PROVA COMO ELEMENTO DE GARANTIA DA QUALIDADE DE SOFTWARE
TEST AS AN ELEMENT OF SOFTWARE QUALITY ASSURANCE

Nzuzi Rodolfo Henriques Manuel

Licenciado em Engenharia informática;

Instituição: Universidade de Luanda-Instituto de Tecnologia de Informação e Comunicação- INSTIC

Endereço: Bairro dos CTT's km 7, Distrito Urbano de Rangel – Luanda; Contactos: 222041728;E-mail: instic2020@gmail.com ;

E-mail: nzuzirodolfo9@gmail.com;

Codigo ORCID: 0009-0007-1463-3369

LUANDA – ANGOLA

RESUMO

Este trabalho apresenta um estudo de revisão bibliográfica sobre elementos que garantem a qualidade de software, de tal modo a dar respostas as questões como: o que é qualidade de software? porque é importante? como se garante qualidade de software ? e descrever de forma suscinta esses elementos como verificação e validação, estratégia de prova, tipos de provas, níveis de prova, métodos de provas inerentes ao processo de realização de prova de software, foram consultados livros, artigos científicos e publicações nos distintos acervo cientifico sobre o assunto, com o intuito de compreender melhor a prova de software como um elemento fundamental na garantia dessa qualidade. Após a pesquisa inicial, realizou-se uma sintetização dos principais achados nas literaturas a partir de uma leitura minuciosa acerca do tema trabalhado, que levou em consideração os aspectos relevantes de acordo com o objetivo da pesquisa.

PALAVRA CHAVE: qualidade,verificação, validação , prova ,software.

ABSTRACT

This work presents a bibliographical survey on elements that assure a software quality, in order to provide answers to questions such as: what is software quality? Why it's important? How to assure a software quality? and briefly describe these elements such as verification and validation, testing strategy, types of testing, testing levels, testing methods inherent to the software testing process, books, scientific articles and publications in the different scientific collections on the subject, with the aim of better understanding

software testing as a fundamental element in assurance of this quality. After the initial research, a synthesis of the main findings in the literature was carried out based on a thorough reading of the topic discussed, which took into account the relevant aspects in accordance with the objective of the research.

KEYWORD: quality, verification, validation, test, software.

I. INTRODUÇÃO

A produção de um software requer um conjunto de processos que garantam a qualidade do produto final. No desenvolvimento de software a qualidade de um projeto engloba o grau de atendimento às funções e características especificadas no modelo de requisitos[1]. No sentido mais geral, a qualidade de software pode ser definida como: uma gestão de qualidade efetiva aplicada de modo a criar um produto útil que forneça valor mensurável para aqueles que o produzem e para aqueles que o utilizam[1].

Os fatores de qualidade de McCall servem de base para uma engenharia de software que produz altos níveis de satisfação do usuário por se concentrar na experiência do usuário geral propiciada pelo produto de software[1] Esses factores são por categoria revisão do produto(Testabilidade, Flexibilidade), transição do produto(portabilidade, reusabilidade, interoperabilidade), operação do produto(correção, confiabilidade, usabilidade, integridade, eficiência)[1], mas trabalho tem como objectivo demonstrar especificamente um dos elementos na categoria de revisão, a testabilidade como elemento fundamental no asseguramento da qualidade de software bem como a descrição dos elementos que o integram.

II. VERIFICAÇÃO E VALIDAÇÃO

Qualidade segundo, [1] pode significar criar um conjunto de atividades que ajude a garantir que todo artefato resultante da engenharia de software apresenta alta qualidade. Um desses conjuntos de atividades refere-se a prova de software.

A prova de software também é conhecido como **verificação** e **validação**[1]**.**

Em seguida vejamos a definição e a diferença entre ambas:

- **Verificação** refere-se ao conjunto de tarefas que garante que o software implemente corretamente uma função especifica[1].
- **Validação** refere-se ao conjunto de tarefas que asseguram que o software foi criado e pode ser guiado segundo os requisitos do cliente[1].

Ainda vejamos segundo [2], **validação** é o processo de conferir se os requisitos definem o sistema que o cliente realmente quer; já **verificação** é o processo de conferir se o software cumpre com os seus

requisitos funcionais e não funcionais declaradas[2]. Ambos os autores fazem menção de um pioneiro de engenharia de software Barry Boehm na qual baseia-se no mesmo para sintetizar ambos os conceitos , verificação, implica a seguinte questão estamos construindo o produto corretamente ?[2],ou seja o papel da verificação implica comprovar que o software está de acordo com sua especificação; validação, estamos construindo o produto certo ?[2] ou seja validação procura assegurar que o software satisfaz as expectativas do cliente.

A verificação e a validação incluem uma ampla gama de atividades de SQA (software quality assurance – garantia da qualidade de software): revisões técnicas, auditorias de qualidade e configuração, monitoramento de desempenho, simulação, estudo de viabilidade, revisão de documentação, revisão de base de dados, análise de algoritmo, prova de desenvolvimento, prova de usabilidade, prova de qualificação, prova de aceitação e prova de instalação. Embora a aplicação de prova tenha um papel extremamente importante em V&V, muitas outras atividades também são necessárias[1].

III. PROVA

Inicialmente, a atividade de prova era encarada simplesmente como a tarefa de navegar pelo código e corrigir problemas já conhecidos. Tais tarefas eram realizadas pelos próprios desenvolvedores, não existindo recursos dedicados exclusivamente a essa atividade, de maneira que os testes eram realizados tardiamente[3]

A prova de software é uma atividade de fundamental importância para a garantia da qualidade de um produto de software, bem como minimização e identificação de falhas, sendo estas das mais simples às mais complexas de serem solucionadas[4] .

As provas de software podem ser aplicadas a várias áreas do desenvolvimento, como acessibilidade, segurança, usabilidade, entre outros. O ideal é que os projetos façam pelo menos alguns testes dentro de cada área de forma a assegurar um funcionamento adequado do produto que será entregue ao cliente.[5]

Das analises das declarações dos actores define-se prova de software como:

- Processo de verificação de que um programa de software funcione como se esperava;
- Processo de execução de um programa com a intenção de descobrir erros;
- Processos que permitem verificar e revelar a qualidade de um producto de software.

Actividades da prova

Segundo [1] esses são as actividades que compõe a prova de software:

- Planeamento da prova
- Desenvolver estratégia de prova
- Estabelecer os Casos de prova
- Provar
- Avaliar os resultados

E os principais elementos são:

- Tipos de prova.
- Estratégia de prova.
- Métodos de prova.
- Níveis de prova.
- Casos de prova.

Estratégia de prova

- Descreve o enfoque e os objectivos gerais das atividades de prova.[1]
- Inclui os níveis, tipos, técnicas e ferramentas a ser executadas em as provas.[1].

Níveis de prova

As provas se aplicam para diferentes tipos de objectivos, em diferentes cenários ou níveis de trabalho.[1]

- Prova do desenvolvedor
- Prova de unidade
- Prova de integração
- Prova de sistema
- Prova de aceitação.

IV. COMO FAZER AS PROVAS?

- Nos módulos, **provas de unidade**.
- Na união dos módulos, **provas de integração**.
- Quando temos todos unidos, **provas de validação**.
- Quando o sistema está funcionando, **provas do sistema.**

Prova de unidade

Teste de unidade focaliza o esforço de verificação na menor unidade de projeto do software, componente ou módulo de software[1].

- Focada ao código fonte dos componentes.
- Verifica todos os fluxos de controlo
- Primeiro passa pela revisão do desenvolvedor.

Prova de integração

- Prova os componentes combinados para executar um caso de uso.
- Descobre erros ou incompleta especificações das interfaces das classes.

Prova de sistema

Visa verificar se a versão corrente do sistema permite executar processos ou casos de uso completos do ponto de vista do usuário que executa uma série de operações de sistema em uma interface (não necessariamente gráfica) e é capaz de obter os resultados esperados[6]**.** Então pode-se sintetizar que a prova de sistema:

- Prova o software funcionando como um todo.
- Realiza-se quando o software se encontra na fase de Construção.

  Tipos de prova de sistema:

  - Recuperação.
  - Segurança.
  - Resistencia.
  - Rendimento

Prova de aceitação

(Prova Piloto)

O teste de aceitação costuma ser realizado utilizando-se a interface final do sistema. Ele pode ser planejado é executado exatamente como o teste de sistema, mas a diferença é que é realizado pelo usuário final ou cliente, e não pela equipe de desenvolvimento[6].

- Prova final antes da implantação do sistema.
- Geralmente é realizada pelos usuários finais**.**

V. MÉTODOS OU TÉCNICAS DE PROVA

Em relação às técnicas de prova, existem duas grandes famílias:

1. **Provas estruturais ou caixa-branca:** testes executados com conhecimento do código implementado, ou seja, que testam a estrutura do programa em si[6]**.**
2. **Provas funcionais ou caixa-preta:** testes executados sobre as entradas e saídas do programa sem que se tenha necessariamente conhecimento do seu código-fonte[6].

Métodos de prova: Caixa preta

- Realizam-se sobre a interface do software;
- Estão centralizadas nos requerimentos funcionais do sistema**;**
- Tem como objectivo demostrar que as funcões do software são operativas, que as entradas se aceitam de forma adequada e se produz um resultado correto.

Métodos de prova: Caixa preta

Permitem encontrar:

a) Funções incorrectas ou ausentes.
b) Erros de interface;
c) Erros em estruturas de dados ou em acceso aos bancos de dados;
d) Erros de rendimento;
e) Técnica da Partição Equivalente[1];
f) Técnica do Análise de Valores Limites[1];
g) Técnica de Grafos de Causa-Efeito[1].

Métodos de prova: Caixa Branca

- Comprovar os caminhos lógicos do software.
- Comprovar o estado do programa em muitos pontos para determinar se o estado real coincide com o esperado.
- Realiza-se sobre o código.

O que garante o método Caixa Branca?

- Exercitar pelo menos uma vez todos os caminhos independentes para cada módulo;
- Exercitar todas as decisões lógicas em suas variantes verdadeira e falsa;
- Executar todos os ciclos em seus limites e com seus limítes operacionais;
- Exercitar as estruturas internas de dados para assegurar sua validação.

Técnicas Caixa Branca

- ✓ Prova do caminho básico[1].
- ✓ Prova de condição[1].
- ✓ Prova de fluxo de dados[1].
- ✓ Prova de ciclos[1].

Técnicas do caminho básico

1. A partir do desenho ou do código fonte, fazer o grafo de fluxo associado;
2. Calcular a complexidade ciclomática do grafo;
3. Determinar um conjunto básico de caminhos independentes;
4. Preparar os **casos de prova** que garante a execução de cada caminho do conjunto básico.

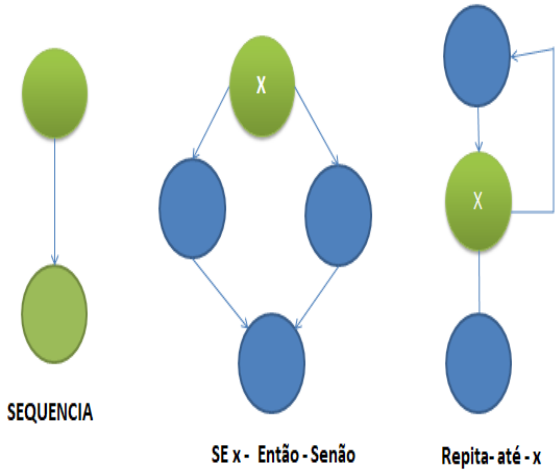

Figura 1 - Regras de grafo de fluxo

1. Um nó por cada linha ou bloco de instrução.
2. Um nó por cada bifurcação.
3. Uma aresta relaciona dois nós

A seguir efectua-se o calculo da complexidade segundo a seguintes formula:

Cálculo da complexidade

C = A – N+2

C= Complexidade.

N= Nós

Vejamos o seguinte exemplo abaixo, sobre um trecho de código:

```
1 {
2     String text="";
3     char L= ReadInChar();
4     if (L=='P') {
5       text= "Professor Nzuzi";
6     }
7     else text= "Estudante";
8     return text;
9 }
```

Figura 2- Trecho de código exemplo

Aplicando a regra segundo a figura 1, apartir do desenho ou do código fonte, fazer o grafo de fluxo associado teremos:

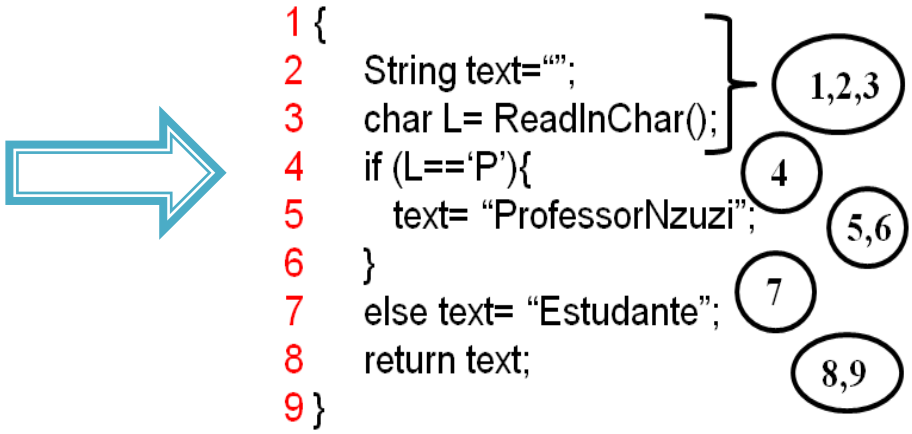

Figura 3 - Regra aplicado no código

Gerando o seguinte grafo de fluxo abaixo:

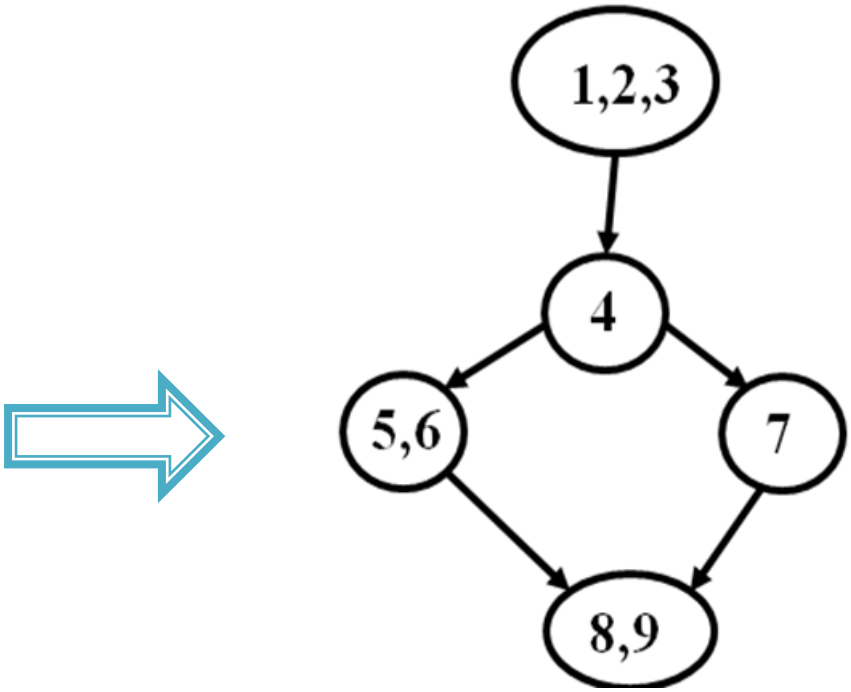

Figura 4 - O grafo do código exemplo

Cálculo da complexidade

C= A-N+2.

C= 5-5+2=2

O que indica dois casos de prova.

Casos de prova ou valor da complexidade ciclomática indica número máximo de execuções necessárias para exercitar todos os comandos do programa[6].

VI. CONCLUSÃO

Portanto pode-se concluir no que toca a construção de um software, a qualidade refere-se ao grau de atendimento ou cumprimento das funções e características especificadas, de tal modo fornecer um valor mensurável para quem utilizará, bem como para quem o produziu. A qualidade é importante pois é necessário que se tenha um produto que satisfaça os requisitos que lhe foram incubidos.

Detalhou-se os factores base da engenharia de software para que se atinja a qualidade em um software como o de **revisão do produto**(Testabilidade, Flexibilidade), **transição do produto**(portabilidade, reusabilidade, interoperabilidade), **operação do produto**(correção, confiabilidade, usabilidade, integridade, eficiência) , e como a prova de software é essencial para garantir o cumprimento desses factores afim de assegurar a qualidade que se espera em um software.

A bibliografia utilizada são conceituadas e das mais relevantes no mercado de engenharia de software e permitiram uma descrição clara e suscinta do processo de valiadação e verificação que resume-se em prova de software.

Este trabalho demostra limitação na descrição de mais técnica de prova de software bem como ferramentas e tecnologias que facilitem e garante eficiência dos mesmo pelo que se deixa como sugestão.

VII.REFERÊNCIAS

[1] B. R. M. Pressman, Roger S. e, *Engenharia de software: uma abordagem profissional*, 9ª edição. Rua Ernesto Alves, 150 – Bairro Floresta 90220-190 – Porto Alegre – RS Fone: (51) 3027-7000 SÃO PAULO Rua Doutor Cesário Mota Jr., 63 – Vila Buarque 01221-020 – São Paulo – SP, 2021.

[2] T. L. C. Ian Sommerville; Queiroz, *Engenharia de Software 10ª Ed. - Ian Sommerville.pdf*, 10 edição. São Paulo - Brasil, 2019.

[3] F. Barreto, V. Benitti, and E. L. Albano, “Teste de Software: o que e como é ensinado?,” *Univ. Parana. Av. Júlio Assis Cavalheiro*, vol. 1, no. 2, pp. 360–88302, 2004, [Online]. Available: http://www.lbd.dcc.ufmg.br/colecoes/wei/2012/0023.pdf

[4] M. Jaqueline and A. G. O. Fassbinder, “Uma Investigação sobre o uso da Aprendizagem Baseada em Casos como Apoio ao Ensino de Teste de Software,” *v. 15 n. 3 15ª Jorn. CIENTÍFICA E TECNOLÓGICA DO IFSULDEMINAS*.

[5] Lara, “No Titleיכה השק תוארל תא המ תמאבש דגנל מיניעה,” *הארץ*, no. 8.5.2017, pp. 2003–2005, 2022, [Online]. Available: www.aging-us.com

[6] R. S. Wazlawick, *Engenharia de software: conceitos e práticas*, vol. 1. 2013. [Online]. Available: file:///C:/Users/matth/Documents/2022/2_FACULDADE/TCC - 01/Livro - Engenharia de software conceitos e pratica - Wazlawick.pdf.