UNIVERSIDADE SÃO JUDAS TADEU

-

CAMPUS BUTANTÃ

GESTÃO E QUALIDADE DE SOFTWARE – CCP1AN-BUE1

Nome do Grupo

CONCEITOS E ESTRATÉGIAS DE TESTES DE SOFTWARES

Membro(s) do Grupo

822160071 – FABRÍCIO PERES – CCP - 822160071@ulife.com.br

824116869 – HERMANO PEREIRA DE SOUSA – CCP - 824116869@ulife.com.br

822127136 – JONATA PABLO GARCIA – CCP - 822127136@ulife.com.br

823214064 – JÚLIA SILVA PEREIRA – ADS - 823214064@ulife.com.br

823126459 – RANGEL RIBEIRO SANTOS – ADS - 823126459@ulife.com.br

8222241099 – VÍTOR DE SOUZA – CCP - 8222240199@ulife.com.br

SUMÁRIO

1 – CONCEITO DE TESTE

Significado de teste é: Prova que verifica a verdade em relação a algo ou alguém; exame, verificação...

Existem diversos tipos de Teste que podem ser realizados para Softwares, como por exemplo:

- Segurança
- Funcional
- Integridade
- Desempenho
- Stress

Todos esses testes têm como funcionalidade, verificar se o sistema produzido dentro do software entrega exatamente aquilo que foi programado e a procura de problemas que podem ocasionar nesses sistemas, como uma falha na busca de uma informação no banco de dados. Assim os testes servem como uma forma de garantir que o sistema produzido execute exatamente para aquilo que foi criado

Os testes representam uma etapa de extrema importância no processo de desenvolvimento de software, pois visam validar se a aplicação está funcionando corretamente e se atende aos requisitos especificados.

Nesse contexto existem diversas técnicas que podem ser aplicadas em diferentes momentos e de diferentes formas para validar os aspectos principais do software.

Para evitar que o teste seja uma mera etapa do ciclo de desenvolvimento, a implantação de um processo relacionado a este garante um maior controle das atividades de teste e, consequentemente, mais qualidade ao software. Dessa forma, podemos dizer que os testes servem para:

- Identificar defeitos e bugs no software,
- Verificar se o software atende aos requisitos do cliente,
- Evitar interrupções ou erros graves que possam impactar a experiência do usuário e a reputação da empresa Auxiliando na documentação.
- Melhorar o software
- Aumentar a satisfação do usuário
- Aumentar a eficiência e a confiabilidade do software
- Reduzir os custos de manutenção corretiva e retrabalho

2 – ESTRATÉGIAS DE TESTES

2.1 – A importância das estratégias de testes de software

As estratégias de teste de software desempenham um papel crucial no desenvolvimento de qualquer aplicativo ou sistema.

São projetadas para garantir a qualidade e a confiabilidade do software antes de seu lançamento no mercado.

A falta de uma estratégia bem definida, podem tornar os testes ineficientes e inadequados, resultando em falhas e experiências negativas para os usuários.

Razões pelas quais as estratégias de teste de software são importantes:

- Ajudam a identificar e corrigir quaisquer problemas ou bugs antes do lançamento.

- Os testes permitem que os desenvolvedores e engenheiros de software verifiquem se o software está funcionando de acordo com as especificações e requisitos estabelecidos.

- Auxiliam no aumento da confiabilidade do software. Ao testar diferentes cenários e condições, é possível identificar vulnerabilidades e inconsistências, garantindo que o software seja robusto e seguro para os usuários.

- Consideram o impacto negativo que falhas e problemas podem ter nos usuários e no negócio como um todo.

Para garantir a eficácia das estratégias de teste de software, é importante considerar diferentes abordagens e táticas. Cada projeto ou sistema requer uma estratégia específica, baseada em suas características e requisitos.

2.2 – Princípios e metodologias para elaborar estratégias de testes de software.

Para elaborar as estratégias de teste de software, deve seguir alguns princípios e metodologias que podem guiar o processo e garantir sua eficácia. Esses princípios são fundamentais para o desenvolvimento de estratégias sólidas e abrangentes.

Eis alguns desses princípios e metodologias:

- Compreender os requisitos -> antes de iniciar a elaboração de estratégias, é necessário compreendermos todos os requisitos. Envolvendo uma análise detalhada das funcionalidades, expectativas do cliente e casos de uso. Com todo esse

entendimento, será possível direcionar os esforços de testes para as áreas mais críticas e importantes do software.

- Definir objetivos claros -> as estratégias de testes, devem ter objetivos claros que poderão incluir a detecção de bugs, avaliação do desempenho, validação de requisitos, entre outros. Com os objetivos estabelecidos, é possível direcionar os esforços de testes de forma mais eficiente e criar testes adequados para alcançar esses objetivos.

- Utilizar técnicas e abordagens adequadas -> existem diversas técnicas de abordagens disponíveis. É importante definir as mais apropriadas para cada projeto. A escolha das técnicas corretas, pode influenciar na eficácia dos testes e a detecção de problemas no software.

- Testar em diferentes cenários -> Testar o software em diferentes cenários e ambientes reproduzindo situações reais de uso. Incluindo testes em diferentes plataformas, sistemas operacionais, navegadores e dispositivos. Isso ajudará a identificar possíveis problemas de compatibilidade e garante a qualidade do software em diferentes contextos.

- Priorizar os testes de acordo com os riscos -> alguns módulos e funcionalidades, podem representar um risco maior caso apresentem falhas. É importante priorizar os esforços de teste, dando ênfase aos componentes mais críticos e com maior potencial de impacto negativo. Isso permitirá que os recursos sejam alocados de maneira mais eficiente, focando nas áreas mais importantes.

2.3 – Algumas estratégias de testes

- Estratégias de Teste de Software para arquitetura de Software Convencionais.

O quê	Quem	Como
Teste de Unidade – Código	Desenvolvedores	Focaliza cada componente individualmente, garantindo que ele funcione adequadamente como uma unidade. Usa técnicas de testes que exercitam caminhos específicos na estrutura de controle de um componente, para garantir completa cobertura e máxima detecção de erros. Teste Caixa-Branca.
Teste de Integração – Projeto e Arquitetura de Software	Desenvolvedores	Os componentes devem ser montados ou integrados para formar o pacote de software completo. Cuida dos tópicos associados com os problemas duais de verificação e construção de programas. As técnicas de projeto de casos de teste que enfocam as entradas e saídas são mais prevalentes durante a integração.
Validação – Sw construído x Análise de Requisitos (alto nível)	Analista de Teste	Fornecem garantia final de que o software satisfaz a todos os requisitos funcionais, comportamentais e de desempenho.
Sistema	Analista de Teste	Verifica se todos os elementos (hardware, pessoal, banco de dados) combinam adequadamente e se a função/desempenho global do sistema é alcançada.

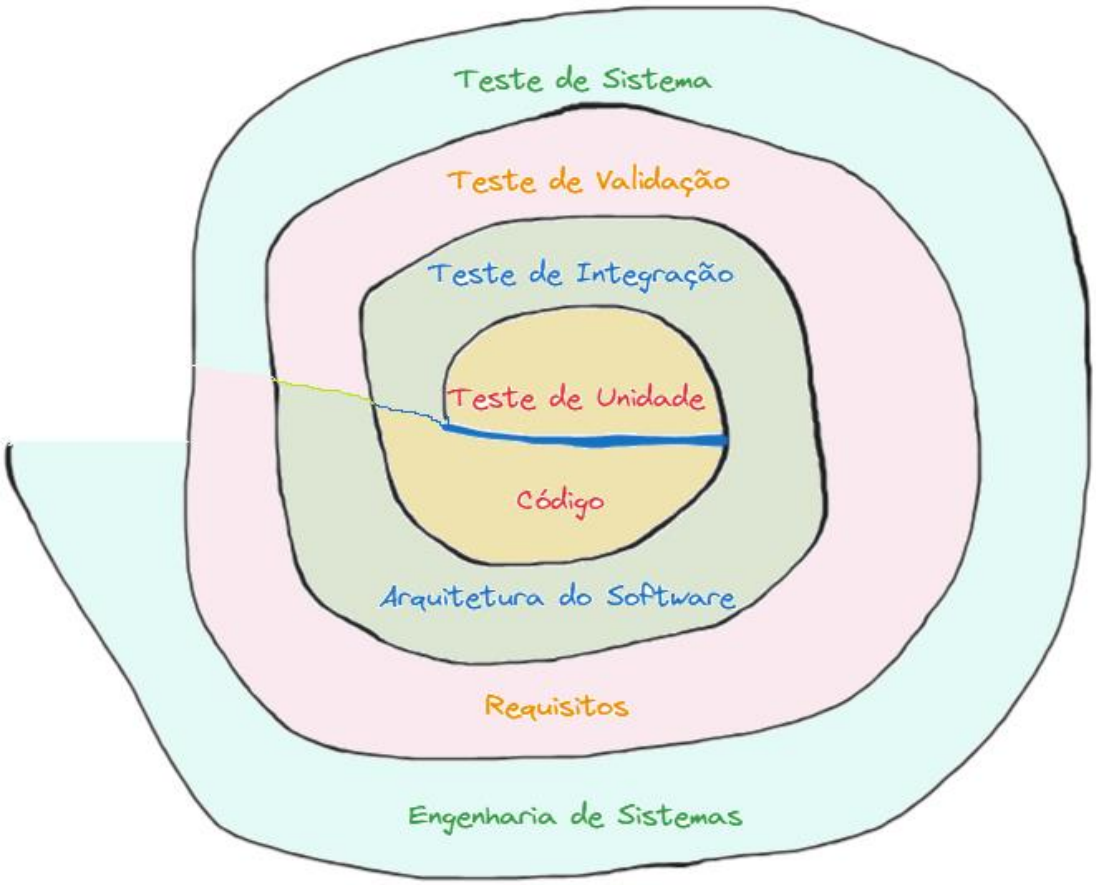
- Estratégias de Teste de Software para arquitetura de Software Orientados a Objetos (OO).

Idêntica a aplicada para arquitetura de software convencional, mas diferente na abordagem.

À medida que as classes são integradas em uma arquitetura OO, uma série de testes de regressão é feita para descobrir erros devidos a comunicação e colaboração entre classes (componentes) e efeitos colaterais causados pela adição de novas classes. Finalmente o sistema é testado como um todo para garantir que erros nos requisitos sejam descobertos.

- Estratégia da Espiral

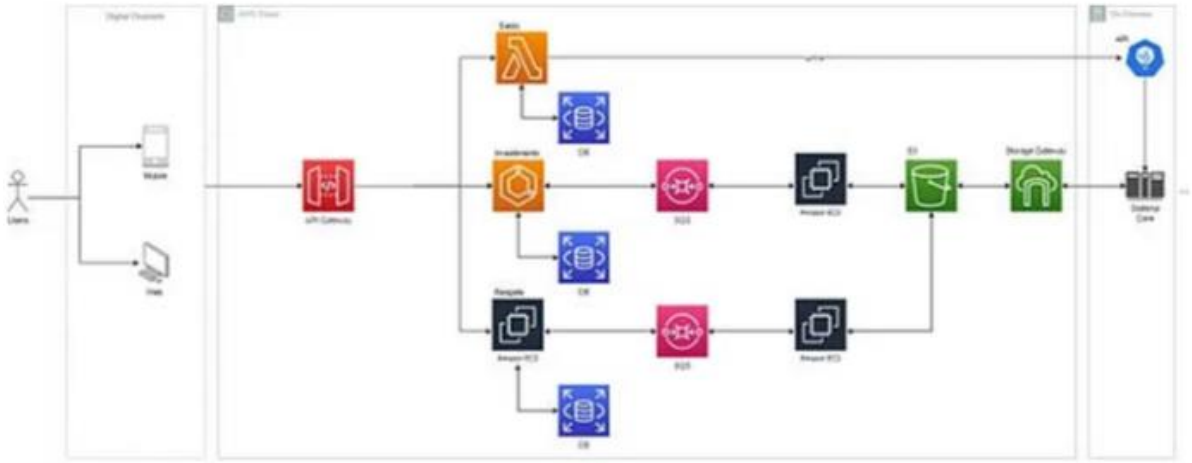
Esta estratégia visualiza o processo de teste de software através de uma espiral. Cada fase do teste é representada por uma volta da espiral, que indica o aumento do escopo do teste.



Começando no centro da espiral, o teste de unidade focaliza em cada unidade do software, como componentes ou classes. Em seguida, movendo-se para fora da espiral, o teste de integração concentra-se na

arquitetura do software. Continuando na mesma direção, o teste de validação valida os requisitos em relação ao software. Por fim, o teste do sistema avalia o software como um todo. Cada fase é representada por uma volta da espiral, indicando o escopo do teste em expansão.

- Estratégia de testes de performance para aplicações em sistemas distribuídos.



Arquitetura do sistema de investimentos fictício

Os atributos de desempenho dos sistemas avaliados durante uma jornada de testes de performance estão listados abaixo:

- Tempo de resposta -> o tempo gasto pelas requisições realizadas para o sistema;
- RPS (Requisições por segundos) -> a quantidade de requisições simultâneas atendida pelo sistema por segundo;
- Utilização de recursos computacionais -> a métrica utilizada para acompanhamento na monitoração do sistema em teste durante ou após a execução (como consumo de CPU e memória, por exemplo);
- Resiliência -> a habilidade do sistema em se recuperar rapidamente e continuar operando mesmo quando uma falha ocorre;

- Capacidade -> o poder de execução das tarefas do sistema sob altas cargas de trabalho;
- Integridade -> a confiabilidade e consistência das informações ao longo do seu ciclo de vida útil.

Os tipos de testes de performance abaixo podem ser explorados para avaliar a performance de um sistema, de acordo com o contexto e objetivos de um teste:

- Testes de carga -> capacidade do sistema em lidar com níveis crescentes de cargas reais, de forma antecipada.
- Testes de estresse -> capacidade do sistema ou componente de lidar com picos de cargas acima dos limites das cargas de trabalho previstas ou especificadas.
- Testes de escalabilidade -> capacidade do sistema em atender a requisitos futuros de eficiência, que podem estar além dos requisitos exigidos atualmente.
- Testes de pico -> capacidade do sistema para responder corretamente a rajadas súbitas de cargas de pico e retornar depois a um estado estável;
- Testes de resistência -> avaliam a estabilidade do sistema ao longo de um período específico para o seu contexto operacional.
- Testes de concorrência -> analisam o impacto das situações em que ações específicas ocorrem simultaneamente, como quando muitos usuários fazem login ao mesmo tempo;

Testes de capacidade -> determinam quantos usuários ou transações o sistema suportará atendendo aos objetivos declarados de performance.

3 – CONCEITOS DE VERIFICAÇÃO E VALIDAÇÃO

Verificação e Validação de software (abrevia-se como V & V) são duas técnicas diferentes, embora muitas vezes sejam confundidos. Barry Boehm, um pioneiro da engenharia de software, expressou sucintamente a diferença entre eles (Boehm, 1979):

Verificação: “Estamos construindo o software da maneira correta?”

A verificação se concentra em garantir que o software está sendo construído de acordo com as especificações. É como seguir um manual de instruções à risca: cada passo é cuidadosamente verificado para assegurar que está sendo realizado corretamente.

- Objetivo: Assegurar que o software está sendo implementado de acordo com o design e os requisitos estabelecidos.
- Atividades:
    - Inspeções de código: Análise manual do código-fonte para identificar erros, inconsistências e não conformidades com as normas de codificação.
    - Revisões de documentos: Análise de documentos como requisitos, design e planos de teste para verificar sua completude e consistência.
    - Walkthroughs: Reuniões onde o desenvolvedor explica o código para outros membros da equipe, buscando feedback e identificando possíveis problemas.

Validação: “Estamos Construindo o Software Certo?”

A validação se concentra em garantir que o software atenda às necessidades dos usuários e aos requisitos do negócio. É como comparar um produto final com um protótipo ou com as expectativas do cliente: ele deve funcionar como esperado e entregar o valor desejado.

- Objetivo: Assegurar que o software atende às necessidades dos usuários e aos objetivos do negócio.
- Atividades:
    - Testes: Execução de casos de teste para verificar se o software funciona conforme o esperado e se atende aos requisitos.
    - Demonstrações: Apresentação do software para os usuários para coletar feedback e garantir que ele atenda às suas expectativas.
    - Inspeções de usabilidade: Avaliação da facilidade de uso do software por usuários finais.

Assim, unindo esses dois grandes conceitos, podemos avaliar se o Software está sendo desenvolvido da maneira correta. Atendendo às demandas do qual foi exigido e se as suas aplicações funcionam com exatidão, sem apresentar problemas ou bugs.

4 – TESTE DE SOFTWARE; TESTE UNITÁRIO; TESTE DE INTEGRAÇÃO

TESTE DE SOFTWARE

O teste de software é uma parte fundamental no desenvolvimento de um software. Ele irá, como o nome sugere, testar e verificar se o software consegue entregar corretamente tudo que ele propõe.

Objetivo do teste de software: O principal objetivo é detectar falhas e problemas no código antes que o software seja entregue ao cliente ou entre em produção. Os testes também servem para validar se o software funciona como esperado em diferentes cenários.

TESTE UNITÁRIO

Esses testes são feitos em um nível muito baixo (próximo ao código fonte) do projeto, por isso, geralmente quem os realiza são os próprios programadores envolvidos no projeto.

Geralmente são realizados de forma isolada do restante do sistema, visto que tem por objetivo assegurar a qualidade das unidades de forma individual e não o sistema como um todo. Podemos entender como “unidade” as menores partes do nosso sistema, ou seja, métodos e funções das classes ou pacotes utilizados no projeto.

Esses testes têm como objetivo verificar as menores unidades isoladamente, garantindo que a lógica de cada uma delas está correta e que funciona conforme o esperado. Geralmente têm um baixo custo para automatização e podem ser executados rapidamente, inclusive por um servidor de integração contínua.

A vantagem dos testes unitários serem implementados é basicamente uma confiança maior no código, devido que ao validar pequenos trechos do código isso dá maior confiança que suas alterações não introduziram novos bugs. Além disso, outra vantagem seria que os testes funcionam como uma espécie de “documentação viva” já que eles mostram o que o código deve fazer.

TESTE DE INTEGRAÇÃO

O teste de integração é uma etapa do processo de desenvolvimento de software em que módulos e componentes são testados em grupo, e não de forma unitária.

O objetivo principal desse tipo de teste é identificar problemas que podem ocorrer quando diferentes partes do software trabalham em conjunto. Esse tipo de teste é muito importante na garantia de um software eficiente e seguro no que se trata da comunicação entre os sistemas.

Existem várias ferramentas que podem ajudar a automatizar o teste de integração, como Jenkins, Selenium, e JUnit para Java. Essas ferramentas permitem que os testes sejam executados automaticamente sempre que há uma atualização no código, facilitando a detecção precoce de problema.

Os testes de integração podem detectar diferentes problemas, como:

- Erros de programação
- Problemas de hardware
- Dados corrompidos -> Erros de dados ocorrem quando os dados são corrompidos ou perdidos durante a transferência entre módulos.
- Erros de sintaxe (erro de digitação) ou semântica nos dados -> São erros que ocorrem quando os dados não estão no formato esperado pelo módulo receptor.
- Diferenças nos resultados esperados -> São diferenças entre os resultados reais e os resultados esperados de um teste de integração.
- Comportamento inesperado dos módulos em combinação -> Comportamento inesperado ocorre quando os módulos exibem um comportamento diferente do esperado.

Os casos de teste podem verificar se os módulos:

- Comunicam-se corretamente entre si;

- Transferem dados corretamente entre si;

- Produzem os resultados esperados.

Como são feitos os testes de integração?

- Identificação dos módulos
- Definição de cenário de teste
- Criação de casos de teste
- Execução dos testes
- Verificação dos resultados
- Documentação dos resultados

Tipos de testes de integração:

- Teste de Integração de Componentes -> Foca na comunicação e interação entre componentes individuais dentro do mesmo sistema.
- Teste de integração vertical -> verificação entre camadas de um sistema.
- Teste de integração horizontal -> verificação da integração entre diferentes módulos ou componentes que atuam no mesmo nível de abstração.
- Teste de integração de sistema -> identifica a integração entre diferentes sistemas ou subsistemas de um software, sistemas externos, como bancos de dados ou serviços web.
- Teste de integração de interface -> Testa a integração através das interfaces de usuário e pontos de entrada do sistema.
- Teste de integração de serviços -> validar a integração entre diferentes serviços que compõem um software.
- Teste de integração de regressão -> validar se as alterações feitas no software não afetaram a integridade do sistema.

Para concluirmos esse tópico, há o teste de integração automatizado que usa ferramentas de automação para executar testes de integração.

Tendo por vantagens: ser mais rápido, mais preciso, mais repetível e mais escalável, além de ser fácil de executar, reduzir custos e melhorar a qualidade.

5 – TESTE DE VALIDAÇÃO, TESTE DE SISTEMA E DEPURAÇÃO

TESTE DE VALIDAÇÃO

Também conhecido como Teste de Aceitação, o teste de validação é fundamental no processo de desenvolvimento de um software, esse tipo de teste assegura que o sistema atenda às necessidades e expectativas do seu usuário final. A importância do teste de validação é garantir a satisfação do cliente e diminuir riscos, e custos.

Possuem várias ferramentas que podem auxiliar no processo de validação, como ferramentas de gestão de testes (por exemplo, Jira, TestRail) e plataformas de automação de testes que suportam a validação de funcionalidades.

O Teste de Validação só começa quando termina o Teste de Integração.

Algumas técnicas mais comuns de teste de validação.

- Testes Alfas -> Este teste ocorre em ambiente controlado.
  - Conduzido pelo desenvolvedor no seu próprio ambiente.
  - Participação de grupo de usuários finais.
  - Software utilizado em condições naturais.

- Desenvolvedor observa, registra erros e problemas de uso.

- Testes Betas -> Este teste ocorre no ambiente do usuário final.
  - Desenvolvedor não está presente.
  - Não é controlado pelo desenvolvedor.
  - Teste do software pelo usuário em condições reais, registrando os problemas e reportando ao desenvolvedor.
  - Com base nos problemas relatados, são feitos os ajustes e lançados para todos os clientes.

Diferentes níveis de validação:

- Teste de Aceitação do Usuário (UAT - User Acceptance Testing): Este é o tipo mais comum de teste de validação, onde os usuários finais testam o sistema em um ambiente que simula o ambiente de produção. O objetivo é garantir que o sistema é utilizável e atende às suas expectativas.
- Teste de Validação de Sistema: Avalia o sistema como um todo para garantir que todos os requisitos e objetivos do negócio sejam atendidos.
- Teste de Validação de Requisitos: Concentra-se em garantir que todos os requisitos especificados foram implementados corretamente no software.

TESTE DE SISTEMA

São criados durante a fase de projeto do sistema.

O teste de sistema é um tipo de teste de software que realiza uma verificação no sistema como um todo. Basicamente ele implica a integração de todos os módulos de componentes individuais e testa se funcionam como um conjunto.

Esse teste serve para validar os requisitos funcionais e não funcionais do sistema, o teste do sistema é uma categoria de teste da caixa negra, o que significa que apenas testa características de funcionamento externas do software, em oposição a testar o design interno da aplicação.

Avalia se o sistema atende aos pré-requisitos estabelecidos na concepção do projeto;

Outros tipos de testes que são inclusos nos testes de sistemas.

- Teste de Recuperação
- Teste de Segurança
- Teste por Esforço
- Teste de Desempenho
- Teste de Disponibilização

Ocasiões em que são efetuados Testes de Sistemas:

- Durante o desenvolvimento de novas versões de software.
- Durante o lançamento da aplicação, quando se realizam os testes alfa e beta.
- Após a unidade e os testes de integração estarem concluídos.
- Quando os requisitos da construção do sistema estiverem completos.
- Quando outras condições de teste são cumpridas.

Envolvidos nos testes:

- Testadores e equipe da Garantia da Qualidade
- Nunca por programadores (os testadores que realizam testes de sistemas não requerem qualquer conhecimento da programação e estrutura do código do software para avaliar completamente um software construído durante os testes do sistema. Em vez disso, os

testadores estão simplesmente a avaliar o desempenho do software a partir da perspectiva de um utilizador).

Alguns dos aspectos de software que os testes de sistema verificam:

- Funcionalidade -> verifica se os diferentes aspectos do sistema completado funcionam como deveriam.
- Integração -> testam como diferentes componentes de software funcionam em conjunto e se se integram sem problemas uns com os outros.
- Produção esperada -> verificar a saída do software durante a utilização regular.
- Erros e falhas -> avaliar a funcionalidade e fiabilidade do software em múltiplas plataformas e sistemas operativos.

Processo de teste do sistema:

- Criar um plano de teste do sistema
- Gerar cenários de teste e casos de teste
- Criar os dados de teste necessários
- Criar o ambiente de teste
- Executar os casos de teste
- Preparar relatórios de bugs
- Re-teste após reparação de bugs
- Repetir o ciclo

DEPURAÇÃO

Depuração, ou debbuging, corresponde ao processo de localizar erros ou bugs no código-fonte de um software.

O processo de depuração normalmente segue algumas etapas dentre elas, identificar o erro, análise do erro, correção e validação.

Normalmente os erros que requerem depuração são os erros associados a erros de sintaxe, erros semânticos, erros lógicos e erros de tempo de execução.

Algumas estratégias para depurar um código podem ser:

Rastreamento inverso: Tem em vista pegar o caminho do código de trás para frente com base o local onde um erro fatal aconteceu, para assim achar o ponto exato de falha, porém isso se torna mais difícil com o tamanho do código.

Depuração Remota: Esse método utiliza um ambiente separado da máquina local onde podemos usar ferramentas de depuração para ir atras do bug.

Registro em log: Grande parte dos programas registram em logs informações importantes, desde tempo de execução até estado do sistema operacional e com isso o desenvolvedor estuda esses logs para achar o motivo do erro.

Com isso podemos notar que depuração é um assunto muito importante tendo em vista que bugs e erros aparecem a todo momento e com isso devemos aprender a resolvê-los e essas estratégias ajudam a corrigir os problemas mais rapidamente, melhorando assim a produtividade e garantindo a qualidade do software.

6 – CONCLUSÃO

Ao aplicar as melhores práticas de estratégias e táticas de teste de software, é possível garantir um produto estável, confiável e com um alto nível de qualidade. Essas práticas contribuem para evitar problemas

futuros, melhorar a experiência do usuário e aumentar a reputação da empresa no mercado.

7 – FONTES DE PESQUISAS

DICIO

https://www.dicio.com.br/teste/

TRT9

https://www.trt9.jus.br/pds/pdstrt9/guidances/concepts/types_of_test_CAE80710.html

AWARI

https://awari.com.br/estrategias-de-teste-de-software-abordagens-e-taticas-em-testes-de-software/

TESTWARE QUALITY

https://testwarequality.blogspot.com/p/estrategias-de-testes.html

ESTRATÉGIA

https://www.estrategiaconcursos.com.br/blog/estrategias-testes-software-caixa-ti/

MEDIUM

https://medium.com/itautech/construindo-uma-estrat%C3%A9gia-de-testes-de-performance-para-aplica%C3%A7%C3%B5es-em-sistemas-distribu%C3%ADdos-fce828f957a7

https://medium.com/@danielemsilva/voc%C3%AA-sabe-o-que-%C3%A9-verifica%C3%A7%C3%A3o-e-valida%C3%A7%C3%A3o-de-software-38124de6a141

WAVINGTEST

https://www.wavingtest.com/post/saiba-quais-as-diferencas-entre-teste-validacao-e-verificacao-de-software#:~:text=Unir%20os%20conceitos%20de%20Validação,sem%20apresentar%20problemas%20ou%20bugs.

DEVMEDIA

https://www.devmedia.com.br/introducao-aos-diferentes-tipos-de-teste/29799

devmedia.com.br, objective.com.br, logicalminds.com

VERICODE BLOG

https://blog.vericode.com.br/testes-de-software/

OBJECTIVE

https://www.objective.com.br/insights/testes-de-software/

https://www.objective.com.br/insights/teste-de-integracao/

CWI

https://cwi.com.br/blog/o-que-e-teste-de-software-por-que-e-necessario/#:~:text=BUG%2C%20DEFEITO%20e%20FALHA,outro%20produto%20de%20trabalho%20relacionado.

HARBOR

https://www.harbor.com.br/harbor-blog/2018/03/15/validacao-de-software/

VISURE

https://visuresolutions.com/pt/guia-de-rastreabilidade-de-gerenciamento-de-requisitos/verifica%C3%A7%C3%A3o-e-valida%C3%A7%C3%A3o-de-requisitos/#:~:text=implanta%C3%A7%C3%A3o%20de%20software.-,O%20que%20%C3%A9%20Valida%C3%A7%C3%A3o%20de%20Requisitos?,%C3%A0s%20regras%20e%20padr%C3%B5es%20ideais.

LUCIDCHART

https://www.lucidchart.com/blog/pt/v-model-verificacao-e-validacao

ALURA

https://www.alura.com.br/artigos/tipos-de-testes-principais-por-que-utiliza-los?utm_term=&utm_campaign=%5BSearch%5D+%5BPerformance%5D+-+Dynamic+Search+Ads+-+Artigos+e+Conteúdos&utm_source=adwords&utm_medium=ppc&hsa_acc=796

TESTE E VALIDAÇÃO **DE SOFTWARE**

Sandra C. Pinto Ferraz Fabbri - sandraf@dc.ufscar.br