FACULDADES INTEGRADAS NORTE DO PARANÁ - UNOPAR
TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

WALDIR BORBA JUNIOR

PROJETO INTEGRADO II

PORTFOLIO INDIVIDUAL

COLOMBO
2021

WALDIR BORBA JUNIOR

PROJETO INTEGRADO II

Portfólio Interdisciplinar Individual, do Curso de graduação, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas – da Faculdades Integradas Norte do Paraná - UNOPAR – EAD Colombo, como requisito para conclusão de curso.

Orientador: Fernanda Caroline da Silva Fernandes
UNOPAR - EAD

Coorientadores: Adriane Aparecida Loper
UNOPAR - EAD

Gilberto Fernandes Junior
UNOPAR - EAD

Leonardo Santiago Sidon da Rocha
UNOPAR - EAD

Vanessa Matias Leite
UNOPAR - EAD

COLOMBO
2021

SUMÁRIO

1 INTRODUÇÃO

Este Projeto Integrado, tem a finalidade de consolidar os conhecimentos adquiridos nas interdisciplinas do curso de Análise e Desenvolvimento de Sistemas.

Como objeto deste trabalho, estão as atividade de qualidade de software, conceitualização de processos e *thread*, segurança e auditoria bem como os tipos de testes de *software*.

2 QUALIDADE - INTRODUÇÃO

Existem muitas definições de qualidade de software propostas na literatura, sob diferentes pontos de vista. Qualidade é um termo que pode ter diferentes interpretações

Definição Peters (2002): "*Qualidade de software é avaliada em termos de atributos de alto nível chamados fatores, que são medidos em relação a atributos de baixo nível chamados de critérios*".

Definição Sanders (1994): "*Um produto de software apresenta qualidade dependendo do grau de satisfação das necessidades dos clientes sob todos os aspectos do produto*".

Definição Pressman: "*Qualidade de software é a conformidade a requisitos funcionais e de desempenho que foram explicitamente declarados, a padrões de desenvolvimento claramente documentados, e a características implícitas que são esperadas de todo software desenvolvido por profissionais*".

Definição ISO9126 (1994): "*Qualidade é a totalidade de características e critérios de um produto ou serviço que exercem suas habilidades para satisfazer às necessidades declaradas ou envolvidas*" (UNIVASF, acessado em: Agosto 2021)

2.1 Definições de Qualidade

O conceito da qualidade tem hoje importância fundamental para alavancar a competitividade das empresas. Atualmente, a preocupação com a qualidade deixou de ser um diferencial competitivo e passou a ser um pré-requisito básico para participação no mercado. No setor de *software* não é diferente. A disseminação do uso do *software* em todas as áreas, envolvendo monitoração, controle e gestão de funções críticas, tem aumentado consideravelmente a importância da qualidade de *software*.

Podemos definir qualidade sob o aspecto de três pilares, que são:

Os requisitos de software são a base na qual a qualidade é medida. A falta de conformidade com os requisitos significa falta de qualidade.

Padrões especificados definem um conjunto de critérios de desenvolvimento que orientam a maneira segundo a qual o software passa pelo trabalho de engenharia. Se os critérios não forem seguidos, o resultado quase que seguramente será a falta de qualidade.

Existe também um conjunto de requisitos implícitos que frequentemente não são mencionados na especificação. Por exemplo, o desejo de uma boa Integridade no acesso ao Sistema.

2.2 Qualidade

Qualidade hoje em dia não é apenas um diferencial de mercado para a empresa conseguir vender e lucrar mais, é um pré-requisito que a empresa deve conquistar para conseguir

colocar seu produto no mercado global. Apesar da ideia de qualidade parecer aparentemente intuitiva, quando analisada com maior atenção, o conceito se revela um pouco mais complexo.

Conceituar qualidade de fato é uma tarefa complexa, mas ela pode ser vista como um método gerencial que através de procedimentos disseminados por toda a organização, busca garantir um produto final que satisfaça às expectativas do cliente, dentro daquilo que foi acordado inicialmente.

Garantia de qualidade: consiste nas funções gerenciais de auditar e relatar.

Controle de qualidade: envolve a série de inspeções, revisões e testes usados ao longo do processo de *software* para garantir que cada produto de trabalho satisfaça os requisitos estabelecidos para ele.

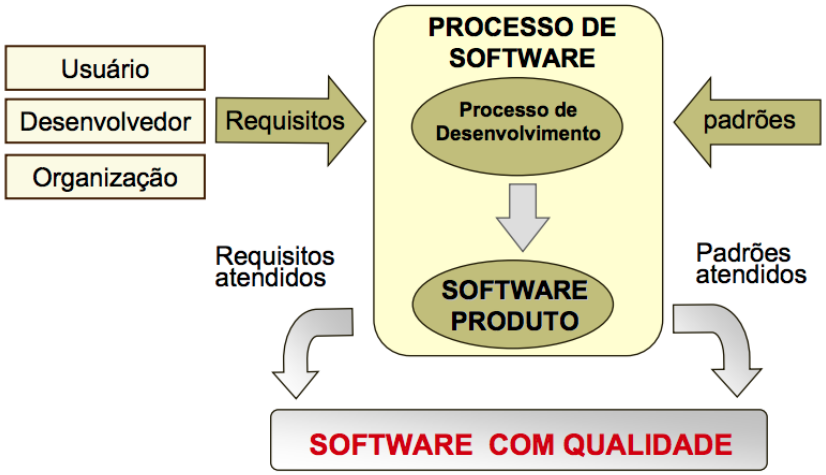

Figura 1 – Ciclo de qualidade

2.3 Modelo de Qualidade CMM - Capability Maturity Model

Conhecido como "Modelo de Maturidade da Capacidade", o CMM é uma iniciativa do SEI (Software Engineering Institute) para avaliar e melhorar a capacitação de empresas que desenvolvem software [EDS Quality Plan, 2004]. O projeto CMM foi apoiado pelo Departamento de Defesa do Governo dos Estados Unidos, que é um grande consumidor de software e precisava de um modelo formal que permitisse selecionar os seus fornecedores de software de forma adequada.

Embora não seja uma norma emitida por uma instituição internacional (como a ISO ou o IEEE), esta norma tem tido uma grande aceitação mundial, até mesmo fora do mercado americano. O modelo, publicado em 1992, pode ser obtido na própria Internet com facilidade. O CMM também é chamado de SW-CMM (Software CMM).

2.4 Maturidade

O CMM é um modelo para medição da maturidade de uma organização no que diz respeito ao processo de desenvolvimento de software. A definição do que é "Maturidade"pode

ser melhor entendida através da análise da figura abaixo.

Organizações Maduras	**Organizações Imaturas**
Papéis e responsabilidades bem definidos.	Processo improvisado.
Existe base histórica.	Não existe base histórica.
É possível julgar a qualidade do produto.	Não há maneira objetiva de julgar a qualidade do produto.
A qualidade dos produtos e processos é monitorada.	Qualidade e funcionalidade do produto sacrificados.
O processo pode ser atualizado.	Não há rigor no processo a ser seguido.
Existe comunicação entre o gerente e seu grupo Resolução de crises imediatas.	

Figura 2 – Organizações Maduras x Organizações Imaturas

2.5 Níveis

O CMM classifica as organizações em cinco níveis distintos, cada um com suas características próprias. No nível 1, estão as organizações mais imaturas, onde não há nenhuma metodologia implementada e tudo ocorre de forma desorganizada. Por outro lado, no nível 5 estão as organizações mais maduras, onde cada detalhe do processo de desenvolvimento está definido, quantificado e acompanhado e a organização consegue até absorver mudanças no processo sem prejudicar o seu desenvolvimento.

2.5.1 Nível 1 – Inicial

O processo de desenvolvimento é desorganizado e caótico. Poucos processos são definidos e o sucesso depende de esforços individuais e heróicos. O processo de software de uma organização de Nível 1 é imprevisível porque é constantemente alterado ou modificado à medida que o trabalho progride. Os cronogramas, os orçamentos, as funcionalidades e a qualidade do produto são geralmente imprevisíveis. O desempenho depende da capacidade dos indivíduos e varia com as suas habilidades, conhecimentos e motivações inatas. Existem poucos processos de software estáveis em evidência e o desempenho só pode ser previsto através da habilidade individual ao invés da capacidade da organização;

2.5.2 Nível 2 - Repetível

Os processos básicos de gerenciamento de projeto estão estabelecidos e permitem acompanhar custo, cronograma e funcionalidade. Os projetos nas organizações de nível 2 têm instalado controles básicos de gestão de software. Os compromissos do projeto são baseados em resultados observados em projetos anteriores e nos requisitos do projeto atual. Os gerentes do projeto acompanham os custos, os cronogramas e as funcionalidades do software. Os problemas

com compromissos são identificados quando surgem. Os requisitos de software e os produtos de trabalho desenvolvidos para satisfazê-los são congelados e a integridade dos mesmos é controlada. Os padrões do projeto de software são definidos e a organização garante que eles sejam seguidos fielmente. O projeto de software trabalha com os seus subcontratados, se existirem, para estabelecer uma forte relação cliente-fornecedor.

2.5.3 Nível 3 - Definido

Tanto as atividades de gerenciamento quanto de engenharia do processo de desenvolvimento de software estão documentadas, padronizadas e integradas em um padrão de desenvolvimento da organização. Todos os projetos utilizam uma versão aprovada e adaptada do processo padrão de desenvolvimento de software da organização. A capacidade do processo de software das organizações de nível 3 pode ser resumida como sendo padrão e consistente porque a engenharia de software e as atividades de gestão são estáveis e repetíveis. Dentro de linhas estabelecidas de produtos, o custo, o cronograma e as funcionalidades estão sob controle e a qualidade de software é acompanhada.

2.5.4 Nível 4 - Gerenciado

São coletadas medidas detalhadas da qualidade do produto e processo de desenvolvimento de software. Tanto o produto quanto o processo de desenvolvimento de software são entendidos e controlados quantitativamente. A capacidade do processo de software das organizações de nível 4 pode ser resumida como sendo previsível porque o processo é medido e opera dentro de limites mensuráveis. O processo desse nível permite que a organização antecipe as tendências na qualidade dos produtos e dos processos dentro das fronteiras quantitativas desses limites. Quando esses limites são excedidos, são tomadas ações para corrigir a situação. Os produtos de software são, como era de se esperar, de alta qualidade;

2.5.5 Nível 5 - Otimizado

O nível de maturidade 5 se concentra no melhoramento contínuo do desempenho de processos através de melhorias tecnológicas incrementais e inovadoras. Os objetivos quantitativos de melhoria de processos para a organização são estabelecidos, continuamente revisados para refletir alterações nos objetivos do negócio. O melhoramento contínuo é conseguido através de um *feedback* dos processos e pelo uso pioneiro de tecnologias inovadoras.

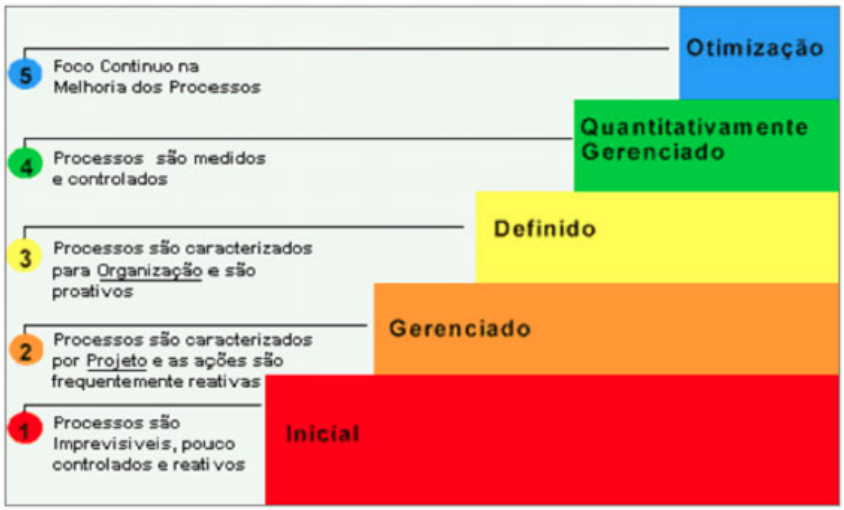

Figura 3 – Níveis de Maturidade [SEI, Software Engineering Institute]

2.6 Entendendo os níveis de maturidade

O modelo CMM descreve atributos essenciais (ou chaves) que seriam esperados para caracterizar uma organização em um nível de maturidade [Maldonado,2001]. É um modelo normativo, no sentido de que as práticas detalhadas caracterizam os tipos normais de comportamento que seriam esperados em uma organização que desenvolve projetos em larga escala em um contexto de contratação governamental. A intenção é que o CMM tenha um nível suficiente de abstração que não restrinja desnecessariamente a maneira que o processo de software é implementado pela organização: ele simplesmente descreve o que normalmente seria esperado que os atributos essenciais do processo de software fossem.

2.7 CMMI

A experiência no uso do CMM durante uma década serviu para identificar pontos em que o modelo poderia ser melhorado. Ao mesmo tempo, o surgimento do projeto SPICE (Software Process Improvement and Capability Determination), da ISO, levou à necessidade de compatibilização do CMM com a futura norma ISO 15504, que será o resultado do projeto. Por estas razões, o SEI iniciou um projeto chamado CMMI (CMM Integration).

O CMMI é um modelo para definir e melhorar a capacidade e a maturidade dos processos das organizações que produzem software. As empresas que investem no CMMI têm como principais benefícios o aumento da qualidade dos processos de software e o reconhecimento internacional dessa qualidade. Também têm uma maior e mais efetiva gerência de projetos de software, no que diz respeito a custo, tempo e recursos utilizados. Esse modelo possibilita uma redução significativa de defeitos nos serviços gerados e maior qualificação do pessoal no atendimento ao cliente, além da customização dos processos de acordo com as necessidades. Com isso, as empresas reduzem o retrabalho, baixam seus custos e agilizam suas soluções, agregando valor para o cliente.

Os objetivos do CMMI são, basicamente, voltados para a redução do custo da implementação de melhoria de processo. São eles [Ahern, 2001]:

- Eliminação de inconsistências e redução de duplicidades;
- Melhoria da clareza e entendimento;
- Utilização de terminologia comum e estilo consistente;
- Estabelecimento de regras de construção uniformes;
- Manutenção de componentes comuns;
- Consistência com a futura norma ISO/IEC 15504;
- Sensibilidade às implicações dos esforços legados.

A busca pela certificação CMMI representa a afirmação da filosofia de aprimoramento contínuo dos produtos e a concretização de um importante passo rumo à exportação de software.

A principal mudança do CMMI em relação ao CMM é a possibilidade de utilização de duas diferentes abordagens para a melhoria de processos. Estas duas abordagens são conhecidas como o "representação contínua" e o "representação em estágios". Existem muitas razões para uma empresa selecionar uma representação ou outra. Pode ser que uma organização escolha utilizar a representação que lhe seja mais familiar.

O projeto CMMI envolveu uma grande quantidade de pessoas de diferentes organizações do mundo todo. Estas organizações utilizavam um modelo CMM ou múltiplos CMMs e estavam interessadas nos benefícios do desenvolvimento de um framework integrado para auxiliar a melhoria de processos no âmbito do empreendimento como um todo.

O trabalho do projeto CMMI é patrocinado pelo Departamento de Defesa dos Estados Unidos (Department of Defense – DoD), especificamente pelo departamento da Sub-Secretaria de Defesa, Aquisição, Tecnologia e Logística (Office of the Under Secretary of Defense, Acquisition, Technology, and Logistics - OUSD/AT&L). O patrocínio da indústria é garantido pelo Comitê de Engenharia de Sistemas da Associação Industrial da Defesa Nacional (National Defense Industrial Association - NDIA).

O produto de software passa a ser , cada vez mais, um componente comum em uma série de outros produtos, desde carros, eletrodomésticos, elevadores, telefones, até sistemas de informação organizacionais. De um produto exige-se qualidade e preço. Portanto, como produto, o software deve ter o nível de qualidade exigido e procurar ser desenvolvido com o menor custo possível.

Produzir software de qualidade é uma meta básica da engenharia de software, que disponibiliza métodos, técnicas e ferramentas para esse fim. Muito tem sido escrito sobre qualidade e seus vários adjetivos. No entanto, o fundamental é que o software seja confiável, ou seja, que seja eficaz e siga os padrões exigidos pelo contexto onde irá atuar. Cada vez mais a sociedade pressiona o setor de software para que a característica "qualidade" seja preponderante. Normas internacionais como a ISO e iniciativas como as do SEI (Software Engineering Institute) são exemplos disso.

Devemos sempre ter em mente que a produção de software é um processo que envolve, como parte fundamental, seres humanos. As tecnologias de produção de software, tem por objetivo automatizar ao máximo a produção, mas ainda são fundamentalmente dependentes da qualidade das equipes, das pessoas, de todo o time envolvido num processo de desenvolvimento.

3 PROCESSOS vs THREAD

Para permitir que o computador faça mais de uma atividade ao mesmo tempo, tanto o processo quanto a *thread* fornecem um ótimo serviço, mas há diferença entre eles na maneira como operam.

3.1 Thread

Uma *thread* é uma linha de execução de código que executa em paralelo com outras linhas do mesmo processo, compartilhando seu espaço de memória. Na prática uma *thread* é equivalente a um "mini-processo" dentro de um processo que permite várias ações sejam executadas em paralelo por um mesmo processo.

Em um programa muitas vezes é necessário executar mais de uma atividade ao mesmo tempo ex.: aguardar a entrada de dados do usuário e reproduzir um som enquanto aguarda.

Uma *thread* é muito mais leve que um processo comum, o ganho de performance na criação e destruição de *threads* se comparada a processos (10 a 100x). Quando uma aplicação tem atividade I/O *bound* e CPU *bound* as *threads* podem acelerar a execução, pois não concorrerão por recurso. O uso de *threads* pode também garantir um uso máximo dos vários processadores existentes em uma CPU.

3.2 Processo

Um processo, em geral, é uma série contínua de ações para alcançar um resultado específico. Mas, no mundo dos computadores, um processo é uma instância de um programa de computador em execução. Em outras palavras, é uma ideia de uma única ocorrência de um programa de computador em execução. Simplesmente os processos são binários em execução que contêm um ou mais threads.

De acordo com o número de threads envolvidos em um processo, existem dois tipos de processos. Eles são processos single-thread e processos multi-thread. Como o próprio nome sugere, um processo de thread único é um processo que possui apenas um segmento. Portanto, este segmento é um processo e há apenas uma atividade acontecendo. Em um processo multi-thread, há mais de um thread e há mais de uma atividade em andamento.

Dois ou mais processos podem se comunicar entre si usando a comunicação entre processos. Mas é bastante difícil e precisa de mais recursos. Ao fazer um novo processo, o programador precisa fazer duas coisas. Eles são a duplicação do processo pai e a alocação de memória e recursos para o novo processo. Então isso é muito caro.

3.3 Qual é a diferença entre Process e Thread?

- Os processos são difíceis de criar porque precisam de uma duplicação do processo pai e da alocação de memória, enquanto os *threads* são fáceis de criar, pois não requerem um espaço de endereço separado.
- *Threads* são utilizadas para tarefas simples, quanto os processos são utilizado para tarefas pesadas, como a execução de um aplicativo.
- Os processos não compartilham o mesmo espaço de endereço, mas os threads dentro do mesmo processo compartilham o mesmo espaço de endereço.
- Os processos são independentes uns dos outros, mas os *threads* são interdependentes, pois compartilham o mesmo espaço de endereço.
- Um processo pode consistir em vários *threads*.
- Como os *threads* compartilham o mesmo espaço de endereço, a memória virtualizada está associada apenas a processos, mas não a *threads*. Mas um processador virtualizado distinto está associado a cada *thread*.
- Cada processo tem seu próprio código e dados, enquanto os *threads* de processos compartilham o mesmo código e dados.
- Cada processo começa com um *thread* principal, mas pode criar *threads* adicionais, se necessário.
- A alternância de contexto entre processos é muito mais lenta do que a alternância de contexto entre *threads* do mesmo processo.
- *Threads* podem ter acesso direto aos seus segmentos de dados, mas os processos têm sua própria cópia dos segmentos de dados.
- Os processos têm *overheads*, mas não threads.

3.3.1 Processo vs. Thread

Processo e *thread* são duas técnicas utilizadas por programadores para controlar o processador e a execução de instruções em um computador de maneira eficiente e eficaz. Um processo pode conter vários *threads*. *Threads* fornecem uma maneira eficiente de compartilhar memória, embora opere várias execuções do que processos. Portanto, os *threads* são uma alternativa para vários processos. Com a tendência crescente de processadores multi-core, os *threads* se tornarão a ferramenta mais importante no mundo dos programadores.

4 SEGURANÇA E AUDITORIA

A sempre crescente demanda por prestação de mais e melhores serviços públicos tornou necessário disponibilizar meios de processamento de dados e de comunicação para troca de informações, bem como para permitir interação entre as organizações públicas e os cidadãos. Essa interação tem sido mediada cada vez mais pela Internet e por meio de serviços de e-gov apoiados em sistemas de informação computadorizados. Consequentemente, a segurança (especialmente a disponibilidade, a integridade, a confidencialidade ou sigilo e a autenticidade) dessas informações passou a ser uma preocupação do poder público e um tema crítico da gestão moderna, seja ela pública ou privada. Uma forma de gerenciar essa criticidade é criar normas, formular políticas, padronizar procedimentos e práticas, estabelecer medições e métricas, além de automatizar controles, de modo a não só aumentar previsibilidade dos resultados dessa prestação de serviço, mas principalmente melhorar a capacidade de lidar com riscos decorrentes das vulnerabilidades dos sistemas – sejam eles manuais ou automatizados - e das ameaças existentes (sobretudo as originadas da Internet).

Essas normas, políticas, procedimentos, práticas, métricas e mecanismos automatizados são controles, e podem ser analisadas através de sua vinculação com os objeto de controle que são decompostos em pontos de controle. A auditoria de segurança de informação é uma atividade devidamente estruturada para examinar criteriosamente a situação desses controles que se aplicam à segurança da informação, especialmente por meio da análise de objetos e seus pontos de controle, vis-a-vis a probabilidade de ameaças às informações críticas sobre as quais atuam esses controles. Isto é necessário porque os controles ou a ausência deles podem se constituir em vulnerabilidades exploráveis ou portas de entrada para produzir incidentes de segurança da informação. A auditoria cria condições técnicas para investigar e emitir um juízo sobre as evidências encontradas, de modo a se antecipar ante a possíveis riscos de violação de um patrimônio precioso: os ativos de informação. Esses ativos correspondem àquelas informações e todos os recursos associados que têm alto valor para o negócio público ou privado. Consideram-se como ativos de informação os processos organizacionais e processuais (procedimentos, roteiros, atividades), itens físicos (instalações, equipamentos, cabeamento) e lógicos (programas, sistemas, estruturas de dados), que devem ser auditados continuamente.

De outra forma, auditoria é a expressão de opinião feita por um profissional devidamente qualificado, acerca de uma determinada situação, e documentada na forma de um relatório ou parecer. Para tal, o auditor emprega práticas geralmente aceitas, baseadas em um método racional, em evidências, em respeito aos princípios éticos, com responsabilidade perante o cliente, com devido cuidado e habilidade profissional, sujeito à revisão por pares, mas com independência no que se refere à roteirização, à investigação e análise e à produção do relato.

Em linhas gerais, a auditoria de segurança da informação pretende assegurar que os ativos de informação, considerados os objetos de auditoria em segurança da informação,

estejam absolutamente sob controle da organização. Para tal, é preciso verificar que os controles estejam de acordo com as normas e políticas de segurança estabelecidas para esses ativos, bem como se o que está em operação alcança os objetivos de segurança definidos. A auditoria de segurança de informação envolve também o provimento de uma avaliação independente dos controles da organização (normas, políticas, padrões, procedimentos, práticas, métricas e mecanismos) empregados para salvaguardar a informação, em formato eletrônico ou não, contra perdas, danos, divulgação não intencional e indisponibilidades. Por fim, a auditoria é uma atividade realizada na forma de ações projetizadas, que tem um início, meio e fim, e que visam produzir resultados dentro de custos, prazos e qualidades esperadas. Para alcançar tais objetos, as auditorias, projetos individuais, agrupam-se em programas, que compreendem a realização de várias auditorias ao longo de um período de tempo de meses ou anos, e que visam melhorar sistematicamente o desempenho, a eficiência e a segurança organizacionais.

4.1 Teste de Software

Teste de software é o processo de execução de um produto para determinar se ele atingiu suas especificações e funcionou corretamente dentro do ambiente para o qual foi projetado. O seu objetivo é buscar falhas em um produto, para que as causas dessas falhas sejam identificadas e possam ser corrigidas pela equipe de desenvolvimento antes da entrega final.

4.1.1 Teste caixa-branca (ou Teste Estrutural)

Teste de caixa branca é a técnica de teste que testa a estrutura interna e a implementação do produto, tornando-as visíveis para o testador. Aqui, a estrutura interna implica design, código, fluxo de dados e bancos de dados. O objetivo do teste de caixa branca é verificar o fluxo de trabalho interno do produto. A eficácia dos testes é medida por meio da cobertura de código. Para definir, a cobertura de código é a porcentagem de linhas de código exercidas por meio dos testes do total de linhas de código. Os outros nomes comuns para o teste de caixa branca são caixa transparente, caixa de vidro, caixa transparente ou teste estrutural.

A realização de testes de caixa branca requer programação, banco de dados e conhecimento de design do produto. Além disso, os testadores também exigem o conhecimento de ferramentas como ferramentas de cobertura e depuradores para testes de caixa branca.

O exemplo típico de teste de caixa branca é o teste de componentes do produto. Por definição, Teste de componentes significa testar as unidades de componentes individuais do produto, que podem ser um único programa ou grupo de programas formando um módulo. Assim, esses testes se concentram na avaliação daquele módulo específico. Pode-se escrever testes *JUnit* ou *Cucumber* para este tipo de teste de componentes. Normalmente, esses testes invocam métodos ou funções que fornecem os dados de entrada e avaliam sua saída comparando-os com os resultados esperados.

4.1.2 Teste caixa-preta (ou Teste Funcional)

Técnica de teste em que o componente de software a ser testado é abordado como se fosse uma caixa-preta, ou seja, não se considera o comportamento interno do mesmo. Dados de entrada são fornecidos, o teste é executado e o resultado obtido é comparado a um resultado esperado previamente conhecido. Haverá sucesso no teste se o resultado obtido for igual ao resultado esperado. O componente de software a ser testado pode ser um método, uma função interna, um programa, um componente, um conjunto de programas e/ou componentes ou mesmo uma funcionalidade.

4.1.3 Teste caixa-cinza

O teste de caixa cinza é um tipo de teste profissional frequentemente usado para software de computador, que combina certos aspectos do teste de caixa preta e teste de caixa branca. A idéia geral é combinar esses dois outros tipos para utilizar os pontos fortes de cada um, minimizando suas limitações ou fraquezas. O teste da caixa cinza consiste basicamente em testes profissionais, nos quais os testadores compreendem algumas das maneiras pelas quais o software funciona, mas eles não entendem tudo sobre ele.

Ao desenvolver e testar software de computador, existem dois modelos comuns de teste frequentemente utilizados. São testes de caixa preta e de caixa branca, e o teste de caixa cinza é basicamente uma combinação de ambos. O teste da caixa preta consiste em testes nos quais os testadores não entendem ou têm acesso ao código que executa o software. Por exemplo, alguém pode utilizar o teste de caixa preta para permitir que uma empresa externa desenvolva software para executar com um sistema operacional (SO) sem fornecer à empresa o código fonte do SO.

Esse tipo de teste é frequentemente usado por muitas empresas de *software* e pode ser usado para testes internos e externos. Uma das maiores fraquezas desse tipo de teste, no entanto, é que o conhecimento limitado dos testadores pode potencialmente dificultar seus testes. O teste de caixa cinza procura aliviar alguns desses problemas combinando esse tipo de teste com certos elementos do teste de caixa branca.

O teste de caixa branca consiste em testes de software realizados por pessoas que compreendem completamente o software que está sendo testado e têm acesso ao código-fonte do software. Isso geralmente é feito internamente em um desenvolvedor de software para garantir que o programa funcione corretamente e para permitir que os testadores interajam diretamente com o código por trás do programa. Porém, existem problemas de segurança em potencial com esse tipo de teste e, portanto, o teste da caixa cinza é frequentemente usado para combinar os dois tipos de maneira produtiva e segura.

No teste da caixa cinza, os testadores compreendem certos aspectos do software que está sendo usado e podem ver algumas partes do código-fonte, mas não todo. Isso permite que os testadores interajam e compreendam mais completamente o programa que estão testando

do que o teste de caixa preta permite, mas sem os problemas completos de acesso e segurança que podem surgir dos testes de caixa branca. Alguém que esteja realizando testes de caixa cinza no software de um novo sistema operacional, por exemplo, poderá ver o código de aspectos do sistema operacional relevantes para o teste do programa, mas não todo o código-fonte.

Figura 4 – Teste Caixa Cinza

5 CONCLUSÃO

Através do estudo durante o curso e das pesquisas realizadas para a elaboração do trabalho de Projeto Integrado II, houve um amadurecimento maior dos vários conceitos que serão empregados na carreira profissional, visto que a inclusão crescente da tecnologia dentro das estratégias de negócio está levando todas as empresas a passarem por uma transformação digital — seja por enxergar as oportunidades que ela traz ou simplesmente por serem forçadas para se manterem competitivas.

Nesse cenário, o profissional de TI se torna uma figura central do planejamento e execução de novos processos, produtos e entrega ao público. O seu conhecimento e a sua visão da tecnologia como veículo prático de inovação são fundamentais para todos os departamentos dentro do escritório.

Quanto mais as tecnologias evoluem, mais rapidamente esse processo acontece. Por isso, para estar sempre à frente das possibilidades que o mercado apresenta e continuar como uma referência, o responsável pela TI de um negócio precisa de uma atualização constante, se possível diária.

Manter-se atualizado não significa apenas ter noção do que está acontecendo no mundo e no seu setor, mas qual é a aplicação prática de novidades e tendências. Se você quer assumir esse papel de liderança e sonha em ver sua empresa se consolidando pela transformação digital, é a sua motivação em buscar o novo, o diferenciado que pavimentará esse caminho.

Referências

BRAGA, P. R. **Qualidade de Software: Visão Geral**. acessado em: Agosto 2021. <https://edisciplinas.usp.br/pluginfile.php/57546/mod_resource/content/1/Aula8-QualidadeSoftware.pdf>. Nenhuma citação no texto.

DEVMEDIA. **Qualidade de Software - Engenharia de Software 29**. acessado em: Agosto 2021. <https://www.devmedia.com.br/qualidade-de-software-engenharia-de-software-29/18209>. Nenhuma citação no texto.

IFRN. **Sistemas Operacionais - Threads e Processos**. acessado em: Agosto 2021. <https://docente.ifrn.edu.br/tadeuferreira/disciplinas/2016.1/sistemas-operacionais/Aula05.pdf>. Nenhuma citação no texto.

RODRIGUES, R. W. da S. **Auditoria e Conformidade de Segurança da Informação**. acessado em: Agosto 2021. <https://www.trf3.jus.br/documentos/rget/seguranca/CLRI/GSIC345_Auditoria_Conformidade_Seguranca_Informacao.pdf>. Nenhuma citação no texto.

SALGUEIRO, L. F. **Modelos de Qualidade de Software**. acessado em: Agosto 2021. <https://bsi.uniriotec.br/wp-content/uploads/sites/31/2020/05/200508SalgueiroMarcus.pdf>. Nenhuma citação no texto.

TI, C. **Teste de Software – Teste caixa-branca e caixa-preta**. acessado em: Agosto 2021. <https://www.canalti.com.br/teste-de-software/teste-de-software-teste-caixa-branca-e-caixa-preta/>. Nenhuma citação no texto.

TUTORIALCUP. **Teste de caixa branca**. acessado em: Agosto 2021. <https://www.tutorialcup.com/pt/ensaio/tipos-de-teste/white-box-testing.htm>. Nenhuma citação no texto.

UFPR. **Teste Estrutural e Funcinal**. acessado em: Agosto 2021. <https://www.inf.ufpr.br/lmperes/testeEstruturalFuncional.pdf>. Nenhuma citação no texto.

UNIVASF. **Qualidade de Software: Visão Geral**. acessado em: Agosto 2021. <http://www.univasf.edu.br/~ricardo.aramos/disciplinas/ESI2009_2/Aula05.pdf>. Citado na página 2.