analisados com base nas respostas do questionário, que foi aplicado com a equipe de teste, no qual se permitiu ser proposta uma nova abordagem no processo de teste de software da empresa, pois conforme os resultados do questionário no **Anexo A**, existem práticas especificas que não são aplicadas atualmente, o que permite melhorias para o processo de teste da organização.

Na avaliação anexada pode-se visualizar que as práticas específicas GPT1, GPT3, GPT4, GPT5, GPT6, GPT7, GPT8, GPT9, GPT10, GPT13, GPT15 e GPT18 não são realizadas atualmente, tal informação evidencia que a organização não possui o mínimo necessário para aplicar os processos de teste de maneira planejada e controlada, pois segundo o guia de implantação do modelo MPT.BR (2013), o nível 1 denominado parcialmente gerenciado contém as mínimas práticas especificas e genéricas que uma organização precisa possuir para executar as atividades de teste de maneira correta.

Na Tabela 7 é possível verificar problemas da não execução das práticas especificas atualmente.

Tabela 7- Problemas no processo por não executar as práticas específicas do primeiro nível

Práticas específicas	Problemas por não executar práticas específicas
GPT1 – Realizar análise de risco do produto	O processo de identificar as áreas mais críticas do *software* que necessitam de um teste mais detalhado é chamado de análise de risco do produto, geralmente uma organização não dispõe de recursos adequados para realizar testes exaustivos que contemplem todas as possibilidades contidas em uma versão do *software*, Desta forma, é preciso direcionar o teste para áreas mais críticas do *software*, buscando maximizar a sua efetividade.
GPT3 – Definir estratégia de teste	A estratégia de teste consiste na declaração, de alto nível, sobre como o teste será implementado e quais níveis serão abordados no âmbito do projeto, Deve abordar o que será testado, como o teste será realizado, onde e quando os testes serão executados, A estratégia de teste deve ser fundamentada nos objetivos de teste e na análise de risco realizada no produto, a não realização da mesma poderá ocasionar problemas nos testes, no que impactará diretamente no custo e cronograma do projeto de teste.

GPT4 – Definir o escopo do trabalho para o projeto de teste	A EAP é uma estrutura orientada ao produto que evolui com o projeto a não execução desta prática específica, permite que unidades lógicas de trabalho a serem gerenciadas, denominadas pacotes de trabalho não sejam identificadas no qual dificulta na hora de elaborar o projeto de teste, pois atribuir esforço, prazo e responsabilidades é necessário.
GPT5 – Estabelecer estimativas de tamanho	O tamanho é o insumo principal para estimativas de esforço, custo e prazo não estabelecendo, não será possível estimar esforço, custo e prazo.
GPT6 – Definir o ciclo de vida do projeto de teste	Normalmente as fases do ciclo de vida do projeto são definidas para apoiar pontos de decisão, nos quais são assumidos compromissos importantes sobre recursos e abordagem técnica. O entendimento do ciclo de vida do projeto é crucial para determinar o escopo da atividade de planejamento, o momento de planejamento inicial e replanejamento, e os critérios para replanejamento (marcos críticos).
GPT7 – Estimar o esforço e o custo	A cultura de algumas organizações não permite que o gerente de projetos gerencie o custo real. Nestes casos a gestão do esforço pode ser considerada a gestão dos custos do projeto. Isto se aplica somente se o projeto não envolver outros custos como aquisição de material ou contratações, e estimar esforço e o custo com testes são necessário para estimar o mais próximo do real.
GPT8 – Estabelecer e manter o orçamento e o cronograma do projeto	Ao não estabelecer e controlar o cronograma do projeto ocorrerá problemas na hora de controlar os custos, e o prazo final do projeto, poiso orçamento e o cronograma baseiam-se nas estimativas de tamanho, esforço e custo do projeto e asseguram que a alocação de recursos orçamentários, complexidade das tarefas e suas interdependências sejam tratadas adequadamente.
GPT9 – Identificar riscos do projeto	Os riscos do projeto de teste devem ser identificados durante o planejamento do projeto, pois é necessário identificar, analisar e planejar resposta para os riscos do projeto de teste, não identificando não será possível analisar o impacto, probabilidade de ocorrência e prioridade de tratamento.
GPT10 – Planejar os recursos humanos	A obtenção de conhecimento para o projeto envolve tanto o treinamento do pessoal do projeto quanto a aquisição de conhecimento externo. Os requisitos para composição da equipe dependem das habilidades e conhecimento disponíveis para apoiar a execução do projeto.
GPT13 – Estabelecer indicadores de desempenho de teste	De acordo com a ISO/IEC 15939 [ISO01a], um indicador é uma medida que fornece uma estimativa ou avaliação de atributos específicos derivados de um modelo relativo a determinadas necessidades de informação. Os indicadores são a base para a análise e tomada de decisão, sem eles o analista de projeto poderá tomar decisões erradas, para o projeto.
GPT15 – Revisar e obter compromisso	Todas as informações contidas dentro do planejamento do projeto devem estar alinhadas com o plano de teste e apoiá-lo, A solução

com o Plano de Teste	dos conflitos e estabelecimento de compromissos é fundamental para que o projeto possa efetivamente contar com os recursos planejados, sem compromisso com o plano de teste não será possível atingir as metas definidas.
GPT18 – Executar revisões em marcos do projeto	As revisões de marco são delineadas durante o planejamento do projeto e geralmente são revisões formais documentadas considerando todos os aspectos do planejamento do projeto. Não realizando estas revisões não será possível ver realmente se pontos significativos do cronograma do projeto como estão sendo cumpridos como, por exemplo, a conclusão de fases selecionadas.

Como demonstra a Tabela 7 pode-se perceber que a não execução das práticas específicas problemas ocorrerão no projeto, o que pode levar ao fracasso do projeto, na próxima sessão é sugerido um novo processo no qual os integrantes do time de testes deverão executar as práticas específicas do primeiro nível, esperando-se que eleve o nível de maturidade do time de teste.

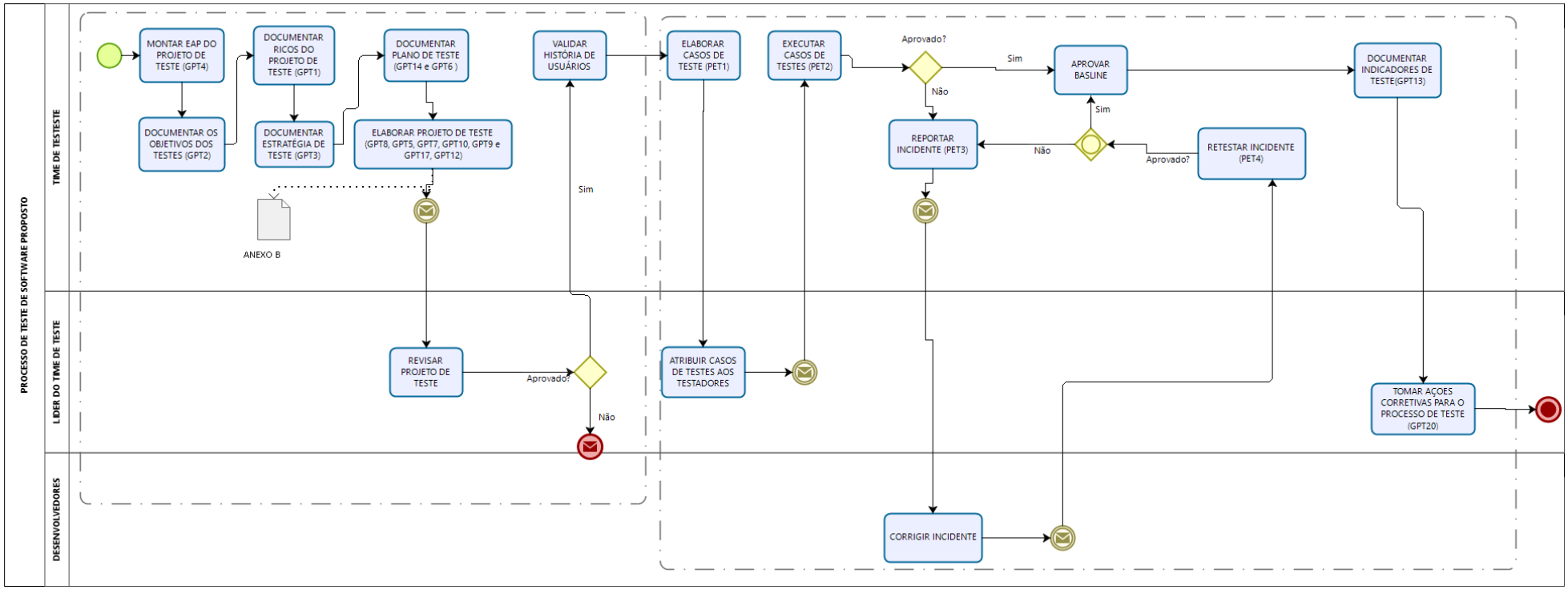
3.3 Proposta de Novo Processo

A organização possui uma cultura de evoluir constantemente a maturidade de seus processos, principalmente quando se trata de processos internos de produção. Atualmente para melhoria dos processos internos da empresa, utiliza-se o modelo MPS.BR como padrão de elevação de maturidade, devido à esta constante busca de aprimoramento, sentiu-se a necessidade de elevar o nível de maturidade nos processos relacionados aos testes de *softwares*, pois quanto mais alto o nível maturidade, melhor é o controle e gestão dos projetos de teste, como por exemplo, a otimização das estimativas, que tendem a ser mais confiáveis e os resultados podem ser mais perto do que é esperado.

Então, optou-se em utilizar o MPT.BR como referência, para melhorias dos processos de teste, por ser um modelo brasileiro baseado no MPS.BR e adepto as áreas de teste de *softwares*, pode ser aplicado em pequenas e grandes empresas. A semelhança das áreas do modelo com as áreas de processos usadas no MPS.BR, torna mais fácil de serem implementadas, pois já faz parte da cultura da organização.

Uma das grandes vantagens de utilizar o modelo MPT.BR é demonstração de resultados a curto prazo, na figura a seguir demonstra a proposta do novo processo de teste baseado no primeiro nível da maturidade do MPT.BR, no qual é dividido nas fases de Planejamento, Execução e Entrega e para os papéis de time de teste e líder de teste. A Figura 7, abaixo, apresenta tais informações.

Figura 7 - Processo de teste de *software* sugerido pelos autores da pesquisa



Fonte: Os Autores (2019)

Na fase de planejamento, qualquer integrante do time de teste pode elaborar a estrutura analítica do projeto de teste, comtemplando a prática específica GPT4 do modelo MPT.BR. Em seguida, este integrante deverá definir os objetivos do teste, de forma a contemplar a prática especifica GPT2 do modelo MPT.BR. Logo, os artefatos de definição de riscos do projeto de teste, estratégia de teste e plano de teste devem ser definidos conforme o ciclo de vida do projeto. Assim, serão satisfeitas mais quatro práticas: GPT1, GPT3, GPT14 e GPT6 para então elaborar o artefato: projeto de teste.

No projeto de teste, deverá conter os seguintes itens: cronograma do projeto, estimativa de tamanho do projeto, esforço e custo para realizar os testes do projeto, orçamento e controle do orçamento durante a execução do projeto, planejamento de recursos humanos necessários, riscos para execução do projeto de testes e todos *stakeholders.* Assim, serão satisfeitas as práticas GPT8, GPT5, GPT7, GPT10, GPT9 e GPT17 conforme o **Apêndice B.**

Com o projeto de teste elaborado, o integrante do time de teste deverá encaminhar para o líder de teste, pois o mesmo será responsável pela aprovação, sendo aprovado o projeto de teste, o time deverá avaliar as histórias de usuários pertencentes ao projeto, encaminhadas pelos Analistas de requisitos e então deverão ser elaborados os casos de testes, caso contrário o projeto de teste não é executado e o líder deverá dar *feedback* para o time.

Com os casos de testes elaborados, o líder do time deve atribuir os casos de testes ao time, cada integrante do time será responsável pela execução dos seus respectivos casos de teste. Os casos de testes deveram ser executados sem um ambiente sistêmico igual ao ambiente de produção, para que os testes fiquem o mais próximo possível do real satisfazendo a prática específica GPT11.

Logo, se os casos de testes forem executados com sucesso, o integrante deverá homologar o requisito, que está sendo testado, para que o mesmo ser integrado ao sistema principal. É necessário documentar todos indicadores de testes, para que seja possível tomar ações corretivas junto ao processo, nos próximos projetos de testes a serem executados. Caso contrário, se houver casos de testes que não foram executados, o responsável pelo o mesmo, deverá relatar e controlar os erros, defeitos ou falhas encontrados no requisito, até que estes sejam corrigidos e então o requisito seja homologado para entrar em produção, no qual satisfaz as práticas GPT13, GPT20, GPT19.

O ciclo deste processo é único e deve acontecer sempre que um novo projeto de teste for iniciado, durante a execução do projeto de teste, os integrantes do time de teste deverão ter compromisso com o plano de teste, o projeto de teste deverá ser monitorado pelos os integrantes do time e todos os artefatos de cada projeto deverão ser evidenciados durante toda a execução do projeto, assim satisfazendo as práticas GPT12, GPT15, GPT16 GPT18.

O processo descrito nessa sessão, foi sugerido na pesquisa como melhorias do processo de teste já existente, no qual satisfaz as 20 práticas especificas do modelo primeiro nível do modelo MPT.BR, juntamente com as PET1, PET2, PET3 e PET4 que são as práticas especificas para o projeto e execução de teste e as PG1, PG2, PG3, PG4, PG5 e PG6, que são práticas genéricas que ocorrem durante a execução dos testes no processo, o mesmo foi elaborado pelos autores da pesquisa e executado pelo time de teste da organização em duas Sprint como teste para obtenção de resultados.

3.4Execução do Processo Propostos

Após a sugestão do processo, foi ministrado um treinamento de 6 horas com a equipe de teste, no qual os autores da pesquisa explicaram o fluxo do processo para os integrantes do time de teste, para então os integrantes do time de teste executarem o mesmo.

O processo foi executado em duas *Sprint* de 10 dias, a primeira que ocorreu entre o dia 1° de abril até o dia 12 de abril de 2019, e a 2º *Sprint* dia 15 de abril até o dia 29 de abril de 2019.

Cada item do *sprint backlog* contou com a implementação de um requisito funcional, no primeiro projeto foi o requisito de troca de equipamentos, no qual permite usuários do sistema realizarem a troca de equipamentos e registra-las dentro do sistema, e no segundo projeto foi o requisito de aquisição de linhas e equipamentos, no qual padroniza a maneira de realizar aquisições de equipamentos junto às operadoras.

No iniciar de cada *Sprint*, foi elaborado o projeto de teste, seguindo todas as práticas especificas do primeiro nível do modelo, não houve dificuldades na execução do processo, pois não houve muito impacto de como são executadas as práticas especificas de projeto de execução de testes, o que ocorreu foi uma mudança de como os testes são tratados, antes da intervenção os testes não eram tratados como um projeto.

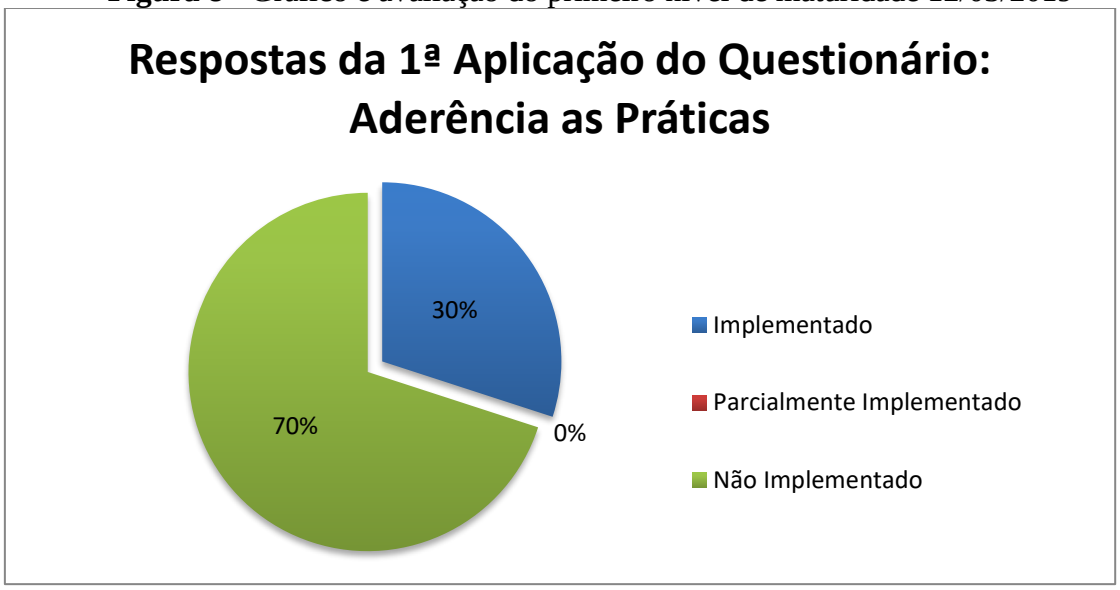
O que acabava impactando diretamente, no mal controle das atividades e também não tinha uma visão de quanto as atividades de teste de *software* custavam para a empresa, o que impacta diretamente no projeto de desenvolvimento que é o projeto principal da organização.

Assim, observa-se, que executar os testes de maneira planejada e controlada diminuirá os custos, permitirá ao líder do projeto de teste controlar as atividades de todo o time de maneira mais eficaz. O resultado provável será redução dos custos do projeto principal, trazendo mais retorno financeiro para a organização.

3.5Considerações Finais

Após a intervenção nas duas *Sprint*, há evidências de que o time de teste obteve uma melhora significativa em relação a sua aderência ao primeiro nível de maturidade. Conforme o questionário em anexo foi gerado um gráfico que demonstra uma aderência aproximada ao primeiro nível de maturidade do MPT.BR de 42% no dia 12/03/2019.

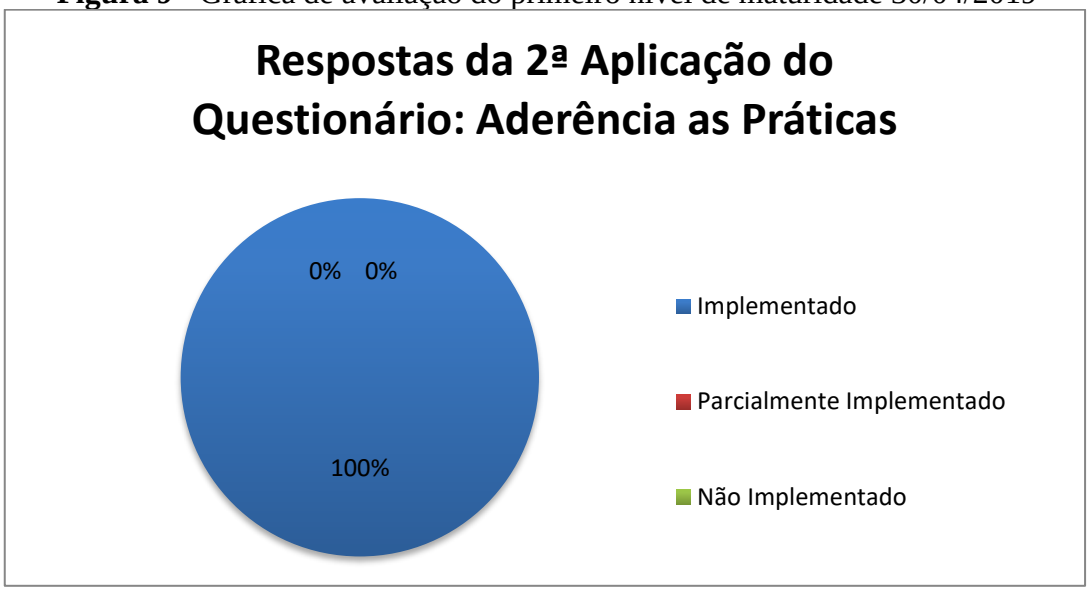
Figura 8 - Gráfico e avaliação do primeiro nível de maturidade 12/03/2019



Fonte: Os Autores (2019)

Logo após a intervenção das duas *Sprint* foi aplicado o mesmo questionário. A partir dos dados obtidos, foi gerado um gráfico que demonstra uma aderência aproximada ao primeiro nível de maturidade do MPT.BR de 100% no dia 30/04/2019.

Figura 9 - Gráfica de avaliação do primeiro nível de maturidade 30/04/2019



Fonte: Os Autores (2019)

Através da Figura 10, que apresenta um resumo das repostas, pode-se perceber que o time de teste obteve uma melhora significativa na sua maturidade atual e que a organização pode aprimorar seus processos se seguir as práticas do primeiro nível.

Na tabela 8, detalha alguns dos principais benefícios observados após a execução das práticas do primeiro nível de maturidade do modelo MPT.BR.

Tabela 8–Benefícios observados ao executar as práticas do primeiro nível.

Práticas específicas	**Benefício**
GPT1 – Realizar análise de risco do produto	Houve a identificação de partes críticas do requisito, no qual os testadores puderam elaborar com mais precisão casos de testes, que cobrissem os cenários críticos.
GPT2 – Estabelecer objetivos do teste	Ao definir os objetivos, permitiu que os testadores executassem suas atividades com mais agilidade e foco no dia a dia do projeto.
GPT3 – Definir estratégia de teste	Com base nos objetivos e riscos levantados, foi possível definir a melhor estratégia de teste para execução do projeto.
GPT7 – Estimar o esforço e o custo	Com está pratica os testadores mensuraram o tamanho de suas atividades e conseguiram estimar o tempo necessário para realização das suas atividades, no qual facilitou para o gerente de projeto gerenciar as atividades de teste, e passou a ter com mais precisão o quanto é gasto com os testes durante o projeto.
GPT8 – Estabelecer e manter o orçamento e o cronograma do projeto	Com o cronograma estabelecido e mantido, as atividades puderem ser entregues no tempo realmente estimado e evitou custo desnecessários com o teste.

GPT14 – Estabelecer o Plano de Teste	Foi gerado um documento onde centralizava todas as informações referentes ao projeto de teste, no qual norteou os testadores durante a execução do projeto de teste.

Fonte: Os Autores (2019).

As práticas sugeridas pelo primeiro nível são os requisitos mínimos para se ter um projeto de teste totalmente gerenciável e controlado. Seguir as práticas específicas é de extrema importância, pois cada uma delas traz benefícios para a organização e os interessados, no qual permite a disciplina de teste da organização ser executada de maneira mais eficiente, o que auxilia na a qualidade do produto final.

4. CONCLUSÃO

Quando se trata de melhoria de maturidade em processo de teste de *software* brasileiro, o modelo MPT.BR, é um ótimo modelo a ser seguido, que se adapta bem a áreas de teste de *software*, tanto de pequenas, quanto grandes empresas, em curto prazo e pouco custo.

O estudo também se permitiu visualizar a importância de evoluir os processos de testes dos *software*s tendo como base o modelo, visto que foram colhidas evidência que uma empresa que não cumprem práticas especificas do primeiro nível, podem possuir lacunas em seu processo de teste de *software*, Assim pode gerar custo desnecessários e comprometimento na qualidade dos produtos oferecidos aos clientes.

Após propor um novo processo de teste de *software* que contempla todas as práticas especificas do primeiro nível do modelo, comprovou-se que aplicar as práticas especifica do modelo, a maturidade do processo de *software* é aumentada. Dessa forma diminui significativamente problemas no andamento dos projetos, tanto para equipe de teste quanto para equipe de desenvolvimento, pois as atividades passam a serem melhores controladas, no que permite um líder de teste ou um gerente, que gerencie o time com êxito em atividades diárias.

O trabalho demonstra que uma empresa que não possui cultura de teste, ou uma empresa que executa as atividades de testes de maneira incorreta, e que querem investir em testes, começar pelo primeiro nível do modelo MPT.BR é vantajoso, pois oferece resultados a custo prazo e se adequa facilmente aos ambientes de teste de *software*.

Tendo os resultados deste estudo como parâmetro, para trabalhos futuros sugere-se, um estudo comprovando a importância de evoluir o processo de teste de *software* tendo como base o segundo nível do modelo MPT.BR, O qual permitirá uma análise qualitativa, e quantitativa, de quais os benefícios que o modelo poderá trazer para uma equipe de teste de *software* no segundo nível.

5. REFERÊNCIAS

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR ISO 9000**: Sistemas de gestão da qualidade – Fundamentos e vocabulário. Rio de Janeiro: ABNT, 2005.

BARCAUI, A. **PMO**: escritório de projetos, programas e portfólio na prática. Rio de Janeiro: Brasport, 2012.

BARTIÉ, A.**Garantia da qualidade de *software*:** adquirindo maturidade organizacional**.** São Paulo: Campus, 2002.

BATISTA, F. de M., **O que é mesmo Qualidade? Blog da Qualidade**. 2014. Disponível em: http://web.archive.org/web/20170506140818/http://www.blogdaqualidade.com.br/oque-e-qualidade/. Acesso em: 09 out. 2018.

BECK, K. et al. **Agile Manifesto**. 2017. Disponível em: http://agilemanifesto.org/. Acesso em: 01 maio 2019.

CÉRGOLI, R. **Uma análise da aplicação de testes no desenvolvimento de um sistema erp**. 2017. Disponível em: https://spo.ifsp.edu.br/images/phocadownload/DOCUMENTOS_MENU_LATERAL_FIXO/POS_GRADUA%C3%87%C3%83O/ESPECIALIZA%C3%87%C3%83O/Gest%C3%A3o_da_Tecnologia_da_Informa%C3%A7%C3%A3o/PRODUCAO/2017/UmaC3%A7%C3%A3o_de_Testes_no_Desenvolvimento_de_um_Sistema_ERP.pdf. Acesso em: 09 out. 2018.

COSTA, D. O. **Avaliação de processos de teste pelo modelo de maturidade TMMi em pequenas empresas.** 2016. Disponível em: http://www.inf.ufg.br/mestrado/sites/www.inf.ufg.br.mestrado/files/uploads/Dissertacoes/dissertacao_Daniella_Oliveir_Costa_22082016.pdf. Acesso em: 21 out. 2018.

ELIZA, R.; LAGARES, V. **Processo de teste de *software***. 2012. Disponível em: https://www.devmedia.com.br/processo-de-teste-de-software/23795. Acesso em: 26 out. 2018.

GIL, A. C. **Como elaborar projetos de pesquisa**. São Paulo, v. 5, n. 61, p. 16-17, 2002.

HETZEL, W. **Guia completo ao teste de *software***. Rio de Janeiro: Campus, 1987.

HUMBLE, J.; FEARLEY, D. **Entrega Contínua**: como entregar *software* de forma rápida e confiável. São Paulo: Bookman, 2014.

ISD, Instituto Santos Dumont Brasil. **O que é CMMI?**. 2011. Disponível em: http://www.isdbrasil.com.br/o-que-e-cmmi.php. Acesso em: 12 out. 2018.

MPT.BR. Melhora do Processo de Teste Brasil. **Estrutura do modelo**. 2016. Disponível em: http://mpt.org.br/mpt/mpt/estrutura-do-modelo/. Acesso em: 29 out. 2018.

MPT.BR. Melhora do Processo de Teste Brasil. **Guia de referência**. 2013. Disponível em: http://mpt.org.br/mpt/wp-content/uploads/2013/05/MPT_Guia_de_referencia.pdf. Acesso em: 03 maio 2019.

MYERS, G. J. **A arte de testar*software***. 2. ed. John Wiley, 2004.

NARCIZO, B. T. **Diretrizes para implementação do modelo MPT.BR a partir do MR-MPS-SW**. 2015. Disponível em: http://monografias.poli.ufrj.br/monografias/monopoli10014200.pdf. Acesso em: 22 out. 2018.

PRESSMAN, R. A. **Engenharia de *Software***. 6. ed. São Paulo: McGraw-Hill, 2009.

PRESSMAN, R. A. **Engenharia de *Software***: uma abordagem profissional. 8. ed. São Paulo:McGraw Hill, 2016.

QUINTELLA, H. L. M. M.; ROCHA, H. M.; MOTTA, W. **Avaliação do nível de maturidade dos processos de desenvolvimento de produtos na indústria automotiva do sul fluminense com base nos critérios do CMMI**. 2005. Disponível em: http://www.producao.uff.br/conteudo/rpep/volume52005/RelPesq_V5_2005_13.pdf. Acesso em: 11 out. 2018.

SCHWABER, K. **Agile Project Management with Scrum**. Redmond, Washington: Microsoft Press, 2004.

SCHWABER, K; SUTHERLAND, J. **The Scrum Guide**: The Definitive Guide to Scrum: The Rules of the Game. 2013. Disponível em: https://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-us.pdf. Acesso em: 01 maio. 2019.

SCRUM. Disponível em:http://www.desenvolvimentoagil.com.br/scrum/. Acessoem: 01 maio 2019.

SINFIC, Sistemas de Informação Industriais e Consultoria. **O valor da qualidade de *software***. 2006. Disponível em: http://www.sinfic.pt/SinficWeb/displayconteudo.do2?numero=24417. Acesso em: 12 out. 2018.

SOFTEX, **Associação para Promoção da Excelência do *Software* Brasileiro**. 2012. Disponível em: https://www.softex.br/. Acesso em: 12 out. 2018.

SOMMERVILLE, I. **Engenharia de *software***. 2007. Disponível em: https://d2sdownloads.blogspot.com/2017/11/download-sommerville-engenharia-de.html. Acesso em: 05 out. 2018.

SOUZA, L. M. **Organização de estrutura de testes em um ambiente coorporativo de TI**. 2017. Disponível em: https://riuni.unisul.br/bitstream/handle/12345/2728/LEONARDO_ MAYER_DE_SOUZA-%5B44747-11301-1-633330%5DLEONARDO_MAYER_DE _SOUZA-ARTIGO_FINAL _PDF.pdf?sequence=1&isAllowed=y. Acesso em: 06 out. 2018.

TMMI. TMMi**Assessment Method Application Requirements**. Foundation TMMi, Ireland, 2009.

TRIPP, D. Pesquisa-ação: uma introdução metodológica. **Educação e pesquisa**, v. 31, n. 3, 2005.

VASCONCELOS, Alexandre Marcos Lins de., [et. al]. **Engenharia de Software para Software Livre 1**. Lavras: UFLA, 2006.

VILAS BOAS, A. L. de C. **Qualidade e Avaliação de Produto de *Software***. Ed. Lavras/MG: FAEPE/UFLA, 2007.

VILELA JR., D. C. **Processo de teste de *software***. 2012. Disponível em: http://dvillela.dlinkddns.com:4000/?p=448. Acesso em: 19 out. 2018.

WAZLAWICK, R. **Engenharia de *Software***: conceitos e práticas. São Paulo: Elsevier, 2012.

6. APÊNDICES

6.1. Apêndice A

Este questionário foi utilizado para identificar aderência do processo de teste ao Nível 1 do MPT.BR.

Avaliação Nível 1 - MPT. BR		
Questionamentos relacionados à Gerência de Projetos de Teste (GPT)	**1º Avaliação**	**2º Avaliação**
	12-03-2019	30-04-2019
GPT1 – é realizado análise de risco do produto antes de começar os testes?	Não Implementado	*Implementado*
GPT2 – Os objetivos do testes São estabelecidos?	*Implementado*	*Implementado*
GPT3 – Estratégia de testes são definidas antes de começar os testes?	Não Implementado	*Implementado*
GPT4 – Escopo do trabalho para o projeto de teste é definido?	Não Implementado	*Implementado*
GPT5 – Estimativas de tamanho é definido antes de começar os testes?	Não Implementado	*Implementado*
GPT6 – O ciclo de vida do projeto de teste é definido?	Não Implementado	*Implementado*
GPT7 – Esforço e o custos de testes são estimados?	Não Implementado	*Implementado*
GPT8 – Orçamento e o cronograma do projeto é definido e mantido?	Não Implementado	*Implementado*
GPT9 – Os Riscos do Projeto são identificados?	Não Implementado	*Implementado*
GPT10 – Os recursos humanos são planejados?	Não Implementado	*Implementado*
GPT11 – O ambiente de teste para o projeto é planejado?	*Implementado*	*Implementado*
GPT12 – Existem artefatos e dados dos projetos planejados?	*Implementado*	*Implementado*
GPT13 – Indicadores de desempenho de teste são planejados?	Não Implementado	*Implementado*
GPT14 – O Plano de Teste é estabelecido?	*Implementado*	*Implementado*
GPT15 – Obtém-se compromisso com o Plano de Teste?	Não Implementado	*Implementado*
GPT16 – O projeto de teste é monitorado?	Não Implementado	*Implementado*
GPT17 – Gerenciar o envolvimento dos stakeholders?	*Implementado*	*Implementado*
GPT18 – Executa revisões em marcos do projeto?	Não Implementado	*Implementado*
GPT19 – Analisa e registra os problemas identificados?	*Implementado*	*Implementado*
GPT20 – Estabelece e acompanhar ações corretivas até a sua conclusão?	Não Implementado	*Implementado*

Questionamentos relacionados à Projeto e Execução de Teste (PET)	1º Avaliação	2º Avaliação
	12-03-2019	30-04-2019
PET1 – Os Casos de Testes são identificados?	*Implementado*	*Implementado*
PET2 – Os Casos de Testes são executados?	*Implementado*	*Implementado*
PET3 – Os Incidentes são reportados?	*Implementado*	*Implementado*
PET4 – Os Incidentes são acompanhados?	*Implementado*	*Implementado*

Tipos de respostas para este questionário
Implementado
Parcialmente Implementado
Não Implementado

6.2. Apêndice B

Modelo do documento utilizado durante a intervenção do processo proposto.

CONTROLE DO DOCUMENTO

Data	Responsável	Atividades

INTRODUÇÃO

1.1. PROPÓSITO DO DOCUMENTO

Plano de teste referente ao SGC2 O plano tem como objetivos:

- Definir estratégias de teste para o projeto;
- Detalhar os requisitos que serão testados;
- Identificar o ambiente de testes, bem como os recursos necessários;
- Prover estimativas para esforço de teste;
- Listar os elementos entregáveis do projeto de teste;

1.2. ESCOPO

O escopo do projeto está disponível na ferramenta Redmine através do link:

1.2.1 ETAPAS DE TESTE NO PROJETO DE TESTE

Neste projeto, serão executados testes funcionais, de stress, carga, unidade, integração e aceitação/regressão para garantia da conformidade entre o produto final e os requisitos descritos. Também deve garantir a utilização do sistema nos navegadores mais utilizados no mercado.

1.2.2 FUNCIONALIDADES A SEREM TESTADAS

Conforme cronograma seguem as user stories do projeto que serão testadas:

1.	Login
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	
11.	

1.2. REFERÊNCIAS E DOCUMENTOS DO PROJETO

Segue tabela que identifica toda a documentação existente que foi disponibilizada para a formação deste documento na versão 1.

Documento	Existente no projeto (Sim/Não)	Recebido pela equipe de testes (Sim/Não)	Autor

USER STORIES DOS TESTES

As user stories que serão testadas estarão disponíveis na ferramenta TestLink. As informações referentes à utilização da ferramenta estão contidas no documento EFTM_001.

ABORDAGEM DE TESTES

As abordagens de testes adotadas no projeto são descritas nas próximas seções. Nesta seção é apresentada a metodologia recomendada para o teste da aplicação, descrevendo o modo como será testado. As principais considerações na estratégia de teste são as técnicas a serem utilizadas e o critério para determinar quando o teste está completo.

3.1. TIPOS DE TESTE

As user stories do projeto passarão por testes funcionais, não funcionais e de requisitos. Alguns testes não funcionais não especificados serão executados e mencionados neste documento.

3.1.1 TESTE FUNCIONAL

Objetivo do Teste:	Garantia dos requisitos funcionais, ou seja, derivar casos de testes com base em análise da especificação da funcionalidade de um componente ou sistema sem fazer referência à sua estrutura interna.

Técnica:	▪ Executar as funções da userstory (inserir, editar...) considerando fatores como obrigatoriedade de campos, tamanho dos campos, para verificar a conformidade desses requisitos com a implementação. ▪ Criar ou modificar os testes para cada formulário para verificar a navegação e os estados de objeto apropriados para cada janela e objetos da aplicação.
Critério de Finalização:	Todas as validações, funcionalidades e resultados obtidos em conformidade com os resultados esperados. É verificado que cada formulário permanece consistente com a versão de comparação ou dentro de padrões aceitáveis.
Considerações Especiais:	▪ Deve haver acesso à aplicação no ambiente de produção.

3.1.2 TESTE NÃO-FUNCIONAL

Objetivo do Teste:	Garantia dos requisitos não-funcionais, ou seja, derivar casos de testes com base em análise da especificação de fatores ambientais e físicos que a aplicação deve atender. O não funcionamento desses requisitos pode impactar gravemente o funcionamento dos requisitos funcionais.
Técnica:	▪ Teste de compatibilidade: Executar a aplicação nos browsers mais utilizados no mercado, são eles: Internet Explorer, Mozilla Firefox, Google Chrome, Safari e Opera. ▪ Os testes de unidades do SGC 2.0, e as API serão realizados pelos desenvolvedores e estão disponíveis na plataforma Wercker e/ou Jenkins. ▪ Verificar o tempo de execução da aplicação, se apresenta lentidão, testando o software em situações limites, exigindo recursos em quantidade, volume ou frequência anormais. ▪ Simular e verificar a facilidade de uso do sistema.
Critério de Finalização:	Todas as validações, funcionalidades e resultados obtidos em conformidade com os resultados esperados.
Considerações Especiais:	▪ Deve haver acesso à aplicação no ambiente de produção.

3.1.3 TESTE DE REQUISITOS

Objetivo do Teste:	Garantir que os requisitos aprovados sejam testados e seus resultados apurados e conformes.
Técnica:	Executar cada teste no cenário principal nos cenários alternativos, levando em consideração os requisitos aprovados, baseando em dados válidos e inválidos, afim de se apurar: ▪ Os resultados esperados ocorrem quando dados válidos são usados. ▪ As mensagens apropriadas são exibidas quando dados inválidos são usados (mensagens de erro/ alerta são comuns ao processo, desde que haja opção de correção e que estejam no mesmo idioma dos usuários). ▪ Cada regra de negócio é aplicada de acordo com o cenário e tipo de requisito abordado.
Critério de Finalização:	▪ Todos os testes planejados foram executados. ▪ Todos os defeitos identificados foram tratados.
Considerações Especiais:	Não se aplica.

3.2. CRITÉRIOS DE PRONTIDÃO

É necessário para o início dos testes a disponibilidade da aplicação em ambiente de produção, com código atualizado.

3.3. CRITÉRIOS DE COMPLETEZA E SUCESSO

Os testes serão considerados como terminados quando todos os casos de teste previstos forem executados com sucesso, os objetivos de qualidade, o grau de confiabilidade exigido foi atingido e estiver em conformidade com cada requisito definido nas user stories.

3.4. CRITÉRIOS DE SUSPENSÃO E RETOMADA

Os testes podem ser suspensos somente sob pedido da coordenação e da gerência de TI, ou ocorrência de algum fator grave de interrupção dos testes, como por exemplo, a

inoperabilidade do servidor da aplicação. Após o reestabelecimento dos critérios de prontidão e ambiente para testes, estes devem ser reaplicados.

3.5. FERRAMENTAS

Tarefa	Ferramenta	Fabricante	Versão
Criar planos de teste	Testlink	TeamST	1.9
Acompanhar cronograma dos testes	Microsoft Project	Microsoft	2016
Acompanhar resultados	Mantis Bug Tracker / Redmine	Mantis Bug Tracker / Redmine	1.3.0
Reportar defeitos	Mantis Bug Tracker / Redmine	Mantis Bug Tracker / Redmine	1.3.0
Manter defeitos	Mantis Bug Tracker / Redmine	Mantis Bug Tracker / Redmine	1.3.0
Acompanhar defeitos encontrados	Mantis Bug Tracker / Redmine	Mantis Bug Tracker / Redmine	1.3.0
Gerar indicadores	Microsoft Office Excel/ Mantis Bug Tracker / Redmine	Microsoft/ Mantis Bug Tracker / Redmine	2013 / 1.3.0

RECURSOS

4.1 RECURSOS HUMANOS

RECURSOS HUMANOS		
Profissional	**Horas trabalhadas/dia**	**Responsabilidades**
	8:00hrs	Analista de TI
	8:00hrs	Analista de Sistemas
	8:00hrs	Coordenador de TI
	8:00hrs	Gerente de TI

4.2. RECURSOS DE AMBIENTE

Recursos do Sistema	
Recurso	**Nome/Tipo**
Servidor de Banco de Dados	
▪ Rede ou Sub-rede	Local (interna)
▪ Nome do Servidor	Alfa
▪ Nome do Banco de Dados	Dependerá do projeto/release
Máquinas de Teste Client	Intel Core i3-3240, 4GB, Windows 8.1 Pro
▪ Requisitos de configuração especial	
Repositório de Defeitos	Mantis

▪ Rede ou Sub-rede	Mantis
▪ Nome do Servidor	Mantis
Test Development PC's	Intel Core i3-3240, 4GB, Ubuntu

PROGRAMAÇÃO DOS TESTES

O cronograma dostestes está disponível na ferramenta Redmine através dos links:

5.1. ABORDAGEM DOS TESTES

As abordagens utilizadas para as iterações do projeto são todas de testes operacionais nas quais serão aplicadas as técnicas de teste.

MILESTONES DO PROJETO EM TESTE

O cronograma completo do projeto será utilizado como controle de Milestones e está disponívelna ferramenta Redmine através dos links:

RISCOS E CONTINGÊNCIAS

Risco	Gravidade	Probabilidade de ocorrência	Impacto previsto	Plano de Mitigação	Responsável
Nível de conhecimento da equipe de teste	Media	Alta	Atraso: Execução e finalização dos testes	Treinamento da equipe	
Disponibilidade da equipe de teste	Alta	Media	Atraso: Execução e finalização dos testes	Planejamento e controle do cronograma de trabalho da equipe	
Ausência de funcionários	Alta	Media	Atraso: Execução e finalização dos testes	Existir dois funcionários capazes de executar uma determinada tarefa	
Problemas com infraestrutura. (Servidor,	Alta	Baixa	Atividades relacionadas suspensas	Rede: Monitoramento diário	

máquinas ou rede).				Máquinas: Manutenção bimestral	
Problemas com pessoal da gerência de configuração.	Alta	Baixa	Perder artefatos do projeto	Nomear responsável pela gerência de configuração	

ENTREGAS

Serão entregues como produtos de trabalho para o projeto em teste:

- Plano de teste;
- Casos de teste.