Uma Ferramenta de Autoavaliação para Melhoria do Processo de Teste de Software

Jheimes France Romualdo[1], Fernando Sambinelli[1]

[1]Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP) – Câmpus Hortolândia – São Paulo – SP – Brasil

jheimes.france@gmail.com,sambinelli@ifsp.edu.br

Abstract. *The objective of this work is to develop a tool with guidance and evaluation in an ongoing process in the software testing process. For the structuring of the tool such as the use as references, some models of software testing and software process enrolment exist in the literature and the PDCA cycle. Three case studies were carried out in firms specializing in software engineering in the Campinas region to evaluate the effectiveness of the proposal. The results show that the developed tool has the potential of effective application in the improvement of software testing in real project equipment, as well as guidance in the technical training of testers.*

Resumo. *O objetivo deste trabalho foi desenvolver uma ferramenta que oriente e avalie a melhoria contínua no processo do teste de software. Para estruturação da ferramenta foram utilizados como referências modelos de teste e de maturidade de processo de software existentes na literatura e o ciclo PDCA. Foram realizados três estudos de caso em empresas especializadas em engenharia de software na região de Campinas visando avaliar a efetividade da proposta. Os resultados obtidos apontam que a ferramenta desenvolvida possui potencial de aplicação efetiva na melhoria do processo de teste em equipes de projetos reais, bem como de orientação na formação técnica dos testadores.*

1. Introdução

Em meados dos anos 1960 e 1970, o termo "crise de *software*" passou a ser usado devido ao período de colapso que marcou a história da indústria. As dificuldades existentes no desenvolvimento chamaram à atenção, diante do aumento da demanda e da complexidade que os programas começavam a exigir, devido a arduidade do mercado em lidar com a falta de qualidade que os sistemas apresentavam com prazos e orçamentos estourados, requisitos não atendidos e inexistência de técnicas estabelecidas para o desenvolvimento. "A crise foi uma decorrência da imaturidade do mercado e dos profissionais da computação da época, pois vinha de um período onde o desenvolvimento do *software* não exigia requisitos e configurações complexas, sua utilização era, em média, limitada ao ambiente em que era desenvolvido "[Manzano 2016].

Diante dos desafios que o avanço da informação traz consigo, a Engenharia de *Software* teve seus princípios formados nesse tempo. Fritz Bauer propôs uma definição em uma conferência sobre o tema em 1968, como cita [Pressman 2011]: "[...] é o estabelecimento e o emprego de sólidos princípios de engenharia de modo a obter *software* de maneira econômica, que seja confiável e funcione de forma eficiente em máquinas reais".

Desta maneira, esta Engenharia propõe a utilização de uma metodologia que seja capaz de controlar todas as fases do processo de desenvolvimento e, assim, garantir qualidade, reconhecendo o *software* como produto e, ao mesmo tempo, um veículo que distribui o produto, transformando informações simples ou complexas, além de ser base para o controle do computador (sistemas operacionais, redes, ferramentas). Tendo isso, seu desenvolvimento necessita de critérios que busquem a qualidade, o custo adequado e os cumprimentos de prazos [Pressman 2011] .

Com a Engenharia de *Software* houve, uma resposta à insatisfação com o produto pouco confiável da época. "Passaram a ser adotadas técnicas formais de gerenciamento de qualidade do *software*, as quais foram desenvolvidas a partir dos métodos usados na indústria manufatureira. Essas técnicas de gerenciamento de qualidade em conjunto com novas tecnologias e melhores testes conduziram a melhoria significativa no nível geral de qualidade de *software*" [Sommerville 2011]. Perante o forte investimento, os improvisos efetuados pela imaturidade e o aumento da demanda, além da busca por qualidade no processo de desenvolvimento de *software* tornaram-se uma necessidade. "O clamor por maior qualidade de *software* começou realmente quando o *software* passou a se tornar cada vez mais integrado a todas as atividades de nossas vidas" [Pressman 2011]. Um sistema de qualidade é um produto que atende ao cliente de acordo com suas necessidades, que é executável de forma confiável e precisa. Para garantir a qualidade, seu desenvolvimento deve ser monitorado desde o início. [Pressman 2011] sugere que essa ação pode ser realizada através da verificação dos resultados de todas as atividades, medindo a qualidade, efetuando a verificação de erros antes da entrega, o que pode vir a resultar na redução do retrabalho, dos custos e do tempo da liberação do produto para entrar em produção.

Entretanto, alcançar a solução desses problemas não é uma tarefa simples, [Koscianski 2007] ressalta tais dificuldades ao abordar o assunto: "A volatilidade dos requisitos é uma das maiores causas de insucesso de projetos de *software*[...] " e além das características mutáveis do sistema, há também o trabalho intelectual das pessoas envolvidas no desenvolvimento.

De acordo com um estudo conduzido pelo *National Institute of Standards and Technology* (NIST) - Instituto Nacional de Padrões e Tecnologia - em 2002, os defeitos em *software* resultaram num custo anual de US$ 59,5 bilhões à economia dos Estados Unidos. Outro dado mais recente foi revelado em um relatório emitido pela CAST, empresa líder mundial em análise e medição de sistemas, que fez uma análise detalhada para medir a qualidade estrutural de 745 aplicações, avaliando fatores como segurança e desempenho, uma das conclusões foi que custa mais corrigir um problema depois da implantação de um programa do que investir no desenvolvimento do processo. O estudo avaliou as 365 milhões de linhas de código dentro dessas 745 aplicações e estimou que o custo médio para eliminar os problemas encontrados por linha de código é de 3,61

dólares calculado sobre o valor da hora cobrado para correção de erros computacionais nos Estados Unidos [Crash 2011]. Esses custos excedentes poderiam ser amenizados, uma vez que devido à ausência de uma gestão formal da atividade de teste, esses defeitos só são encontrados após a liberação para a produção e o orçamentário dos custos para identificar e corrigir esses problemas, aponta de 100 a 1000 vezes mais se fossem testados após sua introdução [Rios 2013]. Além do prejuízo, também existem impactos decorrentes da incredulidade dos clientes em relação ao produto desenvolvido.

Por isso, as atividades de teste de *software* devem ser aplicadas para a entrega com maior qualidade ao cliente. Elas existem para que se obtenha um controle adequado da qualidade do produto desenvolvido, buscando averiguar se os requisitos propostos pelo cliente serão atendidos e trazendo consigo vantagem competitiva ao produto dentro do mercado.

De acordo com [Rios 2013], o teste é, em essência, um processo bem executado com objetivo de avaliar o comportamento daquilo que foi especificado/executado de forma controlada. Ampliando a definição, o foco do Teste está na execução de um produto, determinando se ele atingiu suas especificações e funcionalidades adequadas no ambiente para o qual foi projetado. Seu objetivo é revelar falhas em um produto para que as causas sejam identificadas e possam ser corrigidas. Por conta de dada característica às atividades de teste, pode se dizer que sua natureza é "destrutiva", e não "construtiva", pois visa ao aumento da confiança em um produto através da exposição de seus problemas, porém antes de sua entrega ao usuário final. Quanto maior a extensão dos testes, maior será a qualidade do sistema e menores serão os riscos de erros.

Atualmente, com o uso da internet, um *software* tem que se integrar com muitos ambientes, o que gerou uma mudança significativa na abrangência e complexidade das aplicações, como explica [Rios 2013], onde inúmeros componentes, comunicando entre si, relevam fatores como segurança e performance e, em consequência disso, a atividade de testar se tornou cada vez mais complexa e específica. [Myers 2004] explica que o processo de teste é complexo e não deve ser definido de forma simplificada ou genérica, o devido teste não é uma atividade invariável, mas uma técnica onde requer planejamento e qualificação de seu testador.

Diante de tais informações, pode-se dizer que "a melhoria do processo de teste deve ser uma atividade contínua e permanente" [Rios 2013]. Esse princípio representa a necessidade de o processo de teste permanecer em evolução constante para manter a competitividade e principalmente a qualidade em relação aos concorrentes. "Como testar um sistema ou *software* é uma atividade complexa, sempre existirão itens ou atividades a serem melhoradas"[Rios 2013].

Autores afirmam "as práticas e processos de teste de *software*, em muitas empresas, estão longe de ser maduras[...]. Tais práticas indevidas levam a diversos resultados negativos, por exemplo, à ineficácia das práticas de teste na detecção de todos os defeitos e ao excesso de custos e cronogramas das atividades de teste" [Garousi 2017]. A imaturidade no processo de desenvolvimento chama atenção para a possibilidade de aplicar um método que diminua as dificuldades causadas por ela. Os métodos já existentes estão consolidados e quando seguidos geram os resultados esperados, entretanto as despesas de implantação são altas e demandam tempo, como

afirma [Ellatif 2015] a indústria de Tecnologia de Informação investiu um esforço substancial para melhorar a qualidade de seus produtos como ISO, CMMI e TMMI, sendo os dois últimos modelos de maturidade utilizados em processos. Embora a aplicação dos critérios de maturidade do TMMI (*Test Maturity Model Integration* - Integração do Modelo de Maturidade de Teste) tenha um impacto positivo na qualidade do produto na produtividade de engenharia de testes e no esforço de tempo de ciclo, ela requer processos pesados de desenvolvimento com grandes investimentos em termos de custo e tempo, gerando dificuldades de aplicabilidade em empresas de médio e pequeno porte. Portanto, existem cenários onde, apesar de eficientes, os métodos mais conhecidos não atendem completamente à necessidade das organizações sendo acessíveis à aplicabilidade e, principalmente, ao cunho financeiro.

Diante de muitas metodologias, autores concluem que inúmeras abordagens de melhoria de processo de testes estão disponíveis, mas nem todas são geralmente aplicáveis à indústria (apesar de serem muito semelhantes) [Afzal 2016]. Atentando para essa lacuna citada anteriormente, onde os processos de maturidade de testes são tidos como pesados, caros e de difícil aplicação (apesar de gerarem resultados positivos), este trabalho dedica-se a satisfazer a necessidade de uma ferramenta aplicável e acessível à indústria, seja ela de pequeno ou de grande porte.

O objetivo deste trabalho foi desenvolver uma ferramenta que oriente e avalie a melhoria contínua no processo de teste de *software* em uma forma simples, de baixo custo e de fácil aplicação. Além disso, que a aplicação desse processo de melhoria envolva toda a equipe de projeto. Ainda assim buscando introduzir na ferramenta elementos de maturidade de processo de teste presentes em modelos referenciais na área.

Esse artigo está organizado da seguinte forma: a seção 1 está composta pela introdução; sendo que a seção 2 apresenta o referencial teórico deste documento abordando os modelos de maturidade de processo e de teste, que serviram como base para o desenvolvimento da ferramenta proposta; A seção 3 descreve a metodologia utilizada para compor os argumentos descrevendo os procedimentos para alcançar os resultados esperados; A seção 4 descreve, de forma detalhada, a ferramenta proposta neste trabalho; Na seção 5 é apresentado o ambiente onde foi aplicada a ferramenta e a avaliação do método da mesma; A seção 6 apresenta discussões sobre a ferramenta e os retornos recebidos das avaliações da mesma; E a seção 7 aponta à conclusão do trabalho, bem como suas considerações finais.

2. Referencial Teórico

2.1 Modelos de Maturidade de Processo

A fim de melhorar a qualidade de seus produtos, a Indústria de *Software* vem concentrando esforços na melhoria de seus processos de desenvolvimento com modelos de maturidade, tais como o CMMI (*Capability Maturity Model Integration* – Modelo Integrado de Maturidade e de Capacidade) e o MPS.BR (Melhoria de Processo do *Software* Brasileiro). O CMMI é um modelo de maturidade para melhoria de processo criado pelo *Software Engineering Institute* - Instituto de Engenharia de *Software* (SEI) "destinado ao desenvolvimento de produtos e serviços compostos pelas melhores

práticas associadas às atividades de desenvolvimento e de manutenção que cobrem o ciclo de vida do produto desde a concepção até a entrega e manutenção"[CMMI 2006]. O MPS.BR foi criado pela Softex em 2003 com o objetivo de avaliar e desenvolver um modelo para melhoria de processos de desenvolvimento de sistemas em empresas brasileiras tendo foco em micros, pequenas e médias empresas, baseando-se em normatizações, como: ISO/IEC 9126 e ISO/IEC 15504.

No CMMI, cada nível de maturidade, com exceção do Inicial, é composto por Áreas Chaves do Processo (*Key Process Areas - KPAs*). Cada KPA identifica as atividades que, quando são executadas corretamente, alcançam objetivos importantes para o crescimento da capacidade do processo, sendo requisitos para obtenção de um novo nível. Quando a organização atinge devido nível de maturidade, é porque todas as KPAs daquele nível foram satisfeitas. Cada KPA é representada em termos de práticas-chave (*Key Practices*). Uma prática-chave retrata as atividades e a infraestrutura necessárias para a efetiva implementação e estabelecimento de uma KPA. Uma prática-chave descreve "aquilo" a ser feito, e não "como" deve ser feito. A visão geral do CMMI é um modelo de estágios com 5 (cinco) níveis de maturidade, e a organização é avaliada estando em apenas um dos níveis citados, ou seja, a empresa pode possuir práticas de níveis mais altos e continuar em patamar menor por não atender todas as "áreas chaves" desse nível.

Os modelos de maturidade surgiram para avaliar e melhorar o nível qualitativo dos processos de testes aplicados em uma organização desenvolvedora de *software*. Um dado importante que o SEI estima é o fato das empresas de nível Inicial dedicarem-se a cerca de 55% dos esforços direcionados a fim de corrigir defeitos do projeto desenvolvido. À medida que a empresa vai adquirindo maturidade, esses índices vão sendo gradativamente reduzidos [BARTIÉ 2002]. Entretanto, nenhum desses modelos e dessas normatizações tratam de forma específica a melhoria do processo de testes.

Em resposta a tal deficiência, novas abordagens para melhoria do processo de teste surgiram no mercado, tais como o TMMI (*Test Maturity Model Integration* - Integração do Modelo de Maturidade de Teste), TPI (*Test Process Improvement* - Melhoria do Processo de Teste), *White Book* (Livro Branco), *Career Space* (Espaço de carreira), TMap (*Test Management Approach* - Abordagem de gerenciamento de testes) e ISTQB (*International Software Testing Qualifications Board* - Conselho de Qualificações Internacionais de Testes de *Software*). Assim, na seção 2.2 serão descritos resumidamente cada uma dessas abordagens.

2.2 Modelos de Maturidade de Processo em Teste de *Software*

2.2.1 White Book

O *White Book* é um modelo de maturidade criado na Espanha com o objetivo de servir como ferramenta para avaliar a maturidade da capacidade de Inteligência Institucional em Instituições de Ensino Superior. O modelo baseia-se na experiência de contribuintes e foi concebido como uma ferramenta específica para as instituições de educação, portanto, não é um modelo de maturidade de inteligência de negócios de propósito geral

[Walz 2013]. Além disso, define as habilidades pessoais mais valorizadas pelos engenheiros de Tecnologia de Informação e Comunicação na Espanha [Saldaña-Ramos 2012].

O modelo identifica três perfis: Desenvolvimento de *software*; Gestão de sistemas e Exploração de tecnologias da informação. O *White book* baseia-se na avaliação de nove dimensões conforme Figura 1. Cada dimensão mede a maturidade de uma área chave específica que foi julgada criticamente para o sucesso e a maturidade total do conjunto. Cada dimensão (e todo o modelo) é expressa em 5 níveis consecutivos de maturidade crescente, cada um rotulado com um nome designado e descrito em termos qualitativos [Walz 2013].

		Níveis				
		Ausente	Inicial	Expandido	Consolidado	Institucionalizado
Dimensões	Equipe	Ausente	Local	Global Virtual	Tempo integral global	Centro de Competências
	Escopo	Nenhum / Desconhecido	Especializado	Múltiplo	Generalizado	Completo
	Papel das Unidades Estratégicas de Negócios	Inconsciente	Consciente	Participante	Apoia	Administração de dados
	Produtos de dados	Nenhum / Desconhecido	Limitado	Expandido	Maioria	Completo
	Cobertura do usuário	Nenhum / Desconhecido	Limitado	Expandido	Maioria	Universal
	Engajamento do Usuário	Inconsciente	Consciente	Clientes	Condutores	Co-proprietários
	Gerenciamento de dados	Inconsciente	Consciente	Gerenciado	Suportado	Executado
	Valor de Negócio	Escasso	Opcional	Interessado	Necessário	Crítico
	Estratégia de Suporte	Variável Livre	Localmente incorporado	Base do projeto	Projeto Sustentável	Serviço Viável

1 2 3 4 5

Figura 1. Estrutura do Modelo *White Book* Fonte: Walz 2013 (adaptado pelo autor)

2.2.2 Career Space

É um consórcio de nove das maiores companhias de TIC (Tecnologia de Informação e Comunicação) europeias que fornece um conjunto de perfis de habilidades genéricas envolvendo as principais áreas de trabalho da indústria de *software* [Saldaña-Ramos 2012]. O *Career Space* possui um repositório na Internet com documentos e materiais técnicos visando divulgar amplamente uma descrição dos papéis, das habilidades e competências requeridas pela indústria desse setor, mostrando as necessidades do mercado de Tecnologia da Informação e Comunicação na Europa [Vieira 2006].

O *Career Space* identifica 18 perfis genéricos de trabalho relacionados a quatro áreas principais: telecomunicações; *software* e serviços; produtos e sistemas; e setor transversal (inclusive) para teste. Para cada perfil de trabalho há uma descrição do tipo de área, tarefas, habilidades, competências requeridas e oportunidades futuras, além de características técnicas e pessoais que são interessantes em um profissional [Vieira 2006].

2.2.3 *Test Management Approach* (TMap)

O modelo TMap "é a abordagem de gerenciamento de teste proeminente da Sogeti, empresa do grupo Capgemini, para os testes estruturados de sistemas de informação, e é usado como um modelo de referência para muitas organizações" [Saldaña-Ramos 2012].

É direcionado a empresas e profissionais com interesse em implantar um processo de testes que funciona de forma integrada ao seu processo de desenvolvimento [Lages 2011].

O TMap compartilha o processo de teste global em cinco passos consecutivos: planejamento e controle; preparação; especificação; execução; e conclusão. Enfatiza as fases de planejamento e preparação porque são cruciais para alcançar um teste de alta qualidade no processo [Saldaña-Ramos 2012]. Essa abordagem define 17 funções de teste junto com as atividades, os conhecimentos e habilidades necessários.

2.2.4 *International Software Testing Qualifications Board* (ISTQB)

O *International Software Testing Qualifications Board* (ISTQB) é uma organização que fornece certificação para testadores de *software*. A estrutura baseia-se em um corpo de conhecimento e regras de exame que são aplicadas de forma consistente em todo o mundo com exames e material de suporte disponível em vários idiomas.

Essa estrutura de certificação considera três níveis: Nível de fundação; Nível avançado; E nível especialista. Esse programa fornece aos testadores conhecimentos exigidos para realizar atividades de teste em diferentes níveis de experiência em conjunto com o tempo necessário para adquiri-los [Saldaña-Ramos 2012].

2.2.5 *Test Maturity Model Integration* (TMMI)

O TMM foi concebido por um grupo de pesquisadores do Instituto de Tecnologia de Illinois [Burnstein *et al*., 1998] para ser usado junto ao CMMI, visando importantes questões de profissionais da área de teste e certificação da qualidade de *software*. Sua utilização ajuda a documentar o nível corrente e fornece uma meta para realizar as melhorias necessárias para o processo de teste. Composto por 5 (cinco) níveis de maturidade, são eles: inicial, definição de fase, integração, gerenciamento e medições, otimização e prevenção de falhas e controle de qualidade. O TMM sofreu evoluções ao longo dos anos e, atualmente, é denominado TMMI (*Test Maturity Model Integration*).

Conforme a Figura 2, para atender cada nível é necessário consolidar os requisitos descritos.

Nível	Descrição
Nível 1 - Inicial	Neste nível, uma organização está usando métodos *ad hoc* para testes, de modo que os resultados não são repetíveis e não há padrão de qualidade.
Nível 2 - Definição	Neste nível, o teste é definido como um processo, portanto, pode haver estratégias de teste, planos de teste, casos de teste, com base em requisitos. O teste não começa até a conclusão dos produtos, de modo que o objetivo do teste é comparar os produtos com os requisitos.
Nível 3 - Integração	Neste nível, o teste é integrado a um ciclo de vida do software, por exemplo, o modelo V. A necessidade de testes é baseada no gerenciamento de riscos e o teste é realizado com alguma independência da área de desenvolvimento.
Nível 4- Gestão e medição	Neste nível, as atividades de teste ocorrem em todas as etapas do ciclo de vida, incluindo revisões de requisitos e projetos. Os critérios de qualidade são acordados para todos os produtos de uma organização (interna e externa).
Nível 5 - Otimização	Neste nível, o próprio processo de teste é testado e melhorado em cada iteração. Isso geralmente é alcançado com o suporte a ferramentas, e também apresenta objetivos como a prevenção de defeitos através do ciclo de vida, em vez de detecção de defeitos (zero defeitos).

Figura 2. TMMI Níveis de Maturidade Fonte: https://en.wikipedia.org/wiki/Testing_Maturity_Model (adaptado pelo autor)

2.2.6 TPI–Test Process Improvement

O modelo de TPI ajuda a definir etapas ascendentes e controláveis na melhoria do processo de teste. [Andersin 2004] descreve que para organizar os testes com eficiência, diferentes níveis de teste devem ser usados e cada um deles aborda um certo grupo de requisitos para especificações técnicas ou de requisitos funcionais. Esses são denominados áreas chaves no modelo TPI, sendo cada área classificada dentro de um nível de maturidade. "A maneira como as áreas-chave são organizadas dentro de um processo de teste determina a maturidade do processo." Para ter uma visão das áreas-chave, o modelo fornece níveis progressivos (de A a D), conforme demonstrado na Figura 3. Cada nível superior é melhor do que seu nível anterior em termos de tempo (mais rápido), dinheiro (mais barato) e/ou de qualidade (melhor) [Andersin 2004].

As classificações dos níveis são realizadas por meio de pontos de checagem. Os pontos de checagem são utilizados para determinar os requisitos dos certos níveis. Baseados nos pontos de checagem, um processo de teste pode ser avaliado e, para cada área-chave, o nível apropriado pode ser estabelecido.

Escala / Area Chave	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Estratégia de Teste		A					B				C		D	
Modelo de Ciclo de vida		A			B									
Momento de envolvimento			A				B				C		D	
Estimativa e Planejamento				A							B			
Técnicas de Especificação de teste		A		B										
Técnicas de Teste estático					A		B							
Métricas						A			B			C		D
Automação de teste				A				B			C			
Ambiente de teste				A				B						C
Ambiente de escritório				A										
Comprometimento e Motivação		A				B						C		
Treinamento e funções do teste				A			B			C				
Escopo da metodologia					A						B			C
Comunicação			A		B							C		
Relatório		A			B		C					D		
Gerenciamento do Defeito		A				B		C						
Gerenciamento do Processo de teste		A		B								C		
Avaliação							A			B				
Teste de baixo nível					A		B		C					

Figura 3. Pontos de checagem da Matriz de maturidade de teste TPI Fonte: Andersin 2004 (adaptado pelo autor)

2.2.7 Comparativo Entre os Modelos de Referência e Padrões para Teste de *Software*

[Saldaña-Ramos 2012] publicou um trabalho de revisão dos modelos de referência e padrões para testes de *software*. O levantamento avaliou os modelos: *White Book, Career Space,* TMap, TMMI e ISTQB. Os critérios de avaliação foram definidos com base na amplitude de cobertura dos modelos sobre um conjunto de adjetivos como: competências gerais dos profissionais, competências específicas relacionadas ao teste, nível de competências e documentos de referência de testes.

A Figura 4 demonstra a comparação de abrangência dentro dos modelos citados. Nesta figura, o símbolo "X" indica não-cobertura ou ausência e o símbolo "-" indica que o modelo possui cobertura sobre esse fator de avaliação. O estudo indica que o modelo TMMI é o mais completo e abrangente.

Comparação entre modelos de referência e padrões					
Variável do Fator	**The White Book**	**Career Space**	**TMAP**	**TMMI**	**ISTQB**
Competências gerais	X	X	-	-	-
Competências especificas relacionadas ao teste	-	-	-	-	-
Níveis de competência	-	-	-	-	-
Documentos de testes	-	-	X	-	X

Figura 4. Exemplos de regras da ferramenta Fonte: Saldaña-Ramos 2012 (adaptado pelo autor)

2.3 Ciclo *PDCA*

O *PDCA* é uma das primeiras ferramentas da qualidade. A sigla tem origem das iniciais das palavras no inglês que significam *Plan*, *Do*, *Check* e *Act*, ou seja, em português teria uma tradução aproximada como: *Planejar, Executar, Verificar e Agir*. Essa ferramenta foi criad na década de 1920 por Walter A. Shewhart, que algum tempo depois ganhou fama através de William Edward Deming que disseminou a ideia [Oribe 2010]. É utilizada principalmente com objetivo de trazer melhoria de processos fabris ou de negócio. A primeira etapa do *PDCA* é Planejar (*Plan*): o foco é a definição dos objetivos, metas, para os métodos e procedimentos a serem empregados para atingir a melhoria de processos.

A segunda, Fazer (*Do*) é destinada ao planejamento estruturado na primeira fase e executado. Na terceira etapa, Checar (*Check*), são verificadas as ações executadas e com os dados levantados pela organização efetuam-se as análises críticas de suas ações, realizando, se necessário, ações de correção ou melhoria na solução adotada ou nos próprios processos. Por fim, na última etapa, Agir (*Act*), corrigem-se falhas, quando houverem, identificadas no planejamento e nas ações executadas e padronizam-se as ações que geraram os resultados estabelecidos.

3. Materiais e Métodos

A metodologia utilizada foi composta por uma revisão bibliográfica e uma pesquisa exploratória, que possibilitou maior aquisição de conhecimento em teste de *software* e seu processo de desenvolvimento, assim como, dos modelos existentes para melhoria de processos de teste.

Na revisão bibliográfica foram identificadas as dificuldades e oportunidades de melhoria para cada modelo e foi descoberta a ausência de uma ferramenta visando melhoria contínua com implementações e usabilidades simplificadas. A partir desses dados, foi realizado o planejamento da ferramenta, que consistiu inicialmente na definição dos principais pilares de maturidade, dos critérios de avaliação de melhoria (comparados aos modelos já existentes) e das regras a fim do preenchimento da própria ferramenta. Uma vez implementada a ferramenta, em planilha eletrônica, foi definido como ela seria implantada, em que ambiente a mesma poderia ser aplicada, assim como o público alvo no qual ela beneficiaria pretendendo conhecer a efetividade da ferramenta como orientadora e avaliadora do processo de melhoria de teste.

O próximo passo foi a aplicação de estudos de caso e obtenção de *feedbacks* em 3 (três) empresas especializadas em desenvolvimento de *software* na região de Campinas, São Paulo. Os *feedbacks* coletados nos estudos de caso foram recebidos através de perguntas abertas realizadas por *e-mail* e foram utilizados para corrigir e aprimorar a versão final da ferramenta.

4. Ferramenta de Auto Avaliação para Melhoria do Processo de Testes

A ferramenta proposta neste trabalho foi elaborada utilizando-se do aplicativo *Planilhas Google* - uma planilha eletrônica disponível em "nuvem" que possibilita a criação e edição de documentos de forma colaborativa entre vários usuários. A estrutura da ferramenta é composta por 8 pilares intitulados como Execução de Testes, Equipe, Especificação de Teste, Preparação de Teste, Planejamento de Teste, Gestão de Controle de Testes, Automatização de Testes e Melhoria Contínua, que definem o nível de maturidade do processo de teste conforme a Figura 5.

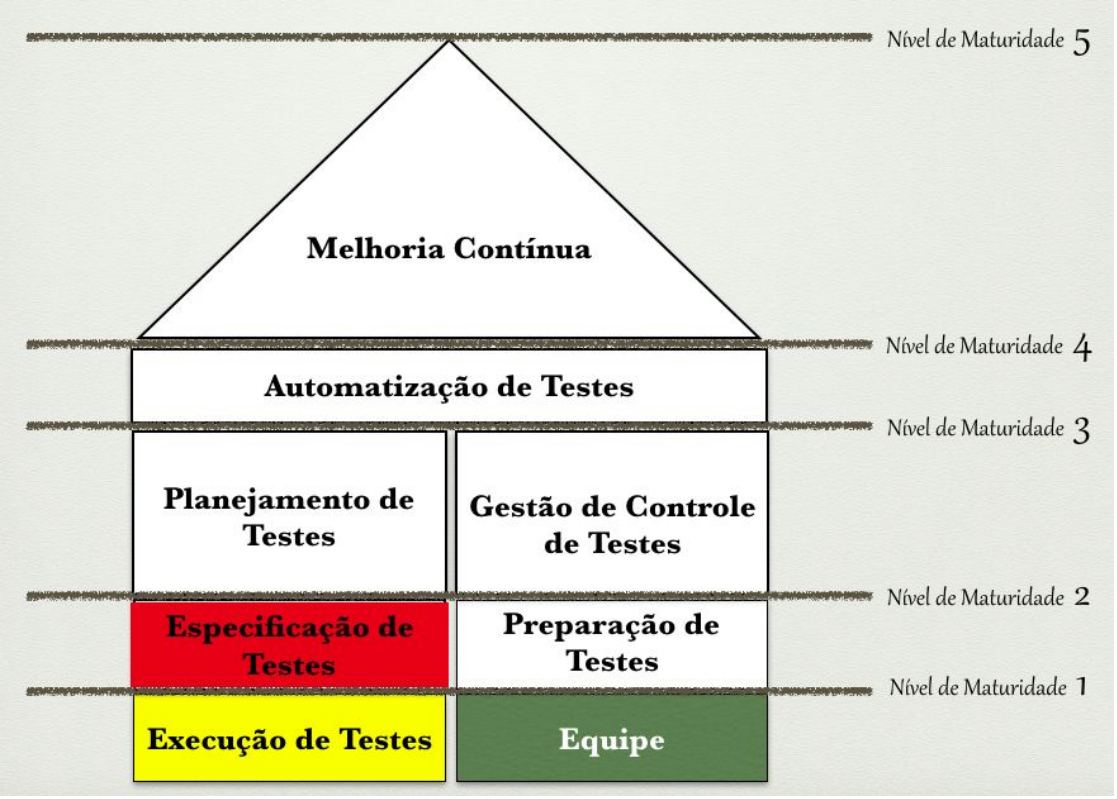
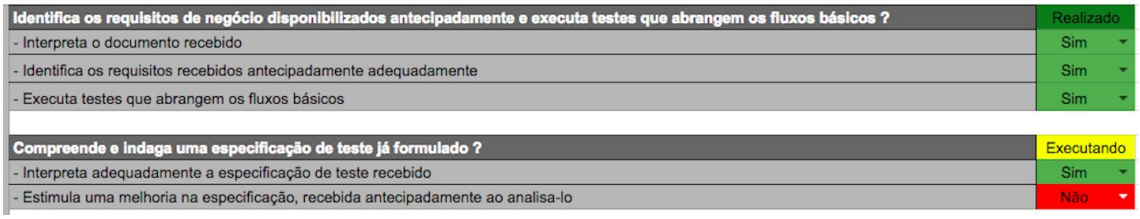


Figura 5. Pilares da Ferramenta e Níveis de Maturidade Fonte: Criado pelo autor

São 5 níveis denominados N1 a N5 compostos por perguntas principais e "subperguntas". Cada nível de maturidade foi definido baseado em pontos chaves do processo de desenvolvimento da Engenharia de *Software*. Os critérios que definem a maturidade são perguntas baseadas em tópicos de validação e verificação do modelo de maturidade CMMI, que sugere metas e práticas genéricas e/ou específicas nesse sentido. As "subperguntas" da ferramenta devem ser respondidas manualmente com "Sim" ou "Não".

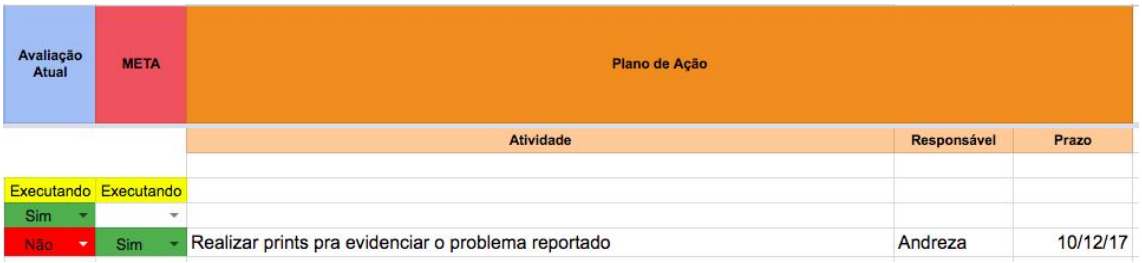
A resposta preenchida com "Sim" indica o cumprimento do processo e "Não" para o caso de ainda ter que atender ao requisito. Quando todas as "subperguntas" são respondidas positivamente, a pergunta principal recebe a definição com o dizer *Realizado* automaticamente e o bloco fica na cor verde. Caso alguma "subpergunta" ainda não esteja sendo cumprida, como previsto pelo o critério adotado, ela é definida com o status de *Executando* e, então, o bloco fica na cor amarela conforme ilustra a Figura 6 ilustra.



Identifica os requisitos de negócio disponibilizados antecipadamente e executa testes que abrangem os fluxos básicos ?	Realizado
- Interpreta o documento recebido	Sim
- Identifica os requisitos recebidos antecipadamente adequadamente	Sim
- Executa testes que abrangem os fluxos básicos	Sim
Compreende e indaga uma especificação de teste já formulado ?	Executando
- Interpreta adequadamente a especificação de teste recebido	Sim
- Estimula uma melhoria na especificação, recebida antecipadamente ao analisa-lo	Não

Figura 6. Preenchimento da pergunta principal e subperguntas Fonte: Criado pelo autor

Dessa forma, é possível atender aos requisitos de diferentes níveis de maturidade, sem que haja limitação para explorar o próximo pilar. Ao verificar a necessidade de melhoria, em dada "subpergunta" é estabelecida uma meta como Plano de Ação, conforme a Figura 7.



Avaliação Atual	META	Plano de Ação		
		Atividade	Responsável	Prazo
Executando	Executando			
Sim				
Não	Sim	Realizar prints pra evidenciar o problema reportado	Andreza	10/12/17

Figura 7. Preenchimento de Metas e Plano de Ação Fonte: Criado pelo autor

A ferramenta também implementa o conceito de Gestão Visual, conforme a Figura 8. A Gestão Visual se resume a uma estratégia de ajudar a gerenciar os problemas envolvendo a todos os interessados onde o problema está localizado. Tal estratégia permite visualização rápida da situação atual de um processo permitindo aos envolvidos priorização e compreensão do que realmente é necessário, tanto em ações imediatas quanto em planejamento de melhoria. "As formas de apresentação visuais são ilimitadas, pois os recursos visuais são guiados apenas pelo objetivo de tornar fáceis e acessíveis às orientações, procedimentos e a comparação do desempenho real versus o esperado" [Teixeira 2012]. A ferramenta gera um relatório assim que o time executa uma avaliação. Nele é gerada uma visão da situação atual e uma visão de meta para melhoria. Entre as duas visões está o plano de ação de melhoria definido pelo time para atingir a esperada meta.

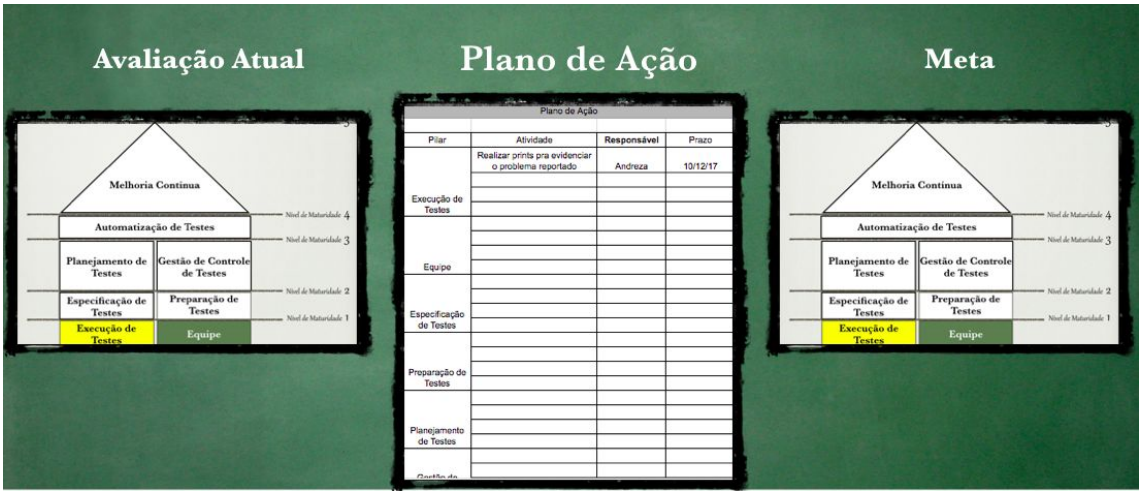


Figura 8. Layout da Gestão Visual Fonte: Criado pelo autor

Quando uma "subpergunta" é questionada e a resposta é Não, é iniciado na ferramenta um processo de Gestão Visual. O pilar, antes em Branco, ganha a cor Amarela e, então, aciona a primeira luz do ciclo *PDCA*, ilustrado na Figura 8, no Planejar (P) onde são definidos uma meta e um plano para que a atividade indicada passe a ser executada tal qual a designação de um responsável por esse plano e um prazo definido. A segunda parte do ciclo é o Fazer (D), que consiste na execução do que foi planejado. A terceira etapa - Checar (C) - consiste na verificação da ação analisando minuciosamente a solução adotada na data estipulada pela etapa anterior. Finalmente, a quarta fase do PDCA - Agir (A) – que é quando se garante a melhoria do processo, corrigindo e alinhando os padrões estabelecidos treinando os envolvidos. Uma vez avaliada, a eficácia do plano realizado gera um ciclo de melhoria com o envolvimento de toda a equipe, garantindo, assim, a qualidade do processo.

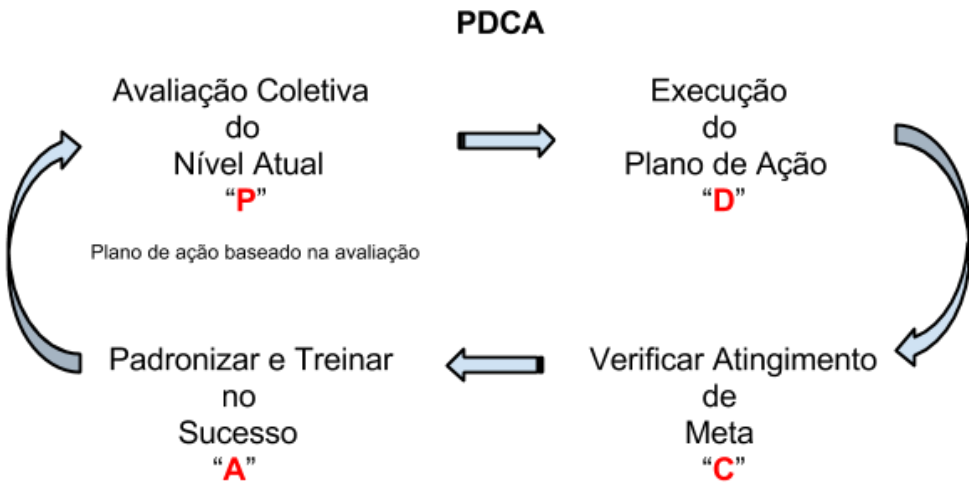


Figura 9. Esquema do PDCA na Ferramenta Fonte: Criado pelo autor

Seguindo esse procedimento, o intuito é de atender positivamente a todas as perguntas do pilar e, ao alcançar esse objetivo, o mesmo é preenchido com a cor Verde no quadro de Gestão Visual, da mesma forma que quando está em processo do ciclo PDCA, ele é preenchido com a cor Amarela. Essa etapa pode ser vista na Figura 8 (página 13).

Na Tabela 1 de Referência ao CMMI, são relacionados alguns critérios da ferramenta desenvolvida com as práticas do CMMI. Ao final deste trabalho, seguirá um anexo com um exemplar da ferramenta que permitirá a visualização da mesma como um todo.

Tabela 1. Tabela de Referência ao CMMI Fonte: Criado pelo autor

Critério de avaliação (pergunta principal)	Referência do CMMI
Pilar: Execução de teste **Nível de maturidade 1**	**SP 2.2 Analisar Resultados de Validação** Analisar os resultados das atividades de validação (Pag 494)
Faz análise crítica sobre as especificações de testes já formuladas ?	Os dados resultantes de testes de validação, inspeções, demonstrações ou avaliações são analisados com relação aos critérios de validação definidos. Os relatórios de análises indicam se as necessidades foram satisfeitas
Pilar: Especificação de testes **Nível de maturidade 2**	**SP 1.3 Estabelecer Procedimentos e Critérios de Verificação** Estabelecer e manter procedimentos e critérios de verificação para os produtos de trabalho selecionados.
São claras e eficientes as especificações de testes de modo que exista uma boa compreensão por parte de qualquer membro da equipe?	Os critérios de verificação são definidos para assegurar que os produtos de trabalho satisfaçam aos seus requisitos
Pilar:Preparação de Teste **Nível de maturidade 2**	**SP 1.2 Estabelecer Ambiente de Verificação** Estabelecer e manter o ambiente necessário para dar suporte à verificação (Pag 505)
O ambiente de teste consegue simular o ambiente de produção de tal forma que os problemas decorrentes na implantação do sistema possam ser minimizados ?	A realização da verificação necessita do estabelecimento de um ambiente apropriado. O ambiente de verificação pode ser adquirido, desenvolvido, reutilizado, modificado ou qualquer combinação dessas ações, dependendo das necessidades do projeto.
Pilar:Gestão de Controle de Testes **Nível de maturidade 3**	**SP 3.2 Analisar Resultados da Verificação** Analisar os resultados de todas as atividades de verificação.
Existem indicadores definidos para medir o desempenho do processo de teste, incluindo desenvolvimento e produção?	Para cada produto de trabalho, todos os resultados disponíveis da verificação são analisados de forma incremental para assegurar que os requisitos tenham sido satisfeitos
Pilar: Automatização de Teste **Nível de maturidade 5**	**SP 3.1 Realizar Verificação** Realizar a verificação nos produtos de trabalho selecionados.
O processo de teste é automatizado com o uso de ferramentas que possibilitam a execução desde o momento de recebimento antecipado dos requerimentos, até a fase de construção e manutenção dos scripts?	A verificação de produtos e produtos de trabalho de forma incremental propicia a detecção precoce de problemas e pode resultar na sua remoção antecipada. Os resultados da verificação economizam gastos consideráveis de isolamento de defeitos e retrabalho associados aos problemas de localização e remoção de defeitos.
Pilar : Melhoria Contínua **Nível de maturidade 5**	**GP 3.2 Coletar Informações para Melhoria** Coletar produtos de trabalho, medidas, resultados de medição e informações para melhoria resultantes do planejamento e da execução do processo de verificação, visando apoiar o uso futuro e a melhoria dos processos e dos ativos de processo da organização.
Existem métodos e gerenciamento para medir melhoria?	Exemplos de produtos de trabalho, medidas,resultados de medição e informações para melhoria: Registros de revisões por pares que incluam tempo de condução e média do tempo de preparação; Número de defeitos de produtos encontrados na verificação por fase de desenvolvimento.

5. Estudo de Caso

5.1. Contexto e Ambiente de Aplicação de Estudo de Caso

Foi aplicado o estudo de caso em três equipes de empresas especializadas em engenharia de *software* da região de Campinas - São Paulo. A Empresa A é uma multinacional especializada em desenvolvimento fundada em 1995 com sede em Campinas e escritórios na América, Europa e Ásia. Possui mais de 2.500 funcionários e atende grandes empresas do mercado com soluções digitais. A Empresa B é de médio porte, com cerca de 500 funcionários divididos em três sedes - sendo uma delas em Campinas - e trabalha com foco em desenvolvimento de sistemas bancários, além de gestão empresarial. A Empresa C é de tecnologia para criação de *websites*, sistemas e aplicações contando com menos de 50 funcionários sendo sediada em Campinas.

5.2. Avaliação do Método e Ferramental Desenvolvido

A ferramenta desenvolvida neste trabalho foi apresentada para as equipes participantes e o objetivo da avaliação foi devidamente explicado aos envolvidos. Por meio de pesquisa, foram avaliados os perfis das empresas participantes e os níveis de maturidade do processo de melhoria contínua de testes das equipes antes da aplicação desse modelo proposto e após o tempo de experimento.

6. Discussões e *Feedbacks*

Na empresa C, a ferramenta foi utilizada por uma equipe composta de 5 desenvolvedores e 1 analista de teste. A empresa, atualmente, encontra-se em processo de certificação ISO 9001. Como pontos positivos foram citados: *"as questões (perguntas principais) são práticas; através da ferramenta foi possível identificar facilmente os processos que precisam ser melhorados; diminuiu a variação dos processos; a tabela sendo dinâmica trouxe maior interação com o usuário"*. Como pontos a serem melhorados, o avaliador frisou que na ferramenta poderia estar presente um histórico de atividades para medir a eficiência e a eficácia do processo como auxílio. Sugeriu também um guia autoexplicativo de como usar a ferramenta fundamentando a importância de cada nível de maturidade. Além dessa sugestão, o avaliador propôs uma melhoria no *layout*: distribuir de forma mais simples na apresentação dos níveis e plano de ação.

Na empresa B, a ferramenta foi utilizada por uma equipe composta por 21 pessoas, sendo 4 analistas de testes, 8 desenvolvedores, 1 arquiteto, 2 analistas de requisitos, 1 gestor de produtos, 4 analistas de suporte e 1 coordenador. Desse usuário, recebeu-se o seguinte *feedback: "Gostei bastante do tema abordado. A planilha aborda diversos pontos de atenção sobre o processo de testes que uma empresa deve monitorar e aprimorar. Aqui na empresa o processo já está bem evoluído, mas ainda existem muitos pontos a melhorar. Pensando em uma empresa que não possui um processo ou planejamento de testes, é bem custoso começar do zero a estruturar uma área e pessoas específicas para isso. A planilha mostra a estaca zero por onde começar e evoluir o*

planejamento de testes, e monitorar a evolução da equipe em relação a qualidade do produto."

Na empresa A, a ferramenta foi avaliada em duas equipes distintas, sendo uma em testes de *API do backend* de 3 aplicativos *mobile* para uma empresa do ramo bancário. O time de API (Equipe 1) é composto por 5 desenvolvedores, 1 arquiteto, 1 *Scrum Master* (líder) e 1 analista de testes, que atende 10 times de *frontend*, cada qual com demandas específicas. Para iniciar o desenvolvimento, os testes precisam gerar valor e tendem a ser objetivos; validando principalmente o funcionamento básico da API.

Na equipe 1, a cada final de *Sprint* - janela de tempo com duração aproximada de 2 a 4 semanas com o propósito de desenvolvimento e entrega de funcionalidades em um projeto - é realizada uma reunião onde o Analista de Teste, junto ao time, repassa os *bugs* abertos e as métricas coletadas ao longo do *Sprint*. Dessa forma, abre-se espaço para a discussão sobre como está a qualidade da entrega e o andamento do processo de desenvolvimento. O usuário revelou: *"A ferramenta de maturidade de testes me ajudou muito a enxergar alguns pontos que realmente faltam no nosso projeto, como a sugestão de ações para defeitos recorrentes e entrega de relatório de entrega de testes. Antes achava que o meu projeto tinha uma maturidade alta; hoje vejo que muito ainda precisa ser feito. Pretendo colocar em prática o PDCA e deixar o processo de testes e qualidade ainda mais completo e organizado, focando na melhoria contínua."* Neste caso, surgiram dúvidas, sendo uma delas, sobre o preenchimento da meta (se a meta já foi atingida ou se aquele objetivo seria a meta propriamente). Isso pode ser visto como um ponto a ser melhorado.

A Equipe 2 questionou vários pontos na interpretação das perguntas, entretanto fez uma sugestão muito plausível, sobre a opção para resposta de critério não ser aplicável naquele momento, já que só existe a opção de resposta como "Sim" ou "Não". Em suma, o avaliador concluiu: *"Foi um belo tópico e foi bem desenvolvido seu trabalho. Com o PDF ficou bem mais clara sua intenção do que quis abordar. Achei de muita valia pois muitas coisas não são lembradas na hora de fazermos um planejamento e são coisas essenciais. Creio que cada um dos níveis também dependa do projeto que você atua. Mas a base dele é bem sólida."*

Levando em consideração as sugestões e críticas apresentadas nos *feedbacks*, pode-se observar que há detalhes que podem ser facilmente adotados para melhor compreensão da ferramenta, bem como sua funcionalidade. Viu-se que uma das melhorias passíveis de serem realizadas é a respeito da apresentação da ferramenta, já que para a realização deste trabalho foram feitas reuniões rápidas via vídeo, um exemplo em PDF, um guia anexo à ferramenta ou um vídeo demonstrativo que seriam eficientes para sanar as dúvidas de usabilidade.

7. Conclusão

Este trabalho teve por objetivo desenvolver uma ferramenta que orientasse e avaliasse a melhoria contínua no processo de teste de *software* em uma forma simples, de baixo

custo e de fácil aplicação, bem como permitisse que o processo de melhoria envolvesse toda a equipe de projeto. Desta forma, a ferramenta foi avaliada por quatro equipes de desenvolvimento distintas em três diferentes empresas.

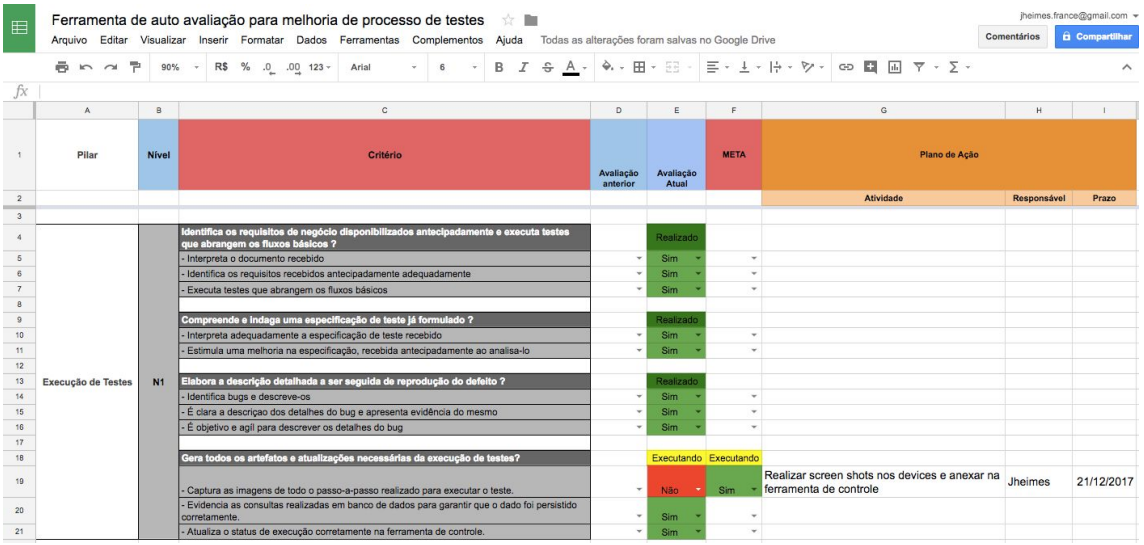
Os resultados dessas avaliações – *feedbacks* - apontaram que a ferramenta é receptiva. Levando em conta o perfil das equipes que a avaliaram, sendo quantidade de membros, tipo de projetos ou cultura da empresa irrelevantes, não houveram dificuldades, tampouco impedimento na sua aplicação. As equipes puderam se auto avaliar e visualizar pontos de melhoria (mesmo quando se acreditava que o processo de teste já estava consolidado), visto que a empresa que estava em processo de certificação viu na ferramenta uma oportunidade de auxílio e clareza para seu processo em implantação.

Uma das dificuldades apresentadas por modelos de maturidades já existentes é o custo da implantação, por isso um dos objetivos era justamente democratizar o acesso a um modelo eficiente; outro ponto que também foi satisfeito, já que em todos os projetos onde a ferramenta foi avaliada não houve nenhum investimento financeiro.

Todavia, alguns pontos de melhoria foram levantados para trabalhos futuros: Primeiramente, acrescentar mais uma opção de resposta aos critérios. Além de "Sim" e "Não", a opção "Não Aplicável"; Desenvolver uma apresentação da ferramenta abrangendo todos os pontos de preenchimento, deixando-os mais explícitos. Outro ponto é o esclarecimento da meta, uma vez que ela surge quando há um alvo a ser atingido, caso contrário, a mesma não precisa ser preenchida. Isso pode ser explorado de forma a ser mais autoexplicativo.

Entretanto, apesar do êxito em partes dos objetivos deste trabalho, foi possível discernir também, que com toda a simplicidade que a ideia da ferramenta possa ter, viu-se que nenhuma equipe ou projeto alcançou os cinco pilares que a ferramenta propõe, demonstrando que simples pontos de um processo podem demandar tempo e complexidade até serem consolidados.

Anexo A - Detalhes da Ferramenta de Autoavaliação



Pilar	Nível	Critério	Avaliação anterior	Avaliação Atual	META	Plano de Ação		
						Atividade	Responsável	Prazo
Execução de Testes	N1	**Identifica os requisitos de negócio disponibilizados antecipadamente e executa testes que abrangem os fluxos básicos ?**		Realizado				
		- Interpreta o documento recebido		Sim				
		- Identifica os requisitos recebidos antecipadamente adequadamente		Sim				
		- Executa testes que abrangem os fluxos básicos		Sim				
		Compreende e indaga uma especificação de teste já formulado ?		Realizado				
		- Interpreta adequadamente a especificação de teste recebido		Sim				
		- Estimula uma melhoria na especificação, recebida antecipadamente ao analisa-lo		Sim				
		Elabora a descrição detalhada a ser seguida de reprodução do defeito ?		Realizado				
		- Identifica bugs e descreve-os		Sim				
		- É clara a descriçao dos detalhes do bug e apresenta evidência do mesmo		Sim				
		- É objetivo e agil para descrever os detalhes do bug		Sim				
		Gera todos os artefatos e atualizações necessárias da execução de testes?		Executando	Executando			
		- Captura as imagens de todo o passo-a-passo realizado para executar o teste.		Não	Sim	Realizar screen shots nos devices e anexar na ferramenta de controle	Jheimes	21/12/2017
		- Evidencia as consultas realizadas em banco de dados para garantir que o dado foi persistido corretamente.		Sim				
		- Atualiza o status de execução corretamente na ferramenta de controle.		Sim				

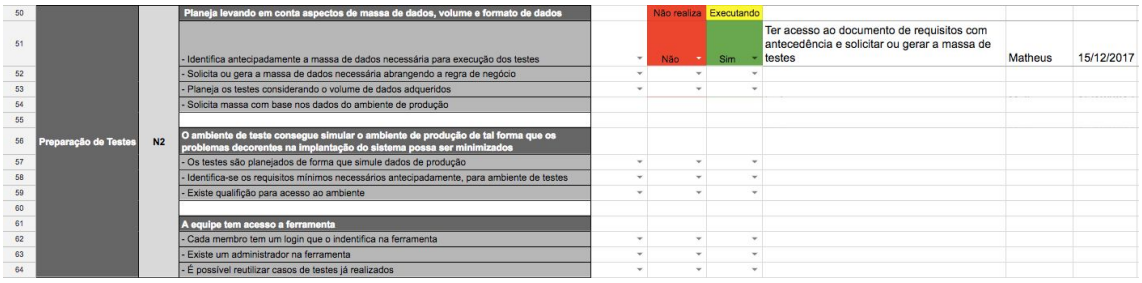
Figura 10 - Detalhes dos Critérios de Avaliação do pilar Execução de Testes Nível 1 de Maturidade.

Equipe	N1	**O teste é considerado como essencial, de forma que haja tempo suficiente para elaboração e execução do mesmo ?**		Realizado	
		- Dedica-se o tempo necessário para o teste		Sim	
		- Dedica-se com comprometimento e foco para entrega		Sim	
		Bom relacionamento interpessoal do tester (ou equipe de teste), desenvolvedores, líderes e cliente ?		Realizado	
		- Demonstra interesse para transmitir as informaçoes à equipe, testers e clientes		Sim	
		- É aberto à orientações e feedback.		Sim	

Figura 11 - Detalhes dos Critérios de Avaliação do pilar Equipe Nível 1 de Maturidade.

Especificação de Testes	N2	**Possui compreensão agil do negócio do cliente, assim como uma visão panorâmica do projeto, considerando a qualidade ?**		Realizado	
		- Dentro de um tempo hábil entende as regras de negócio do projeto		Sim	
		- Realiza especificação de testes cobrindo fluxos básicos e alternativos		Sim	
		- Identifica e descreve fluxos alternativos de testes além do que está representado nos requisitos entregues antecipadamente		Sim	
		- Realiza análise de impacto de negócio		Sim	
		É clara e eficiente a descrição da especificação de testes de modo que exista a compreensão por parte de qualquer membro da equipe?		Realizado	
		- É objetivo ao descrever detalhadamente a especificação de testes		Sim	
		- É simples, claro e não ambígua a especificação de testes, porém capaz de abranger os requisitos necessários da regra de negócios		Sim	
		- O plano de teste é efetivamente executado pelas equipes de testes e/ou desenvolvimento		Sim	
		Consegue priorizar os itens do roteiro de teste especificado para regressão?		Executando	
		- Identifica os pontos críticos do projeto		Sim	
		- Prioriza os pontos críticos do pojeto		Sim	
		- Realiza especificação dos teste de regressão		Não	

Figura 12 - Detalhes dos Critérios de Avaliação do pilar Especificação de Testes Nível 2 de Maturidade.

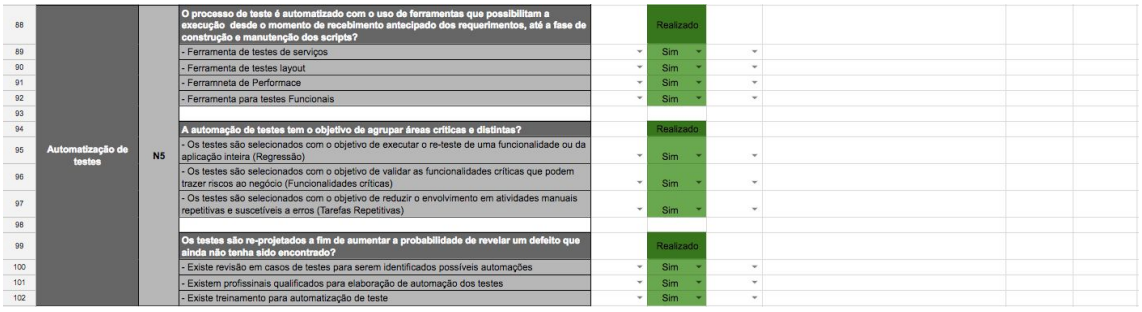


50	Preparação de Testes	N2	**Planeja levando em conta aspectos de massa de dados, volume e formato de dados**		Não realiza	Executando			
51			- Identifica antecipadamente a massa de dados necessária para execução dos testes		Não	Sim	Ter acesso ao documento de requisitos com antecedência e solicitar ou gerar a massa de testes	Matheus	15/12/2017
52			- Solicita ou gera a massa de dados necessária abrangendo a regra de negócio						
53			- Planeja os testes considerando o volume de dados adqueridos						
54			- Solicita massa com base nos dados do ambiente de produção						
55									
56			**O ambiente de teste consegue simular o ambiente de produção de tal forma que os problemas decorentes na implantação do sistema possa ser minimizados**						
57			- Os testes são planejados de forma que simule dados de produção						
58			- Identifica-se os requisitos mínimos necessários antecipadamente, para ambiente de testes						
59			- Existe qualifição para acesso ao ambiente						
60									
61			**A equipe tem acesso a ferramenta**						
62			- Cada membro tem um login que o indentifica na ferramenta						
63			- Existe um administrador na ferramenta						
64			- É possível reutilizar casos de testes já realizados						

Figura 13 - Detalhes dos Critérios de Avaliação do pilar Preparação de Testes Nível 2 de Maturidade.

67	Planejamento de Testes	N3	**Descrever a integração com os projetos de desenvolvimento e até mesmo com outros projetos de teste**		Realizado
68			- É medido o tamanho do projeto de teste para saber quanto ele vai custar. Como métricas de teste temos: Análise de Pontos de Teste ou Pontos de Caso de Teste.		Sim
69			- Se tem conhececimento da tecnologia a ser usada no desenvolvimento		Sim
70					
71	Gestão de Controle de Testes	N3	**Monitora a quantidade de defeitos reportados e fechados durante os testes através da ferramenta de controle**		Realizado
72			- Conhece a curvas de abertura/fechamento de defeitos		Sim
73			- Conhece a quantidade de defeitos abertos e fechados durante o Sprint através da ferramenta de controle		Sim
74			- Realiza análise de causa dos defeitos internos, sugerindo ações		Sim
75					
76			**Monitora os indicadores de qualidade, planejando contra-medidas conseguindo prever indicadores de sprints futuros**		Realizado
77			- Monitora e mobiliza-se para resolver os blocks e os riscos para a execução dos testes		Sim
78			- Conhece os indicadores de qualidade do projeto e suas metas		Sim
79			- Garante que todos os defeitos possuem a causa raiz identificada na ferramenta de controle		Sim
80			- O time é comunicado frequentemente sobre os defeitos recorrentes (tipo, causa raiz, etc)		Sim
81					
82			**Existem indicadores definidos para medir o desempenho do processo de teste, incluindo desenvolvimento e produção?**		Realizado
83			- É realizado um estudo de engenharia de valor com o cliente antecipadamente		Sim
84			- É estabelecido uma meta de bugs a serem reportados antecipadamente		Sim
85			- É realizado um relatório de entrega de testes		Sim
86			- São utilizadas métricas de medição, do tamanho dos projetos de teste?		Sim

Figura 14 - Detalhes dos Critérios de Avaliação dos pilares Planejamento e Gestão de Controle de teste Nível 3 de Maturidade.



88	Automatização de testes	N5	**O processo de teste é automatizado com o uso de ferramentas que possibilitam a execução desde o momento de recebimento antecipado dos requerimentos, até a fase de construção e manutenção dos scripts?**		Realizado
89			- Ferramenta de testes de serviços		Sim
90			- Ferramenta de testes layout		Sim
91			- Ferramneta de Performace		Sim
92			- Ferramenta para testes Funcionais		Sim
93					
94			**A automação de testes tem o objetivo de agrupar áreas críticas e distintas?**		Realizado
95			- Os testes são selecionados com o objetivo de executar o re-teste de uma funcionalidade ou da aplicação inteira (Regressão)		Sim
96			- Os testes são selecionados com o objetivo de validar as funcionalidades críticas que podem trazer riscos ao negócio (Funcionalidades críticas)		Sim
97			- Os testes são selecionados com o objetivo de reduzir o envolvimento em atividades manuais repetitivas e suscetíveis a erros (Tarefas Repetitivas)		Sim
98					
99			**Os testes são re-projetados a fim de aumentar a probabilidade de revelar um defeito que ainda não tenha sido encontrado?**		Realizado
100			- Existe revisão em casos de testes para serem identificados possíveis automações		Sim
101			- Existem profissinais qualificados para elaboração de automação dos testes		Sim
102			- Existe treinamento para automatização de teste		Sim

Figura 15 - Detalhes dos Critérios de Avaliação do pilar Automatização de testes Nível 4 de Maturidade.

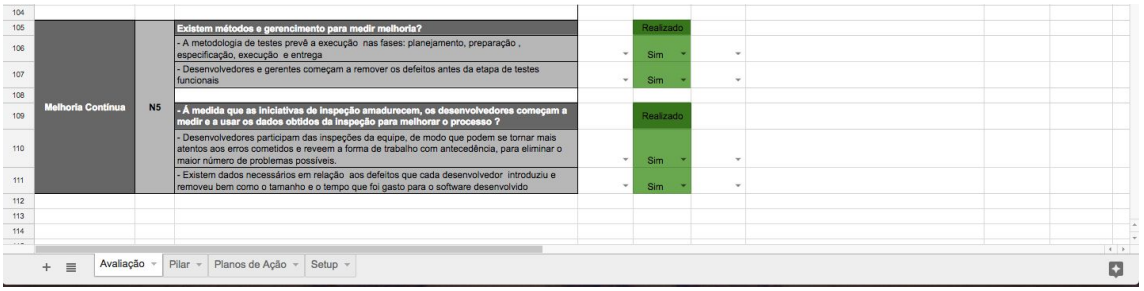


Figura 16 - Detalhes dos Critérios de Avaliação do Pilar Melhoria Contínua Nível 5 de Maturidade.

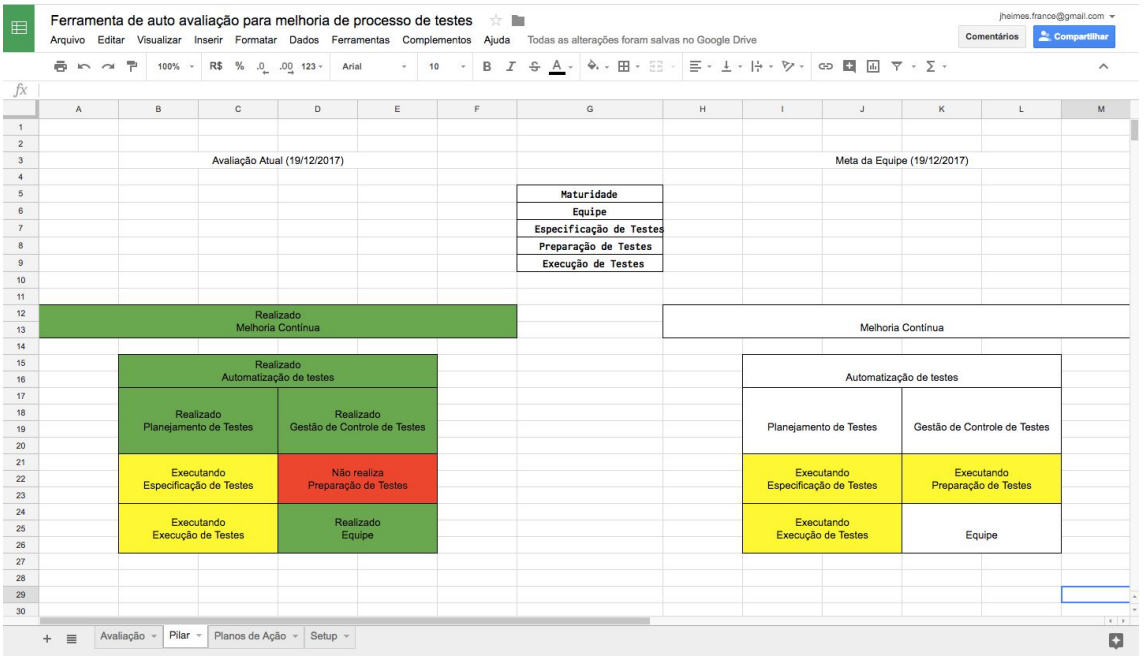


Figura 17 - Gestão Visual dos Pilares Cores Geradas Automaticamente ao Preencher os Critérios de Avaliação.

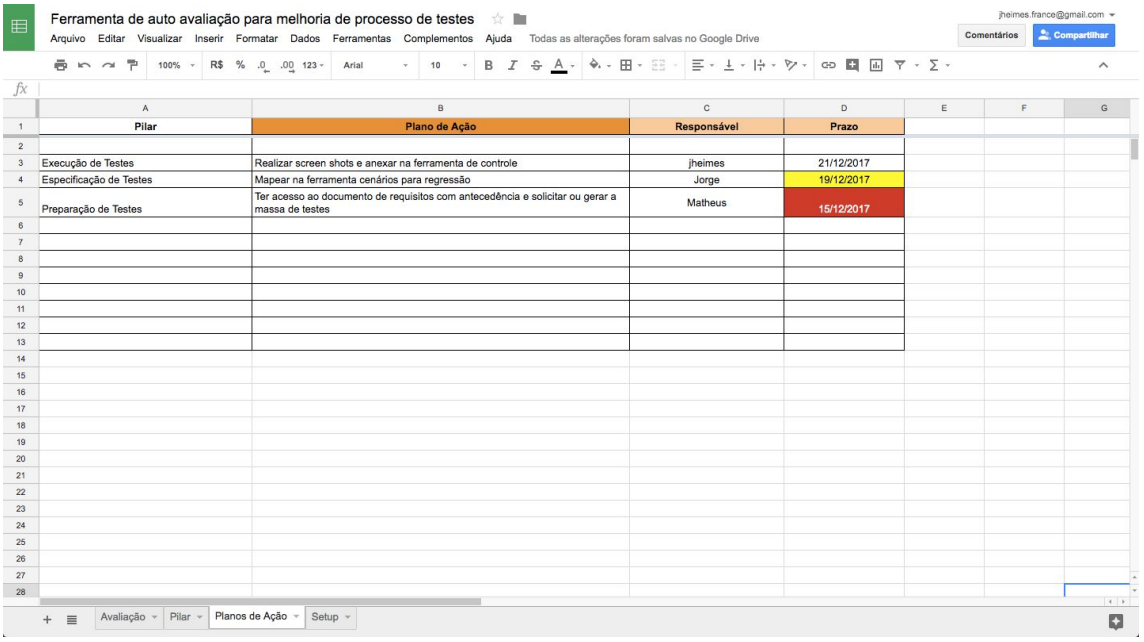


Figura 17 - Plano de Ação Gerado Automaticamente ao Preencher a aba Avaliação.

9. Referências

Afzal, W.; Alone, S.; Glocksien, K.; Torkar, R. (2016). Software test process improvement approaches: A systematic literature review and an industrial case study.

Andersin, J. (2004). A model for test process improvement.

Bartié, A. (2002). Garantia da Qualidade de Software: adquirindo maturidade organizacional. Elsevier, 1o edition.

Carnegie Mellon University Software Engineering Institute. (2006). CCMI para desenvolvimento versão 1.2. Pitsburg U.S.

Crash, CAST. (2011). Report on Application Software Health NEW YORK, DECEMBER 8, http://research.castsoftware.com.

Ellatif, M. A.; Fathy, E. M.; Farid, A. B. (2015). Towards agile implementation of test maturity model integration (tmmi) level 2 using scrum practices.

Garousi, V.; Felderer, M.; Hacaloğlu, T. (2017). Software test maturity assessment and test process improvement: A multivocal literature review.

KosciankI, A.; Santos, M. (2007). Qualidade de Software: Aprenda as Metodologias e Técnicas mais Modernas para o Desenvolvimento de Software. Novatec, 2a edition.

Lages, D. S. (2011). TMAP NEXT - Metodologia de Testes que se tornou um padrão de facto Revista Engenharia de *Software* Magazine - Ano 4, 39ª edição.

Manzano, A. (2016). A Engenharia de *Software*, A qualidade final do *software* e o papel do profissional de desenvolvimento. 53th edition.

Myers, G. (2004). The Art of *Software* Testing. Hoboken, 2nd edition.

Oribe, C.Y. (2010). A História do PDCA Palestra proferida no XIV Congresso Brasileiro de Qualidade em Serviços de Saúde - São Paulo

Pressman, R. (2011). Engenharia de *Software*. Bookman, 7th edition.

Rios, E.; Moreira Filho, T. (2013). Teste de *Software*. Alta Books, 3rd edition.

Saldaña-Ramos, J. ; Sanz-Esteban, A. . García-Guzmán, J. ; Amescua,. A. (2012). Design of a competence model for testing teams.

Sommerville, I. (2011). Engenharia de *Software*. Pearson, 9th edition.

Teixeira, J. M.; Schoenardie, R. P.; Garcia, L. J. ; Merino, E. A. D. ; Paladini, E. P. (2012). Gestão Visual: Uma proposta de modelo para facilitar o processo de desenvolvimento de produtos , II Conferência Internacional de Design, Engenharia e Gestão para a inovação, Florianópolis.

Vieira, A. D. ; Moraes, A. J. C. ; Santos, B. S. ; Pereira, D. T. V. ; Santos Neto, J. J. dos ; Silva, F. Q. B. da. (2006). Formação em Informática na Era da Convergência Digital Recife.

Walz, A. ; Rieger, B. ; Childers, H.; Picazo, J. J. A., Seguí, M. R.; Bailey, S. Maturity (2013). Model for Institutional Intelligence v1.0 White Book of Institutional Intelligence in Universities.