**Diferença entre Validação, Verificação, teste**

**1. Verificação: Fizemos o software corretamente?**

Esta atividade se resume em responder a esta pergunta. A verificação tem o objetivo de avaliar se o que foi planejado realmente foi realizado. Ou seja, se os requisitos e funcionalidades documentados foram implementados, além disso a verificação também pode ser realizada para especificação de sistemas, para avaliar se os requisitos estão sendo documentados como deveriam e ainda prever falhas ou inconsistências entre requisitos.

**2. Validação: Fizemos o software correto?**

A validação tem o objetivo de avaliar se o que foi entregue atende as expectativas do cliente. Ou seja, se os requisitos, independente do que foi planejado, estão sendo implementados para atender a regra de negócio do cliente, se o sistema é realmente aquilo que o cliente quer e está pagando para ter. A validação final do sistema é realizada pelo próprio cliente ou usuário.

**3. Teste de Software: O software tem defeitos!**

O teste de software é considerado uma técnica dinâmica de verificação e validação, pois o software é executado com dados de teste e seu comportamento é analisado. Diferentemente da inspeção de software ou, também chamada de revisão por pares, que é considerada uma técnica estática, pois não é necessário executar o software em um computador.

O **objetivo da validação e da verificação** é assegurar que o sistema seja adequado, e se atende as necessidades, ou seja, a confirmação de que este cumpra suas especificações.

A **Verificação** é uma atividade, a qual envolve a análise de um sistema para certificar se este atende aos requisitos funcionais e não funcionais.

A **Validação** é a certificação de que o sistema atende as necessidades e expectativas do cliente.

O processo de Validação e Verificação, não são processos separados e independentes.

**Tipos de Teste**

**Teste de funcionalidade:** Aqui o objetivo é avaliar a interface do programa, os links, campos de preenchimento, botões etc.

Uma ferramenta que pode ser útil nesse caso é o Push Test Maker. Para aplicações web, temos o Apodora, o Sikuli e o Watir.

**Teste de desempenho:** Esse teste visa avaliar a performance de um programa, se ele atende aos requisitos mínimos preestabelecidos, como o tempo que demora para dar respostas a ações realizadas e se ele rende como o esperado.

Um software que pode ajudar é o Apache Jmater, que pode ser usado em aplicações Web.

**Teste unitário:** Esse visa avaliar pequenas unidades que compõem um software, responsáveis por funções diferentes dentro dele. Podem ser avaliados códigos, sub-rotinas entre outros. O foco aqui é descobrir se todas essas partes estão funcionando adequadamente.

As ferramentas que podem ajudar aqui são: NUnit e JUnit (para aplicações Java).

**Gestão dos testes** – Existem ferramentas que auxiliam no gerenciamentos dos testes a serem realizados, permitindo acompanhamento e controle dos casos e atividades de testes .

Entre elas temos o TestMaster e o Testlink.

**Bugs e defeitos** – Há ferramentas que são importantes para se encontrar bugs e erros durante o desenvolvimento dos programas.

Podemos destacar o Redmine, Mantisbt e o Bugzilla.

**Tabela de descrição testes**

|  |  |
| --- | --- |
| **Nome Teste** | **Descrição** |
| Teste de Unidade | Teste em um nível de componente ou classe. É o teste cujo objetivo é um “Pedaço do Código”. |
| Teste de Integração | Garante que um ou mais componentes combinados(ou unidades) funcionam. Podemos dizer que um teste de integração é composto por diversos teste de unidade. |
| Teste Operacional | Garante que a aplicação pode rodar muito tempo sem falhar. |
| Teste Positivo-Negativo | Garante que a aplicação vai funcionar no “Caminho Feliz” de sua execução e vai funcionar no seu fluxo de exceção. |
| Teste de Regressão | Toda vez que algo for mudado, deve ser testada toda a aplicação novamente. |
| Teste Caixa-Preta | Testar todas as entradas e saídas desejadas.Não se esta preocupado com código, cada saída indesejada é vista como um erro. |
| Teste Caixa-Branca | O objetivo é testar o código. Às vezes, existem partes do código que nunca foram testadas. |
| Teste Funcional | Testar as funcionalidades, requerimentos, regras de negócio presentes na documentação. Validar as funcionalidades descritas na documentação(pode acontecer de a documentação estar inválida). |
| Teste de Interface | Verifica se a navegabilidade e os objetivos da tela funcionam como especificados e se atendem da melhor forma ao usuário. |
| Teste de Performance | Verifica se o tempo de resposta é o desejado para o momento de utilização da aplicação. |
| Teste de Carga | Verifica o funcionamento da aplicação com a utilização de uma quantidade grande de usuários simultâneos. |
| Teste de Aceitação do Usuário | Teste se a solução será bem vista pelo usuário. Ex: Caso exista um botão pequeno demais para executar uma função, isso deve ser criticado em fase de testes. (aqui, cabem requisitos fora da interface, também). |
| Teste de Volume | Testar a quantidade de dados envolvidos(pode ser pouca, normal, grande ou além de grande). |
| Teste de Stress | Testar se a aplicação sem situações inesperadas. Testar caminhos, às vezes, antes não previstos no desenvolvimento/documentação. |
| Teste de Configuração | Testar se a aplicação funciona corretamente em diferentes ambientes de hardware ou de software. |
| Teste de Instalação | Testar se a instalação da aplicação foi OK. |
| Teste de Segurança | Testar a segurança da aplicação das mais diversas formas. Utilizar os diversos papéis, perfis, permissões, para navegar no sistema. |

**Teste Estrutural = Caixa Branca**

**Teste Funcional = Caixa Preta**