MARIA CRISTINA DA SILVA AKAGI

Demonstrar os benefícios do sistema de gestão da qualidade em uma empresa de classificados de imóveis online

São Paulo
2016

MARIA CRISTINA DA SILVA AKAGI

Demonstrar os benefícios do sistema de gestão da qualidade em uma empresa de classificados de imóveis online

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do certificado de Especialista em Gestão e Engenharia da Qualidade – MBA / USP

Orientador:
Prof. Dr. Adherbal Caminada Netto

São Paulo

2016

MARIA CRISTINA DA SILVA AKAGI

Demonstrar os benefícios do sistema de gestão da qualidade em uma empresa de classificados de imóveis online

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do certificado de Especialista em Gestão e Engenharia da Qualidade – MBA / USP

Orientador:
Prof. Dr. Adherbal Caminada Netto

São Paulo

2016

DEDICATÓRIA

Dedico esta monografia a Deus que me dá forças, a Jesus Cristos que me dá coragem para vencer as batalhas da vida.

AGRADECIMENTOS

Agradeço aos professores sempre dispostos a ensinar e compartilhar os seus conhecimentos.

Ao orientador pela paciência.

A minha família pela fé e confiança demostrada.

Aos meus amigos pelo incentivo e apoio incondicional.

A todos que permaneceram ao meu lado, e tornaram este caminhar mais fácil.

RESUMO

Este trabalho trata do aprimoramento da qualidade de sistemas a partir de um gerenciamento eficiente para gestores e líderes de empresas voltadas para o ramo tecnológico. As métricas e indicadores serão demonstrados com o propósito de auxiliar na tomada de decisão de risco e na mitigação de defeitos encontrados pelo usuário final no desenvolvimento de novos projetos. A empresa analisada atua no ramo de classificados de imóveis *online* e, visando manter-se líder de mercado, percebeu que necessitava contratar um colaborador especializado na área de qualidade. Com a preocupação de obter mais qualidade no *site*, observaram-se outras necessidades, como melhorar o tempo de entrega de informações na tela do usuário final, uma vez que o concorrente tinha melhor desempenho. Para que houvesse uma análise assertiva do que a empresa deveria ajustar, contratou-se primeiramente um analista de teste técnico e depois mais três analistas para integrarem a equipe, com o objetivo de assegurar a qualidade, e dois desenvolvedores *frontend*, para trabalharem com uma equipe de desenvolvedores *backend.* Com o resultado, houve uma reestruturação da gestão e os resultados iniciais indicam que este é o caminho correto a ser seguido.

Palavras-Chave: Qualidade de sistema. Engenharia da qualidade. Informática. Processo. Gerenciamento. Engenharia.

ABSTRACT

This present study report deals with the improvement of quality systems from an efficient management for managers and leaders aimed at the technology sector. The metrics and indicators will be stated in order to provide assistance in the risk decision making and defects reduction found by customers in the development of new projects. The company analyzed operates in online real estate classifieds and foreseeing to remain in the position of the market leader, it required a specialized collaborator in quality assurance. Due to the company's concern to obtain high quality on the website, other needs have been brought to light, such as, improving delivery times of information to the customer's screen, given that its main competitor had a better performance in that area. In order to have an assertive analysis of what the company should adjust, a technical test analyst was hired at first, followed by three other analysts, and also two frontend developers to join the team of backend developers. As a result, there has been a management reestructuring and the first results indicate that they are on the right track.

Keywords: Quality Assurance. Quality Engineering. Information Technology. Process Management. Engineering.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIAÇÕES E SIGLAS

PDCA – Plan, Do, Check, Action

SUMÁRIO

Introdução

Com o avanço tecnológico, os sistemas computacionais estão mais presentes no dia a dia das pessoas. O aumento da concorrência impulsiona as pessoas a buscar produtos de melhor qualidade, com um custo menor. Com isso, as fábricas de *software* de sistemas e empresas que prestam serviços tecnológicos entram na corrida para atender as necessidades dos clientes de forma rápida e eficaz.

Durante a análise para o aprimoramento do processo de desenvolvimento de sistemas, verificou-se que havia uma grande deficiência. Os desenvolvedores continuavam a utilizar a metodologia tradicional embora a área houvesse sofrido grandes mudanças de exigências, tais como desenvolver mais rápido e com qualidade, e que utilizassem mais da criatividade para solucionar problemas que poderiam impactar o cliente.

Com o desenvolvimento tornando-se mais complexo, os desenvolvedores de sistema tendiam a utilizar a experiência acumulada durante as décadas passadas para modificar a forma como um sistema é desenvolvido. Em 2011 inicia-se o desenvolvimento de sistemas utilizando uma metodologia ágil, como uma alternativa ao formato de gerenciamento do modelo tradicional.

O gerenciamento de um projeto pode ser uma tarefa árdua e a qualidade de sistema depende fortemente do emprego correto de boas metodologias pelos desenvolvedores. Os desenvolvedores passam a entregar mais valor em menor tempo para o cliente. Entretanto, identificou-se um grande retrabalho que causa dificuldade de manutenção no desenvolvimento, devido a quantidade de erros encontrados na aplicação, divergências entre a expectativa e a realidade do produto entregue, prazo de entrega do produto, orçamento e qualidade, entre os participantes do processo: desenvolvedor, analista de teste, gerente de projeto, usuários e clientes.

Embora a equipe de desenvolvimento tenha o suporte de muitas ferramentas, ainda há pontos de deficiência sem o suporte adequado do analista de teste. A equipe de qualidade deveria atuar juntamente com a equipe de desenvolvimento, participando e compreendendo as premissas do Manifesto ágil.

1.1. Objetivo

Demonstrar que um sistema de gerenciamento de qualidade pode ser adotado em empresas que fazem *softwares* de sistemas que, por meio de métodos eficientes, podem melhorar a qualidade do sistema e trazer benefícios para a empresa, a partir de casos de testes projetados e realizados sistematicamente. Alcançar alta cobertura de verificação, provendo tanto a qualidade técnica do sistema quanto a satisfação do usuário.

1.2. Escopo

O sistema de gerenciamento de qualidade será aplicado dentro de um modelo de gerenciamento de desenvolvimento de sistema que utiliza a metodologia ágil. A avaliação das medidas adotadas e as melhorias implantadas serão acompanhadas pela metodologia Seis Sigmas, baseada no ciclo *Plan*, *Do, Check*, *Action* (PDCA).

O ciclo PDCA permite controlar de forma eficiente processos e atividades, de forma padronizada, diminuindo as chances de erros na tomada de decisões.

2. Fundamentação

Na década de 70 ocorreu uma crise no mercado de desenvolvimento de *software* de sistemas, causando uma mudança na forma como eram projetados os novos sistemas. Desde então as empresas começaram a utilizar novos métodos para a criação de sistemas gerando estruturas documentadas e artefatos organizados para o acompanhamento do desenvolvimento do sistema (PRESSMAN, 2006, p. 86).

Os métodos tradicionais começavam sempre com a análise de requisitos, seguidos da modelagem que contemplam as estruturas necessárias para desenvolver o projeto e finalizam na fase de testes e manutenção (PRESSMAN, 2006, p. 53).

As metodologias ágeis surgiram após a crise, em resposta aos modelos tradicionais de desenvolvimento e do reconhecimento de criar novas soluções de forma alternativa, sem a necessidade da criação de estruturas documentadas complexas (BECK *et al.*, 2001). Buscando uma maior qualidade do produto de *software*, prazo, custo, etc. (ZANATTA, 2004).

> [...] metodologias ágeis são mais adaptáveis e menos previsíveis do que metodologias tradicionais. Há menos ênfase em documentação [...] melhora a capacidade da equipe em lidar com as mudanças de requisitos. Assim, para um projeto, seguindo as práticas ágeis, a inclinação da curva do custo da mudança não é mais exponencial, mas sim linear ou mesmo logarítmica. (KAMEL, BEDIWI, ALRASHOUD, p. 33).

2.1. Metodologia tradicional

As metodologias tradicionais surgiram no contexto de desenvolvimento baseado em *mainframes* e terminais "burros". Durante esse período, todo o planejamento e documentação era efetuada antes do sistema ser implementado. O modelo clássico, segundo Pressman (2002), estabelece uma sequência de etapas. Cada etapa tem associada ao seu término uma documentação que deve ser aprovada para que a etapa posterior posa ser iniciada, conforme Figura 1 a seguir.

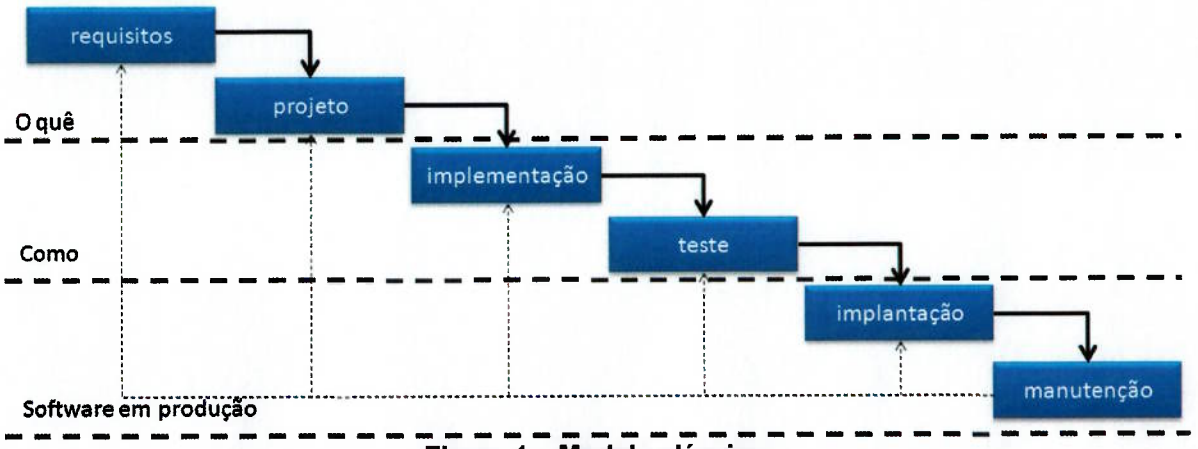


Figura 1 – Modelo clássico

Fonte: DS Consultoria e Sistemas (2014)

2.2. Metodologias ágeis

Apesar de existir mais de uma metodologia ágil, com processos diferentes, todos os métodos são baseados nos princípios do Manifesto para o desenvolvimento ágil de *software*.

As metodologias ágeis são mais suscetíveis a mudanças de requisitos. Variam em termos, práticas e ênfases, compartilham características de desenvolvimento iterativo e incremental, além de reduzirem documentação. Aplicam algumas ideias da teoria de controle de processos industriais para o desenvolvimento de sistemas, reintroduzindo os itens flexibilidade, adaptabilidade e produtividade. Visam tratar mudanças frequentes de requisitos do sistema e outras situações como: trocas de equipes, adaptações durante as interações e orçamento, trocas de ferramentas de desenvolvimento ou mesmo linguagem de programação.

Na Figura 2 a seguir, é possível visualizar como funciona uma metodologia ágil, denominada ferramenta SCRUM.

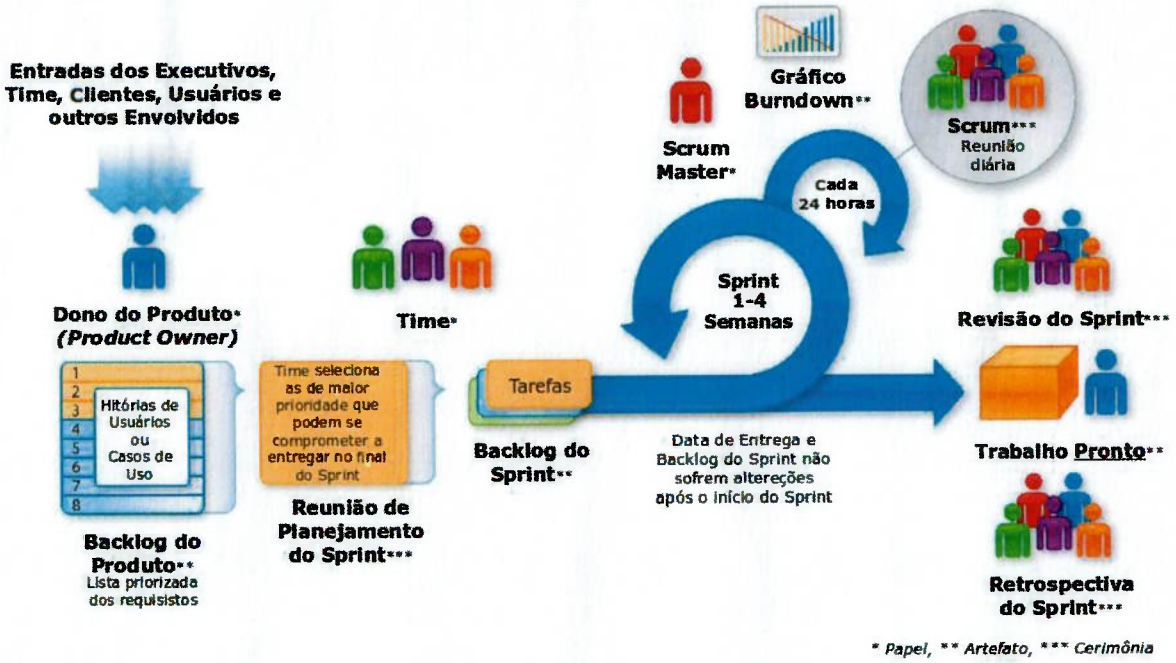


Figura 2 – Ferramenta SCRUM

Fonte: BRAZ (2011)

Definido por 17 (dezessete) líderes, o Manifesto ágil, possui 4 (quatro) valores, onde os itens à esquerda são mais valorizados que os itens à direita (BECK *et al.*, 2001):

1. Indivíduos e interação entre eles mais que processos e ferramentas;
2. *Software* em funcionamento mais que documentação abrangente;
3. Colaboração com o cliente mais que negociação de contratos;
4. Responder a mudanças mais que seguir um plano.

Estes geraram 12 (doze) princípios (BECK *et al.*, 2001):

1. Nossa maior prioridade é satisfazer o cliente, através da entrega adiantada e contínua de *software* de valor;
2. Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adéquam a mudanças para que o cliente possa tirar vantagens competitivas;
3. Entregar *software* funcionando com frequência, na escala de semanas até meses, com preferência aos períodos mais curtos;
4. Pessoas relacionadas a negócios e desenvolvedores devem trabalhar em conjunto e diariamente, durante todo o curso do projeto;

5. Construir projetos ao redor de indivíduos motivados, dando a eles o ambiente e suporte necessário e confiar que farão seu trabalho;
6. O Método mais eficiente e eficaz de transmitir informações para, e por dentro de um time de desenvolvimento, é através de uma conversa cara a cara;
7. *Software* funcional é a medida primária de progresso;
8. Processos ágeis promovem um ambiente sustentável. Os patrocinadores, desenvolvedores e usuários, devem ser capazes de manter, indefinidamente, passos constantes;
9. Contínua atenção à excelência técnica e bom *design* aumentam a agilidade;
10. Simplicidade: a arte de maximizar a quantidade de trabalho que não precisou ser feito;
11. As melhores arquiteturas, requisitos e projetos emergem de times auto organizáveis;
12. Em intervalos regulares, o time reflete em como ficar mais efetivo; então, se ajusta e otimiza seu comportamento de acordo com esta reflexão.

2.2.1. SCRUM

Schwaber e Sutherland (2013) definiram o SCRUM como uma ferramenta ágil que permite o uso de diversos processos e técnicas para desenvolver e manter produtos complexos, entregando produtos com mais alto valor possível para o cliente. Como as outras propostas ágeis, o SCRUM não é um processo ou uma metodologia para construir produtos; é uma ferramenta, que permite empregar outros processos ou técnicas (RUBIN, 2012, p. 13).

Conforme Schwaber e Sutherland (2013), o SCRUM baseia-se em três princípios:

1. Transparência: Compartilhar aspectos com todas as partes envolvidas do projeto a fim de se ter o mesmo entendimento;
2. Inspeção: Examinar os artefatos do SCRUM e detectar variações;
3. Adaptação: Ajustar aspectos fora dos limites aceitáveis que possam interferir na qualidade do produto final.

Em suma, transparência, inspeção e adaptação são os conceitos que auxiliam a definição do ciclo PDCA (Singhal, 2014), como pode ser visualizado na Figura 3 a seguir.

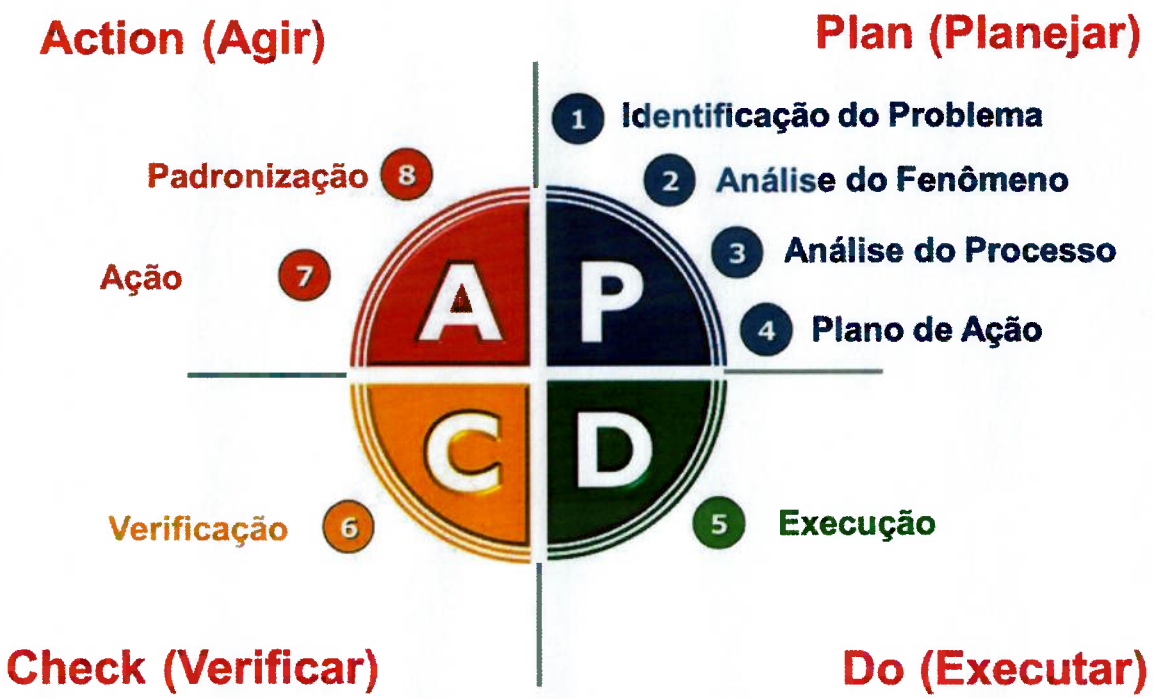


Figura 3 – Visão geral do ciclo PDCA (das fases e etapas)
Fonte: BEZERRA (2014)

2.3. Teste de sistema no ambiente de desenvolvimento ágil

Em um ambiente de desenvolvimento ágil, a forma como o teste será realizado é diferente do modelo tradicional. O analista de teste será integrado à equipe de desenvolvimento, onde terá os mesmos direitos que a equipe de desenvolvimento. Irá participar dos planejamentos para codificar uma história de usuário, realizará estimativa das atividades a serem entregues, assim como todas as outras atividades que cabem ao desenvolvedor. Não será mais um grupo de analistas apartado do time de desenvolvimento que reporta apenas os incidentes para os desenvolvedores.

O analista irá trabalhar em parceria com a equipe de desenvolvimento e a equipe de produto.

O foco do analista de teste não será mais de escrever apenas casos de testes, para quando o desenvolvedor terminar de codificar o sistema poder efetuar os testes. Nessa ocasião, o desenvolvedor irá codificar alguns testes de unidade e o analista de teste irá focar no teste de aceite automatizado e no plano de teste de regressão que irá executar automaticamente antes de cada lançamento do sistema. Algumas vezes, versões adicionais de cenários de teste poderão ser cobertas por testes exploratórios. Os cenários não serão contemplados mais em sua totalidade dentro de um escopo fechado.

A vantagem de analista de teste ágeis é: a resposta rápida e continua do andamento do desenvolvimento; redução de erros encontrados pelo cliente; Aumento da visibilidade das atividades relacionadas com os benefícios dos testes.

2.4. Custo da qualidade de sistema

Segundo Deming (1993), "Mais qualidade significa menos custos". Os custos da qualidade não são estabelecidos em sistema para obter-se a qualidade desejada, ou especificada. Referem-se à diferença entre os custos relativos à implantação e manutenção de tal sistema e custos decorrente da não qualidade, que resultam em produtos rejeitados em produção, produtos devolvidos pelos consumidores, custo de reparo, custo de retrabalhos, custos de garantia, receitas perdidas devida a perda de credibilidade do fornecedor, desgaste de imagem que precisa ser recuperada através de descontos e outros.

Dessa forma, o custo do processo de aquisição de qualidade é dividido em quatro partes: custos de prevenção, custos de avaliação, custos de falhas internas e externas. Diante disso, pode-se afirmar que os custos da prevenção são todos os custos envolvidos para evitar que um erro aconteça. O objetivo principal é controlar a qualidade do produto, considerando a revisão das histórias de usuário, planejamento e desenvolvimento do sistema orientado a teste ou a comportamento, revisão do produto, treinamento, controle e analise do processo, demonstrar resultados a partir de relatórios gerenciais, suporte ao usuário, entre outros. Os custos de avaliação são baseados na primeira experiência com o sistema, com a finalidade de detectar falhas, problemas de usabilidade e inconsistências antes que o produto seja lançado no mercado. Um exemplo de custo de avaliação são as

pesquisas de satisfação, ou testes de usabilidade, em que uma pessoa se candidata a utilizar o sistema e suas ações são gravadas. Qualquer erro no processo produtivo, como retrabalhos ou retestes, deve ser mitigado através dos custos relacionados às falhas internas. Por fim, os custos relacionados às falhas externas têm correlação com os erros encontrados pelo cliente, o que acarreta em perdas intangíveis, como destruição de imagem e confiabilidade da empresa.

Algumas empresas ainda não conseguem enxergar a redução desses custos como benefícios financeiros, conforme a Regra 10 de Myers (1979): "quanto mais cedo corrigir um erro, menos custoso será para empresa". Será utilizado como referência o artigo do Black (2000), onde se compara o custo da qualidade de um sistema em três situações diferentes:

1. Testes informais: realizados pelos desenvolvedores no momento da codificação;
2. Testes formais, manuais: realizados por profissionais capacitados, utilizando técnicas e processos bem definidos.
3. Testes formais, automatizados: realizados por profissionais capacitados, com auxílio de ferramentas de automatização, ou que conhecem linguagem de programação para codificar um teste.

A Tabela 1 a seguir reproduz as informações de Black (2000), onde podem ser observados os investimentos e os retornos obtidos.

Tabela 1 – Retorno sobre investimento de teste de sistema

Retorno Sobre Investimento (ROI)			
	Sem testes formais	**Testes Manuais**	**Testes Automatizados**
Estrutura de teste			
Pessoal	0	60.000	60.000
Infra-estrutura	0	10.000	10.000
Ferramentas	0	0	12.500
Total do Investimento	0	70.000	82.500
Defeitos encontrados pela Equipe de Desenvolvimento			
Defeitos encontrados	250	250	250
Custo dos Defeitos	2.500	2.500	2.500
Defeitos encontrados pela Equipe de Teste			
Defeitos encontrados	0	350	500
Custo dos defeitos	0	35.000	50.000
Defeitos encontrados em Produção			
Defeitos encontrados	750	400	250
Custo dos Defeitos	750.000	400.000	250.000
Custo da qualidade			
Conformidade (investimento em melhorias)	0	70.000	82.500
Não Conformidade (custo total dos erros encontrados)	752.500	437.500	302.500
Total do Custo da Qualidade	752.500	507.500	385.000
Retorno do investimento	NA	350%	445%

Fonte: Black (2000)

Os dados anteriores apresentados por Black (2000) são retificados em dólares, visando demonstrar o custo da qualidade de sistema e demonstrar que o retorno financeiro é obtido através da economia e não do montante investido.

2.5. Métricas: Características gerais sobre métricas, dificuldades e principais problemas em qualidade de sistema

Segundo Park, Goethert e Florac (1996), existem 4 (quatro) razões para medir um sistema: caracterizar, avaliar, prever e melhorar.

A utilização de métricas, embora importantes para medição e controle, não é um processo vastamente difundido (FENTON; NEIL, 1999). Esse fato pode ser atribuído à dificuldade da aplicação de métricas em sistemas muito grandes, ou o fato dos processos de medição nas empresas não serem maduros, além do alto custo dos programas de métricas (KANER; MEMBER; BOND, 2004).

O que pode justificar o porquê desta situação em muitas empresas é que os processos não são organizados ou não são maduros para o uso de medições (SOMMERVILLE, 2006).

2.6. Um estudo sobre como grandes empresas efetuam teste de sistema

O Google, como uma empresa inovadora e visionária, possui um perfil diferenciado de analistas de testes em relação às demais empresas. A empresa apresenta um perfil mais técnico de analista de teste, conhecidos como engenheiros de teste, ou seja, são profissionais com conhecimento básico em automação e programação de testes, foco em testes exploratórios e no produto fornecido para a empresa. Outro ponto interessante é que o engenheiro de teste é considerado um desenvolvedor desde o processo seletivo para a contratação de novos profissionais.

3. Estudo de Caso

3.1. Apresentação

O modelo proposto neste estudo foi aplicado em um modelo de gerenciamento de projeto de desenvolvimento de sistema, realizado em uma empresa de classificados de imóveis *online*.

3.2. Fase 1 – Planejamento

Inicialmente, não existia uma equipe de qualidade para suportar a equipe de desenvolvimento. Apenas uma pessoa era responsável por garantir a qualidade de todos os produtos fornecidos pela empresa. O processo de qualidade era praticamente inexistente e as atividades realizadas eram com base na experiência adquirida em empresas anteriores.

O teste era realizado de forma reativa, quando um problema era encontrado em produção. O custo da não qualidade não era medido. Era necessário tomar uma

decisão para que o processo de qualidade fosse modificado e a empresa entendesse a importância do processo de qualidade.

Apesar do processo de teste não ser bem definido, empregavam-se técnicas de testes para que se pudesse suprir no máximo um único produto da empresa, como incentivar o desenvolvimento a efetuar testes unitários, enquanto se efetuavam testes manuais exploratórios. O seu principal foco era automatizar testes de aceite, para aumentar a velocidade de resposta a erro do sistema desenvolvido.

3.2.1. Identificação do problema

Com base na ferramenta desenvolvida por Taiichi Ohno, responsável pela criação do Sistema Toyota, fizeram-se perguntas de modo a entender os problemas enfrentados pela empresa e corrigir a causa raiz (RANGARAJ, 2009)

1. *What:* Tempo elevado para corrigir um erro crítico em produção.
2. *Where:* Equipes de produto, desenvolvimento e qualidade que são as principais responsáveis pelo processo.
3. *When:* Durante o desenvolvimento de uma nova funcionalidade ou um novo projeto.
4. *Why:* O custo com a manutenção no código é elevado e quando há uma grande dúvida técnica, não é possível dar manutenção, sendo necessário refazer o projeto.
5. *Who:* Parte do código que tem taxa de manutenção muito baixa; Complexidade ciclomática[1] elevada; Erros ou alertas indicados pela ferramenta *Code Analysis*; Baixa ou inexistência de teste.
6. *How*: Falta de seguimento e definição clara do processo de qualidade.
7. *How much*: Custo produtivo 1000 (mil) vezes maior do que na criação do projeto.

3.2.2. Análise do processo

[1] Complexidade ciclomática (ou complexidade condicional) é uma métrica de software usada para indicar a complexidade de um programa de computador. Desenvolvida por Thomas J. McCabe em 1976, ela mede a quantidade de caminhos de execução independentes a partir de um código fonte. (MCCABE, 1976)

Na Figura 4 a seguir é possível visualizar o processo da empresa para criar, desenvolver e sustentar um *site*.

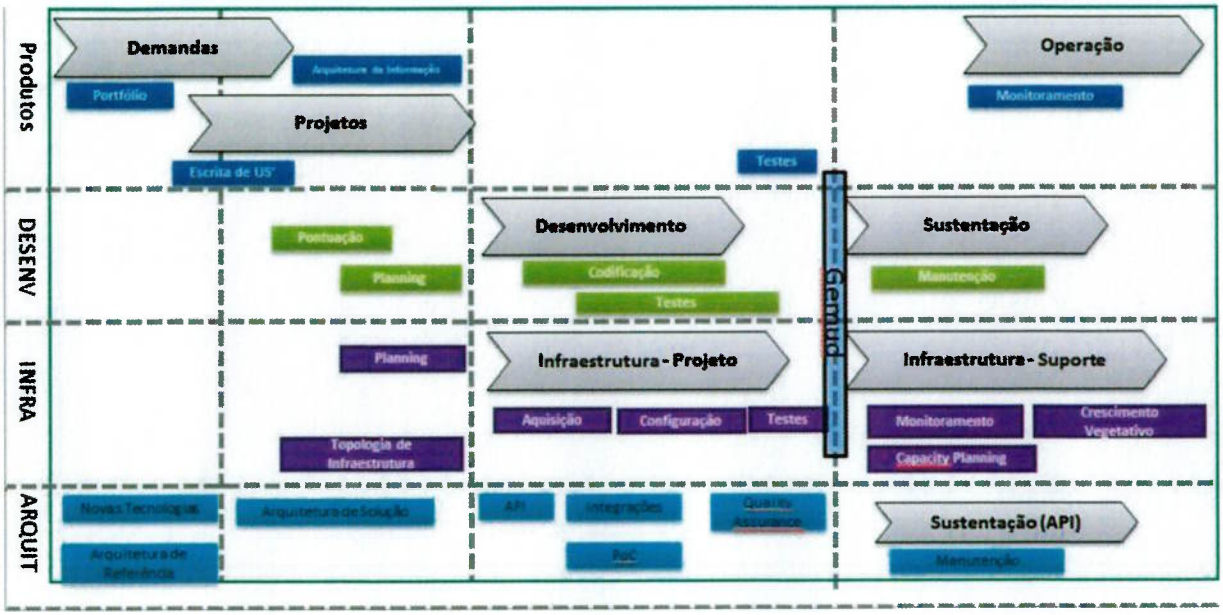


Figura 4 – Processo da empresa

A demanda se inicia com a reunião de portfólio, onde as principais pessoas responsáveis pelo processo discutem sobre as novas tendências de mercado e sobre novas funcionalidades, assim como novos projetos. Em conjunto, a equipe de arquitetura de sistemas avalia novas tecnologias para o desenvolvimento ou aperfeiçoamento dos produtos. A equipe de arquitetura de sistemas é responsável por disponibilizar uma arquitetura referência para a equipe de desenvolvimento, ou seja, um modelo no qual a equipe de desenvolvimento irá se basear para construir o código que irá gerar a nova funcionalidade.

Após a viabilização do projeto, inicia-se a redação da história de usuário, com base na necessidade do cliente levantada na reunião de portfólio.

Logo após a redação da história é elaborado um desenho para suporte da equipe de desenvolvimento, assim como é estimado o esforço e planejada a forma de como a funcionalidade será desenvolvida. Em paralelo, ocorrem atividades similares na infraestrutura. A equipe de arquitetura de sistemas, quando necessário, desenvolve uma arquitetura de solução que facilitará a projeção do código, demonstrando a melhor forma de como o código deve ser implementado.

O desenvolvimento só começa depois de todas as atividades serem definidas. Nesse momento se inicia a codificação tanto da parte visual do *site* como da parte de operação do *site*.

Após o desenvolvimento ocorrem testes para validar se os requisitos de negócios e as especificações do cliente foram atendidas.

Enquanto isso, a equipe de infraestrutura prepara o ambiente para a publicação do *site*.

Já a equipe de arquitetura de sistemas cuida apenas das integrações das partes do código com alguma outra ferramenta que irá dar suporte ao que foi desenvolvido pela equipe de desenvolvimento.

Quando o *site* está em produção ocorre o monitoramento e a sustentação do mesmo para que, em caso de necessidade, alguma medida possa ser tomada o mais breve possível.

3.2.3. Análise do fenômeno

Na figura 5 a seguir pode ser visualizada a qualidade do código gerado pela equipe de desenvolvimento. A análise sobre os alertas identificados pela ferramenta *Code Analysis*, que identifica a violação das regras de programação, demonstra a quantidade de alertas que poderiam ser evitados, pois são passíveis de gerarem erro em produção.

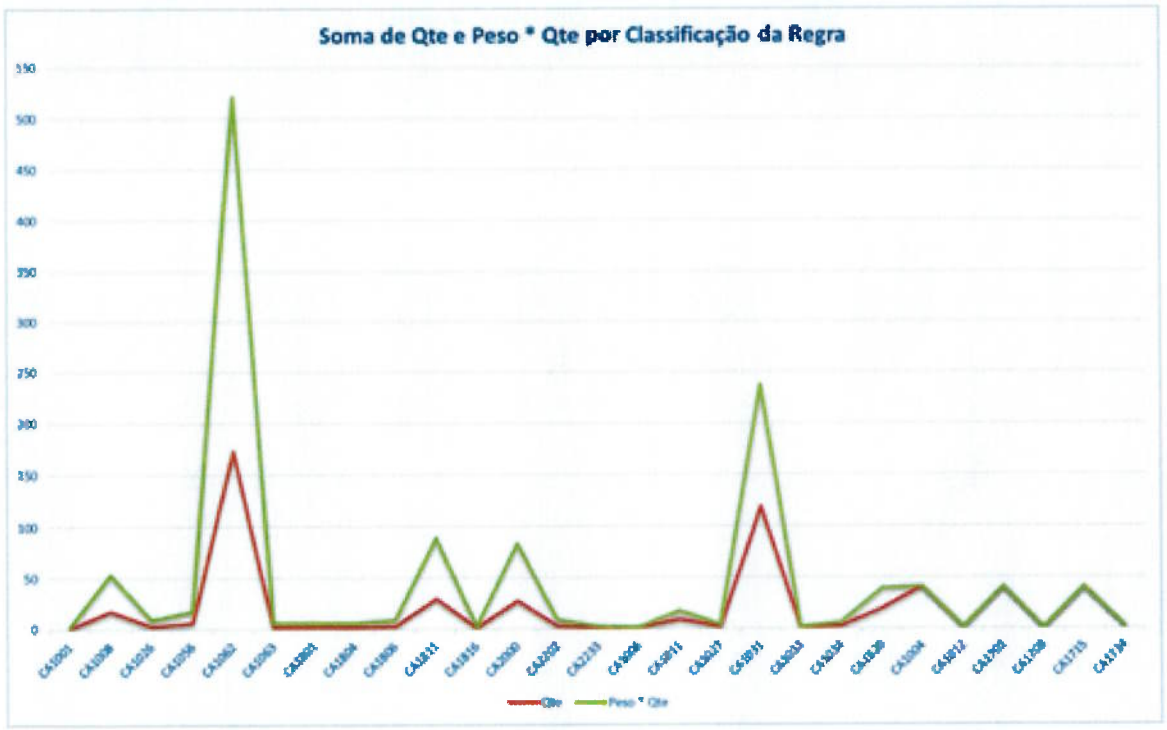


Figura 5 – Análise da qualidade do código desenvolvido entre as iterações 126 e 145 (2014)

A falta de manutenção no teste unitário, que tem foco em garantir o funcionamento da regra de negócio, afeta a prevenção de teste de regressão. Na Figura 6 a seguir pode ser visualizada a análise dos testes unitários efetuados nas aplicações que sustentam o sistema da empresa.

Figura 6 – Análise dos testes unitários durante as iterações 126 e 145 (2014)

Na Figura 7 a seguir pode ser visualizado o levantamento de cobertura de código em algumas aplicações.

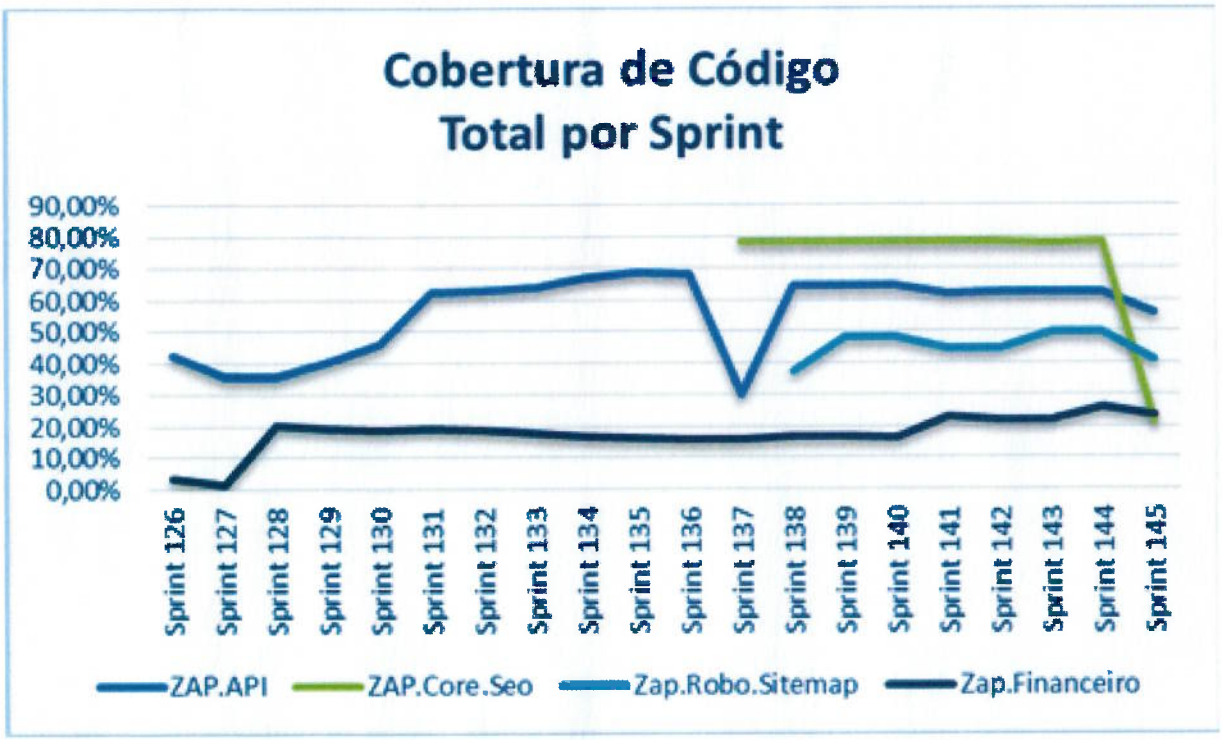


Figura 7 – Análise da cobertura de código durante as iterações 126 e 145 (2014)

A cobertura de código está diretamente ligada à quantidade de testes unitários construídos por aplicação. A baixa ou ausência da cobertura de teste no código produzido ao desenvolver o produto pode refletir na qualidade final do produto. As consequências da não qualidade de sistema afetam diretamente o cliente interno e externo da empresa. Abaixo estão elencados alguns pontos afetados pela não qualidade:

- Cliente insatisfeito;
- Reclamação efetuada em outros *sites*;
- Processo na justiça;
- Custo com bonificação;
- Depreciação de imagem;
- Custo 1000 (mil) vezes maior do que na primeira fase de implantação do projeto;
- Diminuição da margem de concorrência, permitindo que em eventuais momentos o concorrente possa obter fatia do público pagante;

- Diminuição na capacitação de *leads*[2]; etc.

3.2.4. Plano de ação

Definição e implantação de processos de testes, sejam manuais ou automatizados, processo de gestão de defeitos para reduzir em 70 (setenta) por cento a ocorrência de erros no ambiente de produção.

Na Figura 8 a seguir podem ser visualizados os erros gerados por ambiente, antes de aplicar e poder visualizar os benefícios de um sistema de gestão da qualidade.

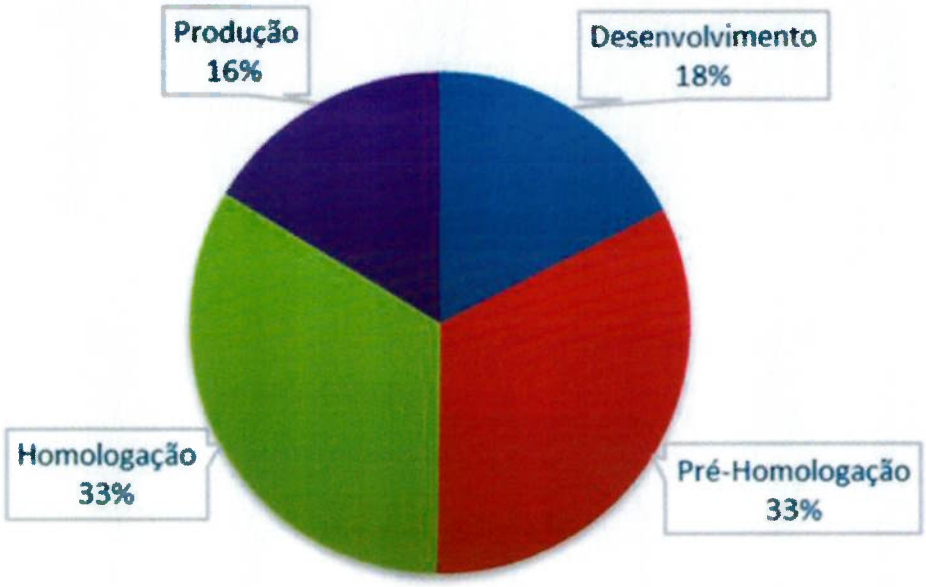


Figura 8 – Análise de erros por ambiente durante as iterações 126 e 145 (2014)

3.3. Fase 2 – Execução

3.3.1. Apresentação do processo da empresa

Atualmente o processo possui 3 (três) atividades antes da reunião de portfólio, com o propósito de trazer novos projetos mais elaborados e assertivos que representem mais valores para a empresa.

Na Figura 9 a seguir é possível visualizar o novo processo da empresa para criar, desenvolver e sustentar um *site*.

---

[2] *Lead*: Tornou-se sinônimo de qualquer visitante que informe seus contatos em troca de algum tipo de conteúdo, muito utilizada em empresas digitais. Fonte: http://marketingdeconteudo.com/o-que-e-lead/ Acesso: 3 de maio de 2016

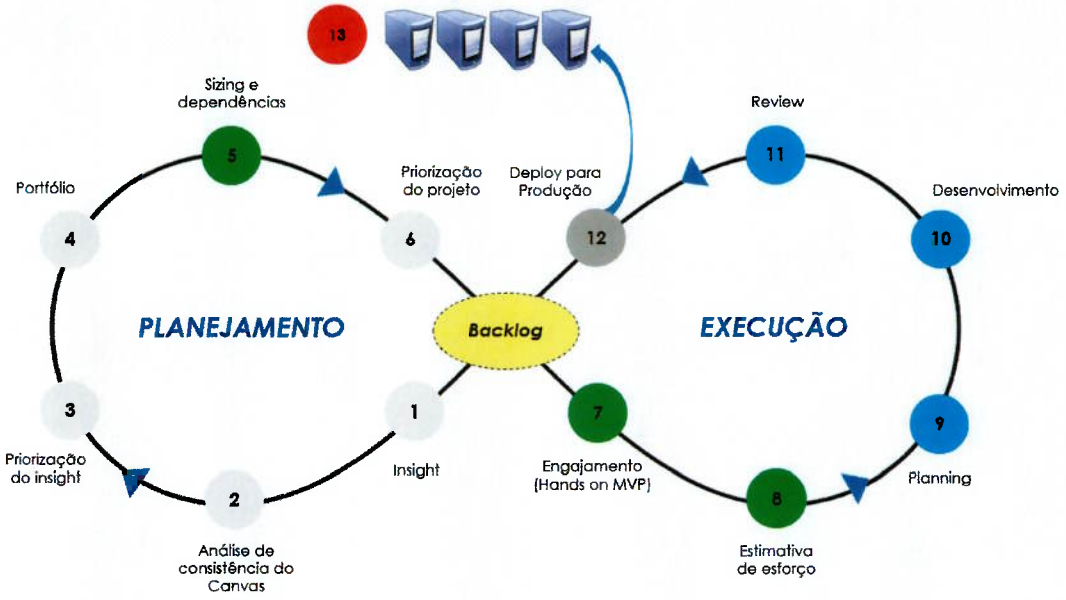


Figura 9 – Novo processo da empresa

Durante as etapas de pré-desenvolvimento, que são representadas pelas etapas 5 (cinco), 7 (sete) e 8 (oito), a equipe de qualidade de *software* é responsável pelas atividades:

- Identificar cenários críticos e repetitivos de cada Mínimo Produto Viável (MVP);
- Elaborar critérios de aceite das histórias;
- Definir cenários de teste;
- Identificar requisitos não funcionais;
- Garantir aderência à técnica *Behavior Driven Development* ou, em tradução livre: Desenvolvimento Guiado por Comportamento (BDD);
- Apontar candidatos à automatização.

Durante as etapas de desenvolvimento, que são representadas pelas etapas 9 (nove),10 (dez) e 11 (onze), a equipe de qualidade de *software* é responsável pelas atividades:

- Automatizar casos de teste;
- Avaliação das entregas;
- Identificar e endereçar erros de escopo;

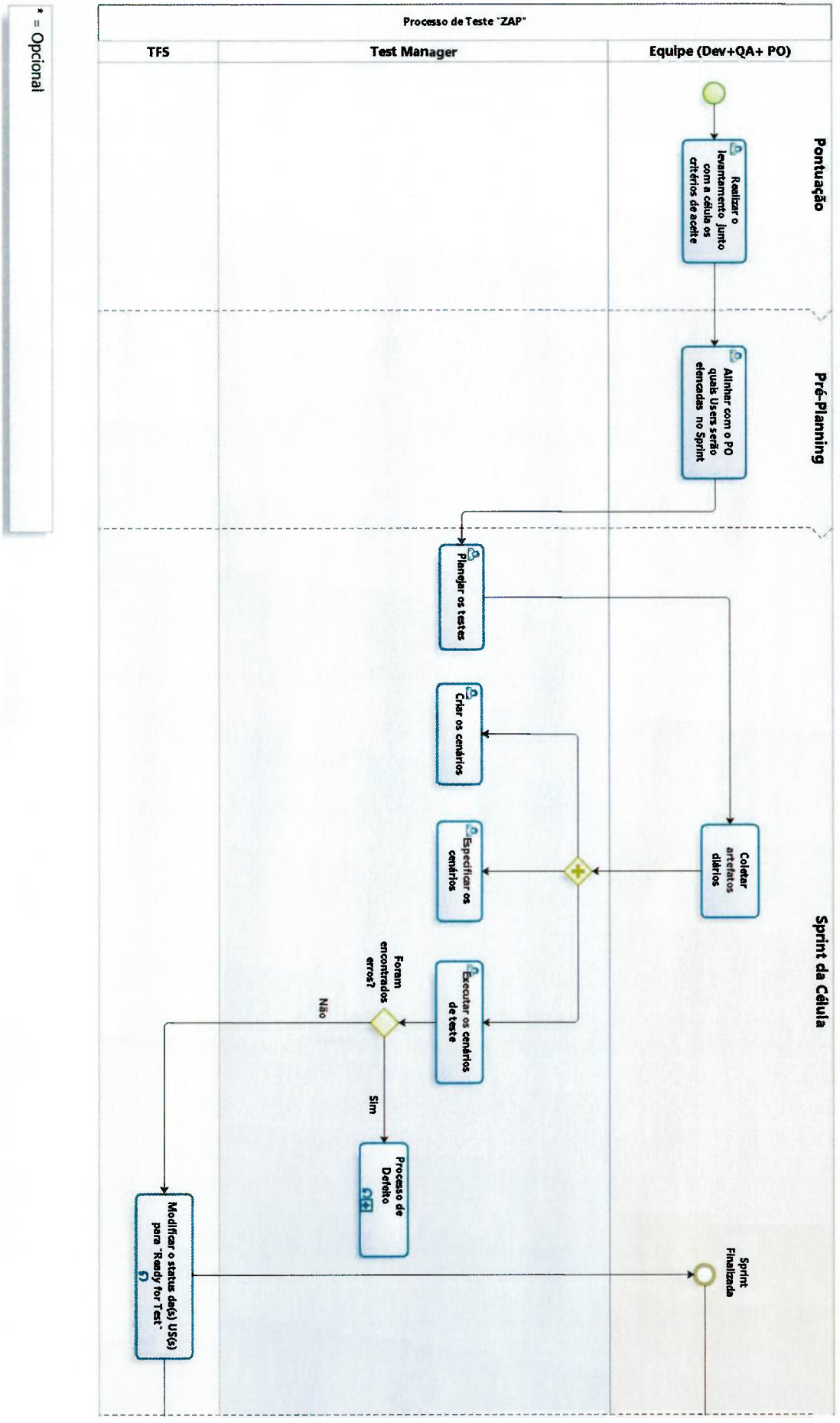
- Acompanhar a qualidade das versões dos projetos entregues;
- Acompanhar indicadores de qualidade;

Por fim, durante a entrega das atividades no ambiente de produção e as atividades do dia a dia, representadas pelas etapas 12 (doze) e 13 (treze), a equipe de qualidade de *software* é responsável pelas atividades:

- Homologação das entregas para o ambiente de produção;
- Identificar e endereçar erros de escopo;
- Avaliar requisitos não funcionais;
- Acompanhar performance dos produtos;
- Acompanhar os indicadores de erros;
- Estudar e apresentar melhores práticas, metodologias e ferramentas;
- Desenvolver e executar testes exploratórios.

3.3.2. Apresentação do processo de teste

Consiste na criação de um processo de testes com o objetivo de definir as etapas, atividades, artefatos, papéis e responsabilidades, proporcionando o controle do ciclo de teste dentro do modelo de desenvolvimento ágil. Na Figura 10 a seguir pode ser visualizado o processo de teste da empresa de forma explicita.


* = Opcional
Processo de Teste "ZAP"
TFS
Test Manager
Equipe (Dev+QA+ PO)
Pontuação
Realizar o levantamento junto com a célula os critérios de aceite
Pré-Planning
Alinhar com o PO quais Users serão etencadas no Sprint
Planejar os testes
Criar os cenários
Especificar os cenários
Executar os cenários de teste
Foram encontrados erros?
Não
Sim
Processo de Defeito
Modificar o status da(s) US(s) para "Ready for Test"
Coletar artefatos diários
Sprint da Célula
Sprint Finalizada

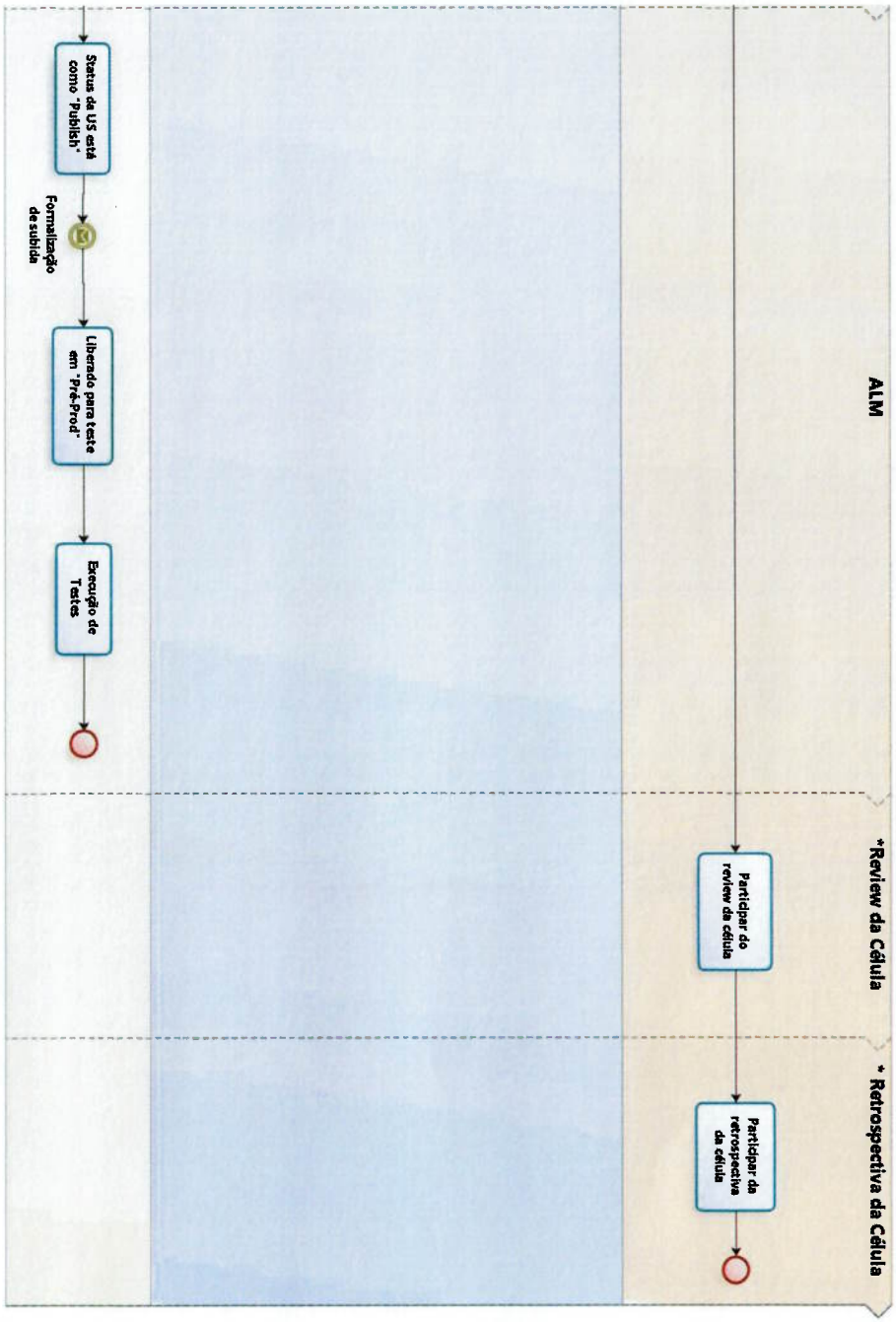


Figura 10 – Processo de teste

O processo é composto por:

- Entradas do processo: Histórias do usuário, escritas pelo *Product Owner* e o Gerente de teste.
- Papéis envolvidos: Membros do time (Analista de Qualidade, Desenvolvedores) e *Scrum Master.*
- Saídas do processo: Cenários de testes desenvolvido pelo analista de qualidade, e sistema testado por todos os membros integrantes do time.

A Tabela 2 a seguir representa a relação entre Seis Sigma e Scrum.

Tabela 2 – Six Sigma vs Scrum

Six Sigma		Scrum
Champion	⇔	Product Owner
Black Belt	⇔	ScrumMaster
Green Belt	⇔	Team Member

Fonte: FILHO, 2010

3.3.3. Definição de métricas

- Cobertura de implementação

$$Cobertura\ de\ implementação\ = \frac{Casos\ de\ testes\ implementados}{Total\ de\ casos\ de\ teste}$$

- Cobertura de testes automatizados
  - Testes unitários
  - Testes de aceite de interface do usuário

$$Cobertura\ de\ automatização\ = \frac{Total\ de\ casos\ de\ testes\ automatizados}{Total\ de\ casos\ de\ teste}$$

- Cobertura de criticidade
  - Alta criticidade;
  - Média criticidade;
  - Baixa criticidade.

- Cobertura de execução

$$Cobertura\ de\ execução\ = \frac{Total\ de\ casos\ de\ testes\ executados}{Total\ de\ casos\ de\ teste\ implantados}$$

- Cobertura de eficiência

$$Eficiência\ da\ Verificação = \frac{Total\ de\ erros\ da\ validação\ +\ Total\ de\ erros\ em\ produção}{Total\ histórias\ do\ usuário\ testadas}$$

- Eficiência da validação

$$Eficiência\ da\ Validação = \frac{Total\ de\ erros\ da\ validação}{Total\ de\ erros\ da\ validação +\ Total\ de\ erros\ em\ produção}$$

3.4. Fase 3 – Verificação

Com a criação do processo de teste, ocorreram mudanças no processo de como a empresa cria e desenvolve seus produtos. A forma como a qualidade é avaliada na empresa modificou-se bastante desde o início do projeto e a importância dada ao processo de qualidade evoluiu positivamente.

Atualmente, a quantidade de ambientes existentes diminuiu e o ambiente de pré-produção não faz mais parte do processo de qualidade, o que não impacta na avaliação de quantidade de erros que ocorrem no ambiente de produção.

Entretanto a quantidade de equipes que trabalha nos produtos aumentou, juntamente com a expansão do negócio, o que pode afetar diretamente na avaliação e no objetivo de redução de erros no ambiente de produção.

Dessa forma, avaliando todas as equipes que hoje operam, podem ser visualizados os erros por ambiente conforme mostrado na Figura 11 a seguir.

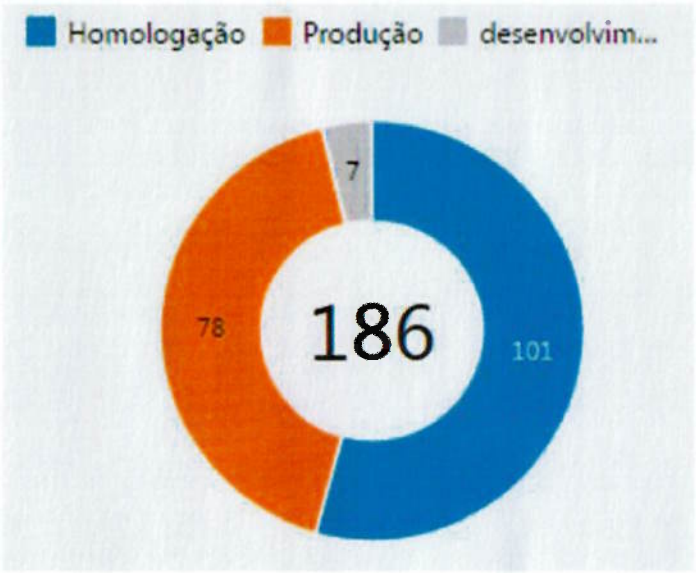


Figura 11 – Análise de erros por ambientes em 2016

Na Figura 12 pode ser visualizada a quantidade de erros de cada time por ambiente.

	desenvolvim.	Produção	Homologaç...	Total
DevRs		1	2	3
iOS		1	13	14
Backoffice I...		17	4	21
Android		6	19	25
Produtos Cl...		14	12	26
Produtos Pe...	7	3	17	27
Backoffice F...		20	14	34
Produtos A...		16	20	36
Total	7	78	101	186

Figura 12 – Análise de erros: ambiente por time em 2016

O aumento do número de equipes não afetou drasticamente a quantidade de erros colocados em produção.

Entretanto, não se obteve o resultado esperado, de redução de erros no ambiente de produção em 70 (setenta) por cento, mas de apenas 5 (cinco) por cento.

3.5. Fase 4 – Ação

Após aplicar as ações corretivas, que visam a correções das falhas encontradas durante o processo, é necessário repetir o ciclo de avaliação.

Entretanto, para que o ciclo seja efetivo, é necessário, criar um processo de capacitação e nivelamento de conhecimento entre as equipes, devido ao fato de que algumas equipes cometem mais erros no ambiente produtivo do que outras.

É importante instruir todas as equipes a criar um histórico de erros no ambiente de desenvolvimento quando necessário, com a finalidade de conseguir mensurar o custo da não qualidade e avaliar se o novo processo é eficiente.

4. Conclusão

Apesar da empresa utilizar uma metodologia ágil, é notável a percepção de como a escolha por trabalhar com uma ferramenta que permite a integração com outros processos e metodologia agrega valor ao negócio. Assim, é visível a importância da execução das etapas do PDCA dentro do processo de criação e desenvolvimento de sistema para alcançar a qualidade do produto final.

A partir da melhoria do processo, é possível identificar as equipes que precisam de mais suporte e analisar dentro de cada equipe a causa e o efeito da não qualidade, o que permite proporcionar a elas treinamento, compartilhando experiências e nivelando conhecimentos para obter um produto com mais qualidade.

Entretanto, a empresa precisa implantar mais testes automatizados e contratar profissionais que detenham esse conhecimento, de modo a reduzir a quantidade de atividades manuais ou com interferência humana, com isso possibilitando entregas com qualidade e de maneira mais rápida.

4.1. Limitações da pesquisa

O fato do processo de criação e desenvolvimento de produto da empresa ter mudado muito pode colocar em xeque as informações contidas no trabalho, apesar das lições aprendidas. Devido a limitação de tempo, não foi possível reavaliar o processo e aplicar todas as correções necessárias.

4.2. Trabalhos futuros

A continuidade desta pesquisa prevê acompanhar a nova metodologia, aplicar todas as correções, treinamentos necessários e reavaliar o ciclo para obter uma análise conclusiva, além de manter a melhoria continua do processo.

REFERÊNCIAS

BECK, Kent. [et al.] **Manifesto for Agile Software Development**. 2001. Disponível em: <http://www.agilemanifesto.org/>. Acesso em: 20 ago. 2016.

BEZERRA, Filipe. **Ciclo PDCA – Conceito e aplicação (Guia Geral)**. Disponível em: <http://www.portal-administracao.com/2014/08/ciclo-pdca-conceito-e-aplicacao.html>. Acesso em: 20 ago. 2016

BLACK, Rex. **Investing in Software Testing: The Cost of Software Quality**. Massachusetts: Computer Aid, Inc, 2000. Disponível em <http://www.compaid.com/caiinternet/ezine/cost_of_quality_1.pdf>. Acesso em: 31 jul. 2011.

BRAZ, Alan. **Precisa-se de Projetos Scrum para Estudo de Caso.** São Paulo, 2011. Disponível em <https://alanbraz.wordpress.com/2011/05/17/precisa-se-de-projetos-scrum/>. Acesso em: 01 fev. 2016.

DEMING, W. E. **Dr. Deming O Americano que Ensinou a Qualidade Total aos Japoneses**. Rio de Janeiro: Record, 1993

FENTON, N. E.; NEIL, M. **Software metrics: success, failures and new directions.** J. Syst. Softw., Elsevier Science Inc., New York, NY, USA, v. 47, p. 149–157, July 1999.

FILHO, Heitor Roriz. **Can Scrum Support Six Sigma?** Scrum Alliance – Fevereiro, 2010. Disponível em:<https://www.scrumalliance.org/community/articles/2010/february/can-scrum-support-six-sigma> Acesso em: 03 mai. 2016.

KAMEL M., BEDIWI I. and AL-RASHOUD M, **Planned methodologies vs. agile methodologies under the pressure of dynamic market**. JKAU: Eng. Sci. p. 19-35, 2010.

KANER, C.; MEMBER, S.; BOND, W. P. **Software engineering metrics: What do they measure and how do we know?** 2004.

MCCABE, Thomas J. **A Complexity Measure**. IEEE Transactions on Software Engineering: 308–320, 1976.

PARK, R. E.; GOETHERT, W. B.; FLORAC, W. A. **Goal Driven Software Measurement - A Guidebook**. Pittsburgh, PA 15213: Software Engineering Institute, Carnegie Mellon University, 1996.

PRESSMAN, R. S. **Engenharia de Software**. 5º ed. Rio de Janeiro: McGrawHill, 2002.

_____._____. 6º ed. Rio de Janeiro: McGrawHill, p. 53-85, 2006.

RUBIN, Kenneth S. **Essential Scrum: A Practical Guide to the Most Popular Agile Process.** 1ª ed. Arbor, Michigan: Addison-Wesley Professional, 2012.

SCHWABER, K.; SUTHERLAND, J. **The SCRUM Guide™**. Harvard Business Review, Boston, IV, Disponível em: <http://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-us.pdf>. Acesso em: 20 ago. 2016

SOMMERVILLE, I. **Software Engineering: (International Computer Science Series)**. 8ª ed. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 2006.

SINGHAL, Keshav Ram. **Quality Concepts and ISO 9001:2008 QMS Awareness**. Disponível em: <http://iso9001-2008awareness.blogspot.com.br/2014/04/pdca-cycle.html>. Acesso em: 20 ago. 2016

RANGARAJ. **Supply Chain Management For Competitive Advantage.** Tata McGraw-Hill Education, 2009.

ZANATTA, Alexandre. **xScrum: Uma Proposta de Extensão do xScrum para Adequação ao CMMI**. Dissertação de Mestrado. Universidade Federal de Santa Catarina. Florianópolis: 2004.

BIBLIOGRAFIAS COMPLEMENTARES

BASTOS, A. [et al.] **Base de conhecimento em teste de software** - São Paulo: Martins, 2007 - 2. ed.

Douglas. **Desenvolvimento ágil não é mais uma alternativa, é a solução.** Rio de Janeiro, 2014. Disponível em http://www.dsconsultoria.com.br/blog/desenvolvimento/desenvolvimento-agil-nao-e-uma-alternativa-e-a-solucao. Acesso em: 01 fev. 2016.

MYERS, J. G.; BAFGETT, T.; SANDLER, C. **The art of *software* testing** - United States of America: Wiley, 1979 - 3. ed.

OHNO, Taiichi. **O Sistema Toyota de Produção: além da produção em larga escala** – Porto Alegre, 1997

PRIES, K. H., & Quigley, J. M. **Scrum Project Management** - Nova Iorque: CRC Press, 2011.

SOCCOL, Ana Paula; GOMES, Thiago Simões. **O custo da não-qualidade: um estudo de caso em uma empresa do ramo automobilístico.** Revista CEPPG – Nº 25 – 2/2011 – ISSN 1517-8471. Disponível em <http://www.portalcatalao.com/painel_clientes/cesuc/painel/arquivos/upload/temp/c4a929354dc7894cc1176f87db0ebfe2.pdf>. Acesso em: 01 fev. 2016.