Universidade Paulista - UNIP

Richard Dias Conde

PROCESSO DE ATUALIZAÇÃO DE DOCUMENTAÇÃO DURANTE E PÓS RELEASE

Limeira 2016

Universidade Paulista - UNIP

Richard Dias Conde

PROCESSO DE ATUALIZAÇÃO DE DOCUMENTAÇÃO DURANTE E PÓS RELEASE

Trabalho de conclusão de curso apresentado à banca examinadora da Faculdade UNIP, como requisito parcial à obtenção da Graduação em Ciências da Computação sob a orientação do professor Antônio Mateus Locci.

Limeira 2016

Richard Dias Conde

PROCESSO DE ATUALIZAÇÃO DE DOCUMENTAÇÃO DURANTE E PÓS RELEASE

Trabalho de conclusão de curso apresentado à banca examinadora da Faculdade UNIP, como requisito parcial à obtenção da Graduação em Ciências da Computação sob a orientação do professor Antônio Mateus Locci.

Aprovada em 12 de dezembro de 2016.

BANCA EXAMINADORA

_____

Prof. Dr. Nome completo

_____

Prof. Me. Nome completo

_____

Prof. Esp. Nome completo

DEDICATÓRIA

Dedico este trabalho a minha família e a Júlia minha namorada pelo apoio incondicional, ao meu professor orientador Antônio Mateus Locci pela motivação, ensinamentos e orientações e aos profissionais que estão todo dia ao meu lado buscando o mesmo objetivo para obtermos um serviço de maior qualidade sempre.

“Sempre superestimamos a mudança que ocorrerá nos próximos dois anos e subestimaremos a mudança que ocorrerá nos próximos dez anos. Não se deixe levar pela inação”.

(Bill Gates)

RESUMO

A metodologia Waterfall vem sendo substituída pelos métodos ágeis pouco a pouco, porém ainda existe a uma grande demanda de uso desta metodologia e, portanto, não podemos deixar de lado a forma como os testes devem ser reproduzidos.

O novo processo para atualização de documentação durante e pós release visa a melhoria de maneira significativa da qualidade da documentação atualizada devido a defeitos na qual, não apenas o código ou interface foram afetados para que no futuro não haja desperdício de tempo gasto em revisões de defeitos antigos afim de interpretar as especificações novamente e correr o risco de atrasar projetos devido a defeitos já resolvidos anteriormente.

Desenvolvimento de uma solução prática para gerenciar a qualidade de documentação durante a execução dos testes e atualização de documentação onde por motivos de divergências na funcionalidade pode ser necessária uma atualização de documentos. Nos processos atuais é comum que defeitos com alteração de documentação sejam pouco revisados e a documentação fira algum outro requerimento, causando defeitos futuros ou extensas investigações.

Palavra-Chave: Testes, Métodos Ágeis, Metodologia Waterfall, defeitos, análise de requisitos.

ABSTRACT

The Waterfall methodology is being replaced by agile methods little by little, but there is still a great demand for using this methodology and, therefore, cannot leave aside the way the tests are reproduced.
The new process of the requirement updates during and after the release of quality improvement looks for means of quality for the updated documentation due to defects in quality when not only the code or interface are affected, so there is no waste of time spent on reviews of old defects in order to interpret as new specifications and run the risk of delaying projects due to defects previously solved. Developing a practical solution to generate quality documentation during a test run and updating documentation for reasons of divergence in functionality can be a review of documents. In the current processes with which the change of documentation is little reviewed and documentation of some other requirement, causing future defects or extensive investigations.

Key Words: Tests, Agile Methods, Waterfall Methodology, defects, requirements analisys.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS

ERP – Enterprise Resource Planning, sistema de informação que integra todos os dados e processos de uma empresa em um único sistema.

ISO – International Standardization Organization, é uma entidade que congrega os grémios de padronização/normalização.

IEEE – Instituto de Engenheiros Eletricistas e Eletrônicos, é uma organização sem fins lucrativos dedicada ao avanço da tecnologia em beneficio da humanidade

SUMÁRIO

INTRODUÇÃO

Os métodos Ágeis vêm tomando conta do mercado com novas atualizações e melhorias para atender da melhor forma a organização dos times de desenvolvimento e melhorar a qualidade e velocidade de entrega dos projetos, porém uma grande quantidade de projetos resiste e continua utilizando a metodologia Waterfall pois atende as necessidades do projeto e não há interesse do cliente em alterar a forma com que o time trabalha, portanto ainda existe um mercado buscando melhorias para o método aumentando ainda mais sua qualidade.

É normal que dificuldades apareçam a cada nova release independentemente do método utilizado, porém existem problemas em comum que poderiam ser resolvidos com boas maneiras ou novos processos tornando a qualidade do sistema maior.

O método Waterfall é muito utilizado hoje em dia, houveram muitas mudanças para se adequar as novas realidades, novos profissionais e técnicas utilizadas no desenvolvimento do modelo e muito estudo para adequar o método para diversos projetos. Mas neste momento é importante ainda pensar em o que é possível melhorar para que a qualidade esteja sempre em progresso de melhoria, assim como a efetividade de entrega.

Para melhorar é preciso levantar questionamentos referentes a qualidade de serviço que está sendo entregue pois, entregar de maneira rápida e gerar vários defeitos no sistema não significa sucesso.

A expectativa de uma maior qualidade, menos defeitos e a extinção do retrabalho estão associadas ao sucesso esperado pelo cliente. Utilizando de muitos feedbacks, a experiência na área de qualidade de software e gerenciamento de defeitos e requerimentos é possível desenvolver soluções para melhorar os processos de qualidade e novas maneiras para que sem ferir a efetividade da entrega e das datas limite.

Desta forma será apresentado uma solução para o gerenciamento de defeitos e atualizações necessárias ao requerimento e as especificações de um

release que recebe mudanças em um sistema em andamento criado a partir da metodologia Waterfall.

1.1 Objetivo

O objetivo desse trabalho é melhorar a qualidade da documentação utilizada para o desenvolvimento de novas funcionalidades ou alterações na forma em que o sistema funciona, seja esta mudança nos requerimentos ou especificações de cada módulo.

Esta melhoria tende a esclarecer a documentação e evitar novas reuniões para discutir como o sistema funciona ou como um defeito alterou alguma funcionalidade do sistema, desta forma evitando ambiguidade para entender como o sistema deve ou não funcionar.

Para isso, um estudo foi levantado para entender qual o gasto de tempo em revisões de defeitos fechados em releases anteriores e quanto tempo era gasto em novas discussões e novas alterações no requerimento.

Ao final do estudo, é proposto o ajuste no processo de revisão de defeitos que alterem ou faz referência em como o sistema deve funcionar, para que os Designers, Desenvolvedores e Testers tenham uma visão clara da documentação sem precisar recorrer a defeitos fechados durante a alteração desta documentação.

1.2. Justificativa

Os estudos sobre a alteração da documentação afetada por defeito ou por brainstorms durante ou no fim de um mesmo release soluciona de forma mais eficaz dúvidas e sugere melhorias mais claras a documentação para que a qualidade da mesma seja de fácil entendimento e possa esclarecer quaisquer dúvidas ou referências a defeitos já fechados, diminuindo também o número de defeitos fechados como Working as Designed.

Com a maior qualidade e clarificação da documentação qualquer novo recurso conseguirá entender a especificação ou requerimento dos módulos e poder por si próprio buscar referências de defeitos já fechados, evitando abrir defeitos sem necessidade e parar outros recursos para longas discussões de como deveria funcionar o sistema.

2. SOBRE A QUALIDADE DE SOFTWARE

Com o crescimento no número de softwares independentes, sejam eles aplicativos para aparelhos celulares, televisões, computadores e até mesmo carros, algumas empresas de software ainda conseguem liderar as vendas de softwares e aplicativos mesmo havendo milhares de outros softwares gratuitos.

Não é de pouco tempo atrás que muitos desenvolvedores começaram a surgir com novas ideias para criar softwares e que conseguiriam fazer isso por conta própria já que é totalmente acessível um kit de desenvolvimento Java e uma licença de um entre vários compiladores.

Estamos vivendo uma realidade onde existe uma competição de novos aplicativos e softwares que fazem sucesso na internet, sejam eles pequenos jogos que geram milhares de dólares ao redor do mundo com pequenas transações (como comprar moedas e avançar rapidamente no jogo) ou com propagandas em seus aplicativos. Mas por que o ranking de softwares e aplicativos ainda são liderados por softwares de empresas maiores se alguns softwares gratuitos realizam as mesmas tarefas destas?

Uma das respostas de maior peso é a qualidade que estes softwares oferecem. Os softwares desenvolvidos por empresas maiores seguem processos rigorosos de qualidade e planejamento para que a melhor experiência seja entregue ao usuário e que seja possível continuar com novas atualizações ou novas versões destes mesmos para não sair do mercado e poder oferecer novidades ao usuário que mesmo sem interesse acaba se envolvendo com novas funcionalidades oferecidas pelo sistema.

Existem muitas formas de desenvolver um sistema, desde a forma mais simples onde um programador desenvolve um pequeno aplicativo para lista de compras até um ERP que exige muito detalhamento de documentação, muitas reuniões para definir alterações, se elas são ou não necessárias, quais os riscos e vantagens, e principalmente, qual será o lucro em realizar tais alterações.

Grandes empresas como a IBM e Microsoft desenvolvem sistemas gigantescos para clientes em todo o mundo. Estas empresas estão entre as de entrega de maior qualidade de serviço e são focadas em clientes de Telecomunicações, automotivos, entre outros. A qualidade do serviço tem impacto em muitas áreas, como segurança pois, um software desenvolvido para avisar de problemas médicos durante cirurgias, tecnologias de comunicação que não poderia falhar ao enviar um dado importante entre satélites e acabar em mãos que poderiam destruir a rede mundial.

A qualidade de software também está presente em uma das mais lucrativas áreas dos últimos tempos, os videogames. Esta área tem movimentado milhões de dólares todos os anos com milhares de novos jogos criados todos os anos. O último jogo da franquia Grand Theft Auto vendeu mais de 65 milhões de cópias em 3 anos, e a qualidade de software está totalmente envolvida nesse número pois o game chegou ao mundo com números baixíssimos de bugs aumentando ainda mais a promoção do jogo e ganhou prêmios pela qualidade entregue aos jogadores.

Esta é a importância da qualidade de software, satisfazer o cliente para que o produto principal tenha uma ótima performance, realize seus processos sem travamentos ou de forma incorreta. Desta forma o sistema será seguro, realizará exatamente o que o cliente precisa e terá abertura para melhorias futuras sem comprometer o sucesso do produto.

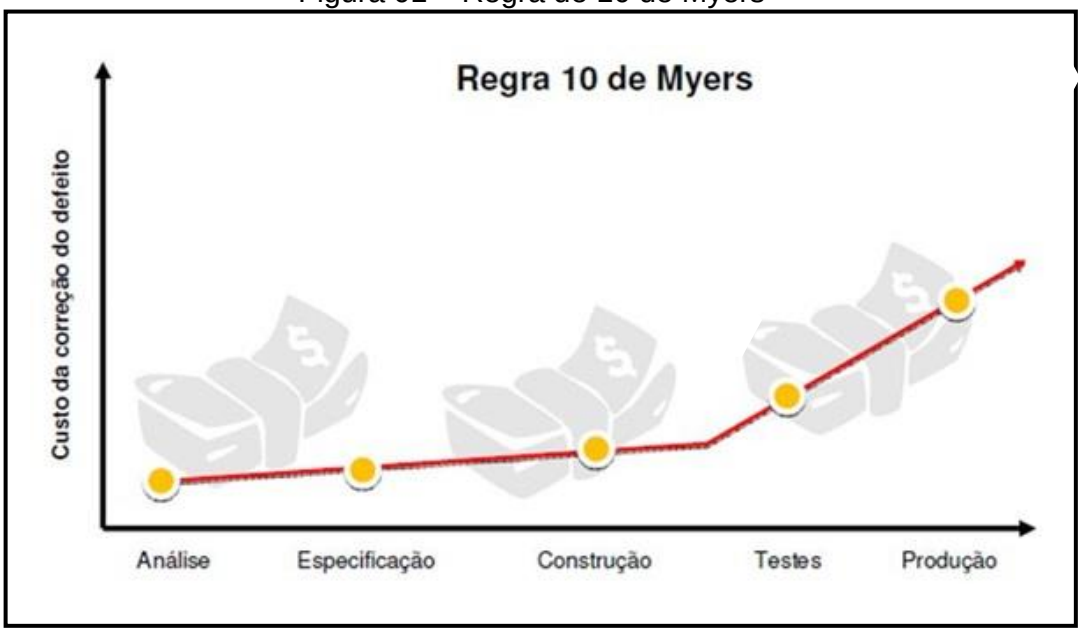
2.1 Qualidade no Software

Há não muito tempo atrás os softwares fossem eles simples ou mais complexos, eram criados e testados pelos próprios desenvolvedores, não

haviam testers para desenvolver cenários, pensar e repensar os testes, eram todos muito básicos e realizados apenas para validar o código, ou seja, não havia teste encima de o que o usuário iria fazer ou "e se" o usuário fizesse algo de maneira diferente ou até mesmo se o usuário utilizaria de forma incorreta. Com isso aumentaram as reclamações, os problemas que não tinham soluções no sistema e erros que viraram mais erros.

Até este momento algumas empresas fabricantes de software continuavam apenas no planejamento de investir seu dinheiro numa área de qualidade pois sabiam que isso iria consumir muito dinheiro, mas não tinham certeza dos benefícios que teriam em retorno pois seus softwares ficariam com certeza mais caros. Porém, como especificado na Regra de 10 de Myers, o custo de correção de defeitos tende a aumentar caso este defeito seja encontrado muito próximo a data de entrega.

Figura 01 – Regra de 10 de Myers



Fonte: http://www.galitezi.com.br/2012/02/teste-de-software-e-o-custo-da-nao.html

É possível notar que se o processo seguir uma linha em que se encontra os defeitos no início do processo de desenvolvimento, o custo pode ser minimizado e o impacto que pode causar será menor

O artigo “The cost of software quality” de Rex Black mostra através de uma tabela (Planilha 1) como poderiam ser estimados custos de empresas quando

não há testes em um processo formal, e sim apenas uma revisão no código, mostra também como seria com testes manuais e com testes automatizados.

Quadro 1 – Investimento em Testes: ROI Análise

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: https://thebugbox.wordpress.com/2011/07/10/custo-dos-testes-de-software-x-custo-do-defeito-de-software/

Desta forma é possível analisar o quanto o investimento vale a pena, qual o seu retorno e quanto poderá ser investido numa estrutura que dependendo da empresa pode ser mais simples e não necessitar um investimento tão alto e se adequar para que o investimento traga apenas lucro.

Mesmo que o teste não seja automatizado, e haja erros humanos, seria economizado muito pelo investimento na área de testes; caso seja feito um investimento na automação dos testes, as vantagens seriam enormes podendo retornar o valor investido de forma muito significativa.

Apesar de a automação parecer ser a peça chave do investimento, é preciso que profissionais capacitados estejam aptos a trabalhar com esta

automatização afim de melhorar scripts conforme as novas mudanças do sistema. Também não podemos esquecer que essas novas mudanças devem ser implantadas de forma organizada e processual para que a automação não seja configura de forma incorreta, o que acarretaria em novos problemas.

Este tópico de forma simples pode ser resumida em: Qualidade, Conhecimento e Lucro, pois através destes temos uma conclusão de que os testes de softwares são importantes e pode trazer não só um sistema estável como um sistema seguro e lucrativo.

2.1.1 Normas para Qualidade de Software

Para garantir que a qualidade de software esteja sendo realmente feita da maneira como foi estruturada para ser realizada, existem as certificações que comprovam as conformidades da estrutura de qualidade de software de forma mundial.

Existem padrões e normas para a certificação, ISO (International Organization for Standardization); IEEE (Instituto de Engenharia Elétrica e Eletrônica); ABNT (Associação Brasileira de Normas Técnicas) que garantem que as certificações sejam seguidas conforme a empresa padroniza seus processos.

Abaixo podemos ver o Quadro 02 que mostra as normas nacionais e internacionais para a correta avaliação da qualidade:

Quadro 02 – Normas

Norma	Comentário
Norma	Características da qualidade de produtos de software
NBR 13596	Versão brasileira da ISO 9126
ISO 14598	Guias para a avaliação de produtos de software, baseados na utilização prática da norma ISO 9126
ISO 12119	Características de qualidade de pacotes de software (software de prateleira, vendido com um produto embalado)

IEEE P1061	Standard for Software Quality Metrics Methodology (produto de software)
ISO 12207	Software Life Cycle Process. Norma para a qualidade do processo de desenvolvimento de software.
NBR ISO 9001	Sistemas de qualidade - Modelo para garantia de qualidade em Projeto, Desenvolvimento, Instalação e Assistência Técnica (processo)
NBR ISO 9000-3	Gestão de qualidade e garantia de qualidade. Aplicação da norma ISO 9000 para o processo de desenvolvimento de software.
NBR ISO 10011	Auditoria de Sistemas de Qualidade (processo)
CMM	Capability Maturity Model. Modelo da SEI (Instituto de Engenharia de Software do Departamento de Defesa dos EEUU) para avaliação da qualidade do processo de desenvolvimento de software. Não é uma norma ISO, mas é muito bem aceita no mercado.
SPICE ISO 15504	Projeto da ISO/IEC para avaliação de processo de desenvolvimento de software. Ainda não é uma norma oficial ISO, mas o processo está em andamento.

Fonte: unemat.br

Para um melhor entendimento da qualidade do produto e do processo, apenas a ISO 9126 e ISO 12207 serão utilizadas nas explicações a seguir que referenciam a qualidade de software.

A ISO é uma organização não governamental fundada em Genebra no ano de 1947 e, hoje está presente em 162 países tendo como principal finalidade promover a normatização de produtos e serviços, para que a qualidades seja permanentemente melhorada.

2.1.2 ISO 9126

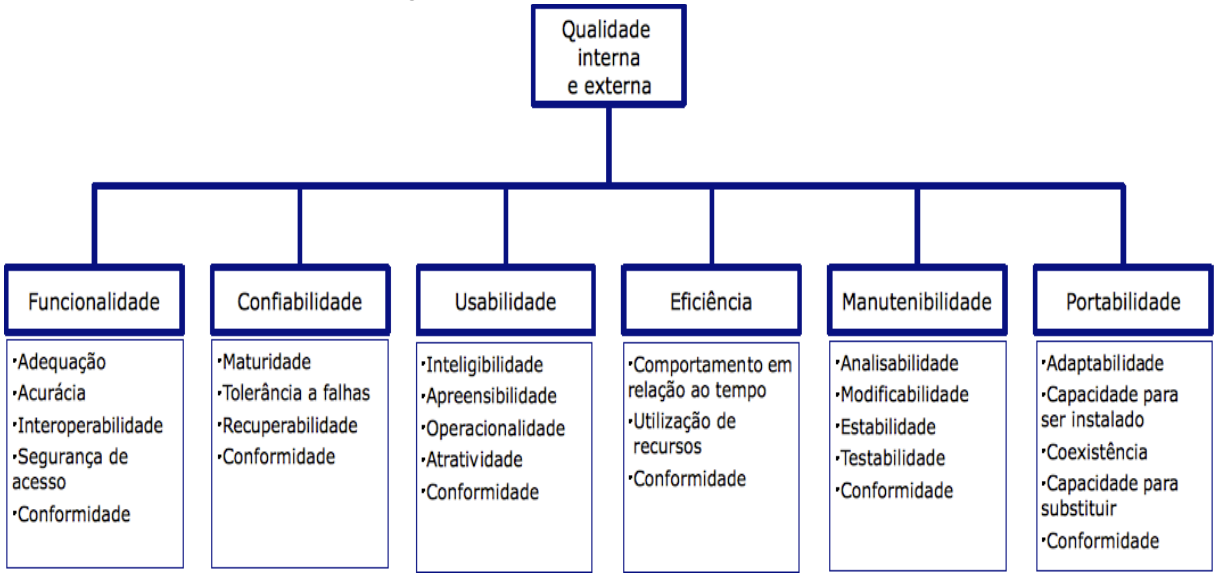
A ISO 9126 é uma norma para qualidade do produto de software sendo da família da norma 9000 que designa um grupo de normas técnicas estabelecendo um modelo de gestão de qualidade para organizações em geral.

Na ISO 9126 é estabelecido um modelo de qualidade com os seguintes componentes:

- Processo de desenvolvimento: cuja qualidade afeta a qualidade do produto de software gerado e é influenciado pela natureza do produto desenvolvido.

- Produto: compreendendo os atributos de qualidade do produto (sistema) de software. Estes atributos de qualidade conforme a figura 2 podem ser divididos entre atributos internos e externos. Estes se diferenciam pela forma de como são aferidos (interna ou externamente ao produto de software) e em conjunto compõem a qualidade do produto de software em si:

Figura 02 – Atributo de Qualidade



Fonte: https://pt.wikipedia.org/wiki/ISO/IEC_9126

- Qualidade em uso: que consiste na aferição da qualidade do software em cada contexto especifico de usuário.

Cada um dos atributos de qualidade do software possui uma característica e sub-caracteristica que deve ter nos sistemas criados pela empresa. Note que a sub-categoria “Conformidade” aparece em todos os atributos, no qual verifica o quanto o software está obedecendo aos requisitos de legislação e da padronização proposta pela ISO.

2.1.3 ISO 12207

A ISO 12207 define o processo do ciclo de vida dos softwares. Ela é a primeira norma internacional que descreve em detalhes os processos, atividades e tarefas que envolvem o fornecimento, desenvolvimento, operação e manutenção de produtos de software.

Os processos que essa norma detalha estão resumidos na tabela abaixo, no qual a metodologia Waterfall modelo V utiliza em todo o seu planejamento:

Quadro 03 – Processos do Ciclo de Vida do Software

Processos Fundamentais	**Início e execução do desenvolvimento, operação ou manutenção do software durante o seu ciclo de vida.**
Aquisição	Atividades de quem um software. Inclui: definição da necessidade de adquirir um software (produto ou serviço), pedido de proposta, seleção de fornecedor, gerência da aquisição e aceitação do software
Fornecimento	Atividades do fornecedor de software. Inclui preparar uma proposta, assinatura de contrato, determinação recursos necessários, planos de projeto e entrega do software.
Desenvolvimento	Atividades do desenvolvedor de software. Inclui: análise de requisitos, projeto, codificação, integração, testes, instalação e aceitação do software
Operação	Atividades do operador do software. Inclui: operação do software e suporte operacional aos usuários.
Manutenção	Atividades de quem faz a manutenção do software.
Processos de Apoio	**Auxiliam um outro processo.**
Documentação	Registro de informações produzidas por um processo ou atividade. Inclui planejamento, projeto, desenvolvimento, produção, edição, distribuição e manutenção dos documentos necessários a gerentes, engenheiros e usuários do software.
Gerência de Configuração	Identificação e controle dos itens do software. Inclui: controle de armazenamento, liberações, manipulação, distribuição e modificação de cada um dos itens que compõem o software.
Garantia de Qualidade	Garante que os processos e produtos de software estejam em conformidade com os requisitos e os planos estabelecidos.
Verificação	Determina se os produtos de software de uma atividade atendem completamente aos requisitos ou condições impostas a eles.
Validação	Determina se os requisitos e o produto final (sistema ou software) atendem ao uso específico proposto
Revisão Conjunta	Define as atividades para avaliar a situação e produtos de uma atividade de um projeto, se apropriado.
Auditoria	Determina adequação aos requisitos, planos e contrato, quando apropriado.
Resolução de Problemas	Análisar e resolução dos problemas de qualquer natureza ou fonte, descobertos durante a execução do desenvolvimento, operação, manutenção ou outros processos.

Processos Organizacionais	Implementam uma estrutura constituída de processos de ciclo de vida e pessoal associados, melhorando continuamente a estrutura e os processos.
Gerência	Gerenciamento de Processos.
Infraestrutura	Fornecimento de recursos para outros processos. Inclui: hardware, software, ferramentas, técnicas, padrões de desenvolvimento, operação ou manutenção.
Melhoria	Atividades para estabeler, avaliar, medir, controlar e melhorar um processo de ciclo de vida de software
Treinamento	Atividades para prover e manter pessoal treinado

Após esta breve explicação sobre as duas mais importantes ISOs envolvidas na qualidade de software podemos começar a tratar dos modelos de software que dispõem maneiras, processo e ordem para o desenvolvimento de um software.

3. METODOLOGIA DE TESTE DE SOFTWARE

3.1. Modelo V (V-Model)

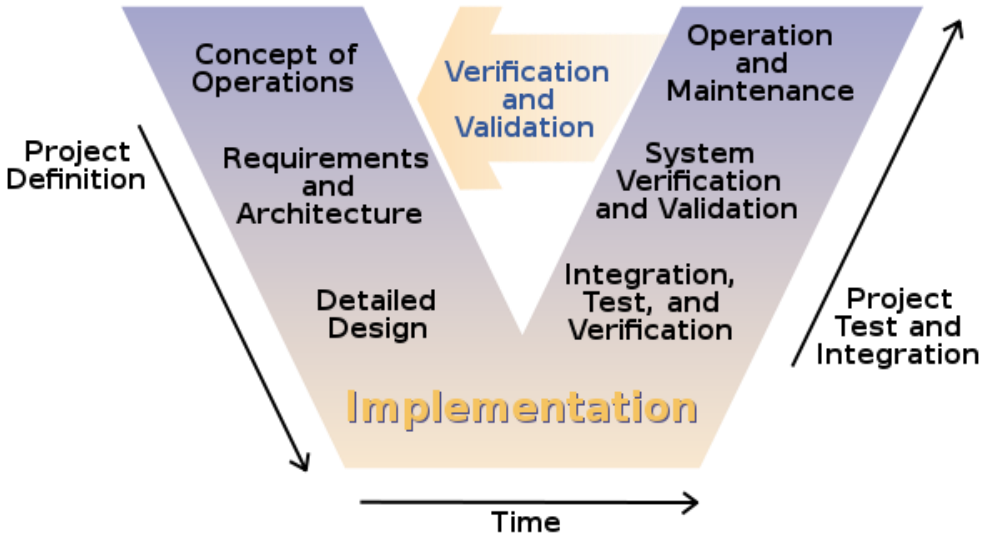


FIGURA 03 – MODELO V

O modelo V representa um processo de desenvolvimento de software (também aplicável ao desenvolvimento de hardware) que pode ser considerado uma extensão do modelo em cascata. Em vez de se mover para baixo de uma forma linear, as etapas do processo são dobradas para cima depois de a fase de codificação, para formar a forma típica V. O Modelo V demonstra as relações entre cada fase do ciclo de vida do desenvolvimento e da sua fase de teste associado. Os eixos horizontal e vertical representam o tempo ou a integralidade do projeto (da esquerda para a direita) e nível de abstração, respectivamente.

Fases de Testes:

- **Unit testing** – São testes feitos ainda na fase de codificação para evitar defeitos grotescos e/ou de conexões; são testes mais básicos.

- **Teste de Integração** - Planos de teste de integração são desenvolvidos durante a Fase de Projeto Arquitetônico. Estes ensaios verificam se as unidades criadas e testadas de forma independente podem coexistir e se comunicar entre si. Os resultados dos testes são compartilhados com a equipe do cliente.

- **Teste do Sistema** - Planos de testes do sistema são desenvolvidos durante Sistema fase de projeto. Ao contrário de Unidade e Planos de teste de integração, planos de teste do sistema são compostas por equipe do negócio do cliente. Teste de sistema garante que as expectativas de aplicação desenvolvida sejam atendidas. Toda a aplicação é testado quanto à sua funcionalidade, interdependência e comunicação. Teste do sistema verifica se os requisitos funcionais e não funcionais foram cumpridos. Carga e testes de desempenho, testes de stress, testes de regressão, etc., são subconjuntos de teste do sistema.

- **UAT - User Acceptance Testing (Teste de aceitação do usuário/cliente)** – Planos de testes são desenvolvidos durante a fase de Análise de Requisitos, são compostos por usuários de negócios. UAT é realizada num ambiente de utilização a que se assemelhe ao ambiente de produção, utilizando dados realistas. O time de UAT verifica que o sistema entregue satisfaz a exigência do usuário e o sistema está pronto para uso em tempo real.

3.2. Waterfall (Cascata)

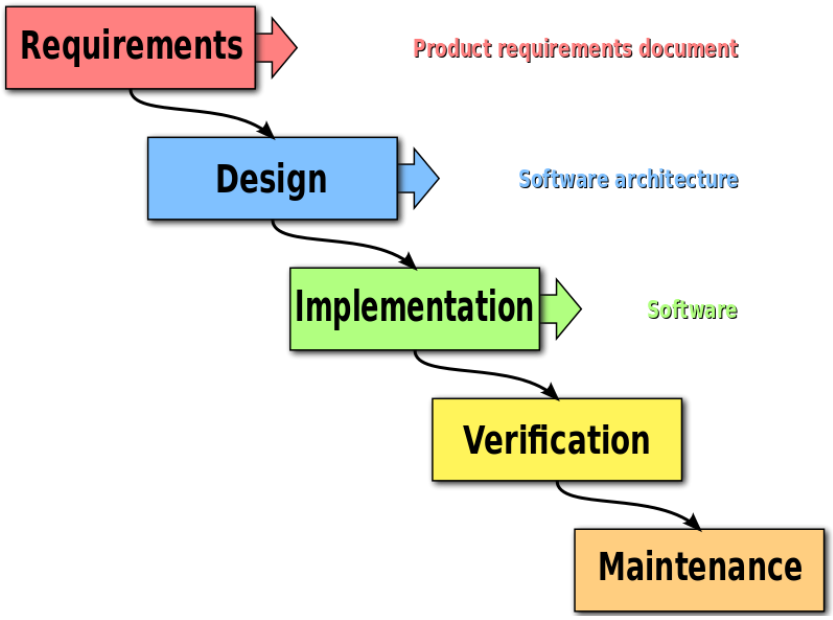


FIGURA 04 – MODELO WATERFALL

O modelo em cascata é um processo de projeto seqüencial, usado em processos de desenvolvimento de software, em que o progresso é visto fluindo de forma constante para baixo (como uma cachoeira) pelas fases de concepção, iniciação, análise, projeto, construção, teste, produção / implementação e manutenção.

O modelo de desenvolvimento em cascata origina nas indústrias de manufatura e de construção: altamente estruturado ambientes físicos em que, após o fato, as mudanças são proibitivamente caro, se não impossível. Desde os tempos em que metodologias de desenvolvimento de softwares não formais existiam, este modelo orientado para o hardware foi simplesmente adaptado para desenvolvimento de software.

O modelo original segue os passos:

1. Requerimentos do software;
2. Análise de resultados, regras de negócios;
3. Design;
4. Desenvolvimento do código;
5. Teste;
6. Operações (instalação, migração, suporte, etc).

3.3. Agile (Ágil)

FIGURA 05 – AGILE DEVELOPMENT

Desenvolvimento de software Agile é um grupo de métodos de desenvolvimento de software em que as soluções evoluem através da colaboração entre a auto-organização, equipes multifuncionais. Ele promove o planejamento adaptativo, o desenvolvimento evolucionário, entrega antecipada, melhoria contínua, e incentiva a resposta rápida e flexível para mudanças. O Manifesto para Desenvolvimento Ágil de Software, também conhecido como o Manifesto Ágil, foi proclamada pela primeira vez em 2001, seis anos depois de "Metodologia Ágil" foi originalmente introduzido pelos engenheiros de software mais proeminentes do final dos anos 80 e início dos anos 90 e saiu do DSDM Consortium em 1994.

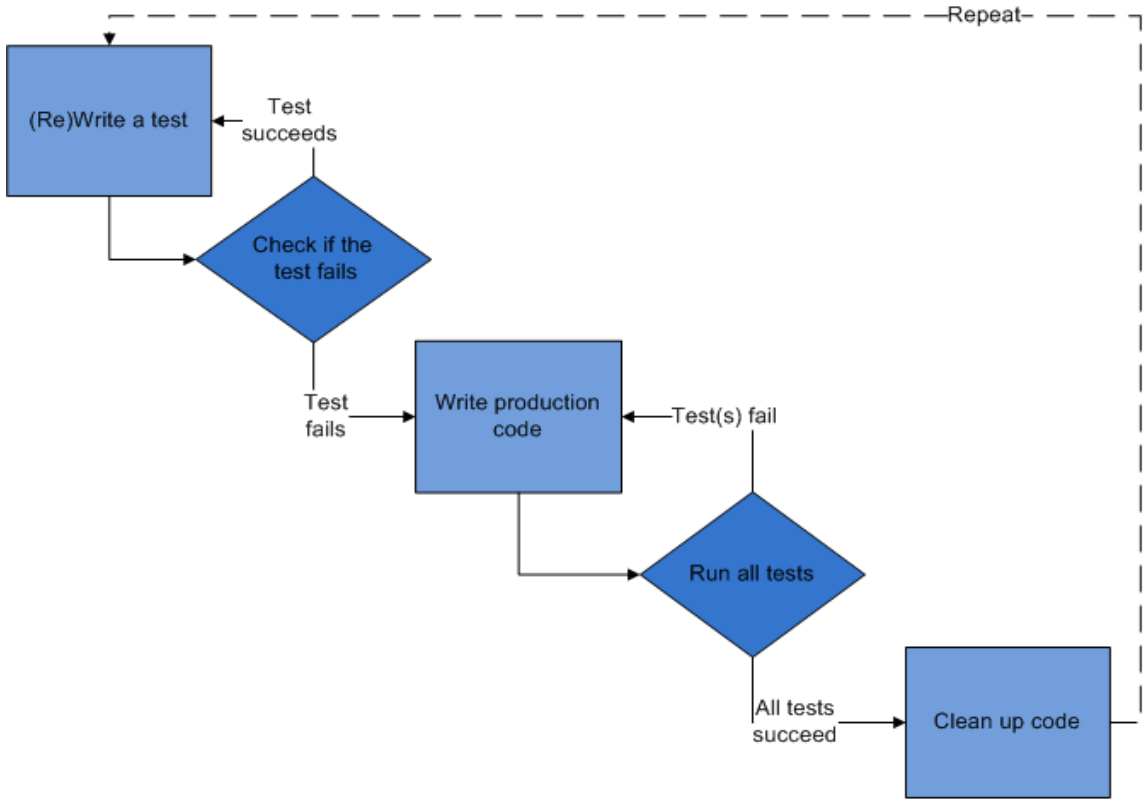


FIGURA 06 – TEST DRIVEN DEVELOPMENT

3.4. Scrum

SCRUM É UM FRAMEWORK DE GERENCIAMENTO DE PROJETO ÁGIL LEVE USADO PRINCIPALMENTE PARA DESENVOLVIMENTO DE SOFTWARE. ELE DESCREVE UMA ABORDAGEM ITERATIVA E INCREMENTAL PARA O TRABALHO DE PROJETO.

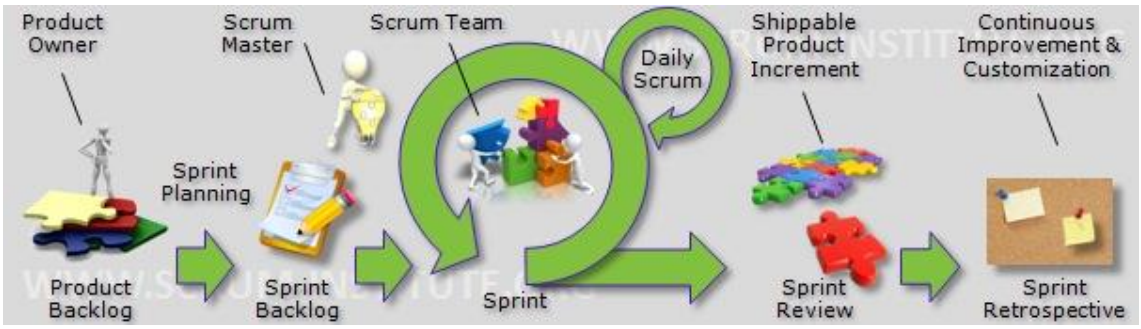


FIGURA 07 – SCRUM OVERVIEW

Scrum pode ser usado em todos os tipos de desenvolvimento de software: para desenvolver pacotes de software completos, para desenvolver apenas algumas partes de sistemas maiores, para projetos internos ou de clientes.

O Scrum Framework implementa as pedras angulares definidas pelo manifesto ágil:

- Indivíduos e interações sobre processos e ferramentas
- Software de trabalho sobre documentação abrangente
- Colaboração do cliente sobre negociação de contrato
- Respondendo a alterações ao seguir um plano

O Scrum Framework em si é muito simples. Ele define apenas algumas diretrizes gerais com apenas algumas regras, papéis, artefatos e eventos. No entanto, cada um destes componentes é importante, serve a um propósito específico e é essencial para uma utilização bem sucedida do quadro.

Os principais componentes do Scrum Framework são:

- Os três papéis: Scrum Master, Scrum Product Owner e Scrum Team
- Um Backlog priorizado contendo os requisitos do usuário final
- Sprints
- Eventos Scrum: Reunião de Planejamento Sprint (WHAT-Meeting, HOW-Meeting), Reunião Diária Scrum, Sprint Review Meeting, Sprint Retrospective Meeting

Importante em todos os projetos Scrum são a auto-organização e comunicação dentro da equipe. Não há mais um gerente de projeto em um sentido clássico. No Scrum Framework, o Scrum Master eo Scrum Product Owner compartilham suas responsabilidades. No entanto, no final, a equipe decide o que e o quanto eles podem fazer em uma dada iteração do projeto (Sprint).

Outro aspecto central no Scrum Framework é a melhoria contínua: inspecionar e adaptar. As equipes Scrum têm que inspecionar e avaliar freqüentemente seus artefatos e processos criados a fim de adaptá-los e otimizá-los. A médio prazo, otimizará os resultados, aumentará previsivelmente e, portanto, minimizará o risco geral do projeto.

O Scrum Framework tenta lidar com o fato de que os requisitos são susceptíveis de mudar rapidamente ou não são completamente conhecidos no início do projeto. Os requisitos de baixo nível só são definidos no momento em que eles vão ser realmente implementados. Em Scrum, as alterações e otimizações de produtos, requisitos e processos são parte integrante de todo o ciclo de engenharia.

Outra pedra angular do Scrum Framework é a comunicação. O Scrum Product Owner trabalha em estreita colaboração com a Scrum Team para identificar e priorizar a funcionalidade. Essa funcionalidade é anotada em histórias de usuário e armazenada em um Scrum Product Backlog. O Product Backlog consiste em tudo o que precisa ser feito para entregar com sucesso um sistema de software de trabalho.

A Equipe Scrum está habilitada a selecionar apenas as histórias de usuários que têm certeza de que podem terminar dentro de 2-4 semanas de Sprints. Como a equipe Scrum é autorizado a cometer seus próprios objetivos, eles serão mais motivados e trabalhar com o melhor desempenho possível. O Scrum Master é outro papel importante no Scrum Framework, já que funciona como um servidor-mestre com a Scrum Team. Suas principais tarefas são fazer com que a equipe do Scrum entenda como o Scrum opera, proteger a equipe Scrum de interrupções externas e remover os impedimentos que impedem a equipe Scrum de atingir sua máxima produtividade.

O Scrum Framework em sua forma simples é melhor usado para projetos menores, de uma equipe. Mas com a introdução de papéis adicionais como o "Chief Scrum Product Owner", também é útil em projetos de equipes multi-equipes e / ou distribuídas.

4. Processo para atualização de documentação

O modelo de processo para atualização de documentação se baseia no modelo Waterfall (cascata), porém todas as suas etapas voltam para os requerimentos para uma revisão e atualização de documentos, além de produzir feedbacks de cada etapa para uma melhoria do entendimento do projeto.

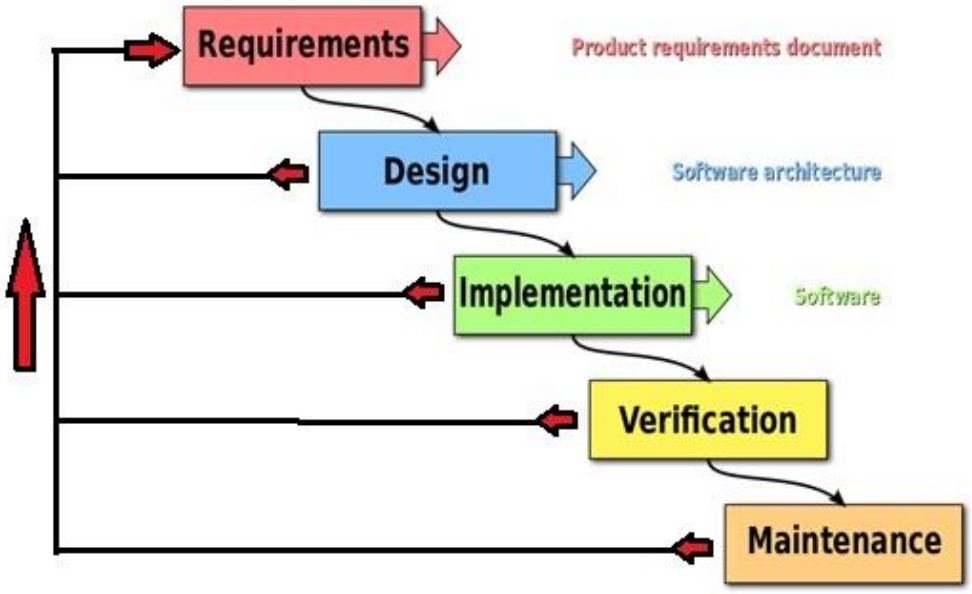


FIGURA 08 – MODELO DE PROCESSO DE ATUALIZAÇÃO DE DOCUMENTAÇÃO

A importância de uma documentação 100% correta é enorme, mas, além disso, os requerimentos podem ser melhorados e deixar menos brechas para interpretações que saem do entendimento principal do projeto nas últimas etapas.

CONCLUSÃO

O trabalho permitiu que um novo processo para a melhora na qualidade de documentação pudesse ser realizado de maneira que o custo não seja elevado e os padrões de ISO continuem a ser certificados. Dessa forma é possível aumentar ainda mais a produtividade no futuro com documentações mais ricas em informação e mais clarificadas.

A ideia do trabalho é mostrar que com novos processos são bem-vindos na área de testes de software e que existe uma variedade enorme na forma de trabalho que pode ser adequada para suportar a qualidade e manter de forma clara uma atualização para que qualquer futuro recurso tenha acesso e entendimento da documentação.

Me permitiu utilizar de muito conhecimento para criar uma solução prática em que eu pude compartilhar com meus colegas de trabalho para melhorar o problema de documentação que estava sempre tirando a atenção do time, o que atrapalhava de forma significativa a qualidade dos testes e dos scripts de automação.

Também me permitiu aprender mais sobre a área de testes e novas metodologias que podem adicionar novas ideias e novos processos para que diferentes times tenham a forma correta de trabalhar segundo a ISO. Desta forma posso me informar de novas certificações para aumentar meu conhecimento dentro da área de testes, me tornando um profissional mais qualificado na área.

A área de testes cresce mais a cada dia e, principalmente com a automação em alto crescimento e com grande visão, é necessário para os profissionais expandirem o conhecimento nas metodologias existentes e explorar novos caminhos para que a qualidade esteja sempre aumentando e não haja defeitos funcionando para o cliente.

Ainda há muito o que evoluir em relação ao processo como um todo, mas com uma ótima qualidade de software é possível encontrar os caminhos para construir um sistema mais protegido contra falhas onde o principal foco do sistema seja nas melhorias e não em ajustar problemas que não geram lucro nem novas funcionalidades ao sistema.

No geral, a área de testes é uma área muito importante para o desenvolvimento de software e que usando como base o novo processo por este trabalho descrito, seja possível aumentar de forma significativa o número de profissionais capacitados para utilizar de diversas maneiras novas técnicas e processos onde os futuros testers estejam aptos a utilizar o sistema sem necessidade de longas reuniões e aprendizados sobre o sistema.

REFERÊNCIAS BIBLIOGRÁFICAS

SHOKOYA Ade – **Waterfall to Agile.** Great Britain: TamaRe House, 2012.

WILLI, Renato ; MILANI, Fabiano – **Métodos Ágeis para Desenvolvimento de Software.** Porto Alegre: Bookman, 2014.

SITES

SCRUM ALLIANCE. Informações sobre Scrum. – Disponível em <https://www.scrumalliance.org/>. Acessado em: 25 de nov. 2016.

Instituto Brasileiro de Qualidade em Testes de Software – Disponível em <http://www.ibqts.com.br/>. Acessado em: 25 de nov. 2016.

TREINAMENTOS

- Cacillo, Daniel – Treinamento ITIL v3, IBM – Hortolândia

CERTIFICAÇÃO

Scrum Master Accredited Certification - International Scrum Institute™ - LICENSE: 96483321491163. Nov. 2016.