Software Testing Course

Visão geral dos testes de software

Tópicos a serem cobertos

Visão Geral do Teste de Software
SDLC & Modelos
Principios de Testes
Ciclo de vida útil do Teste de Sofware
Tipos de teste de software
Métodos de teste de software
Níveis de teste de software
Documentação de teste de software
Processo de gestão de Defeitos
Teste de automação
Seleniun
Localizadores Seleniun

BUG

Teste de Desempenho

Atividade para avaliar o funcionamento do
seu Software com carga crescente de requisições

Testes Funcionais

Recomendado para sistemas que possuem
documentação e objetivo e validação de requisitos

Testes Automatizados

Recomendado para otimizar o tempo de
ciclo em ambiente iterativo incremental

Testes de Carga

Utilizado para avaliar requisições simultâneas
e paralelas

Testes de Estresse

Atividade para avaliar o tempo de resposta
do software quando o mesmo está no limite
especificado

Testes de Invasão

Recomendado para analisar o sistema pela otica
do invasor

Testes de Exploratórios

Ideal para software que possuem documentação e
pouco prazo para validação e testes.

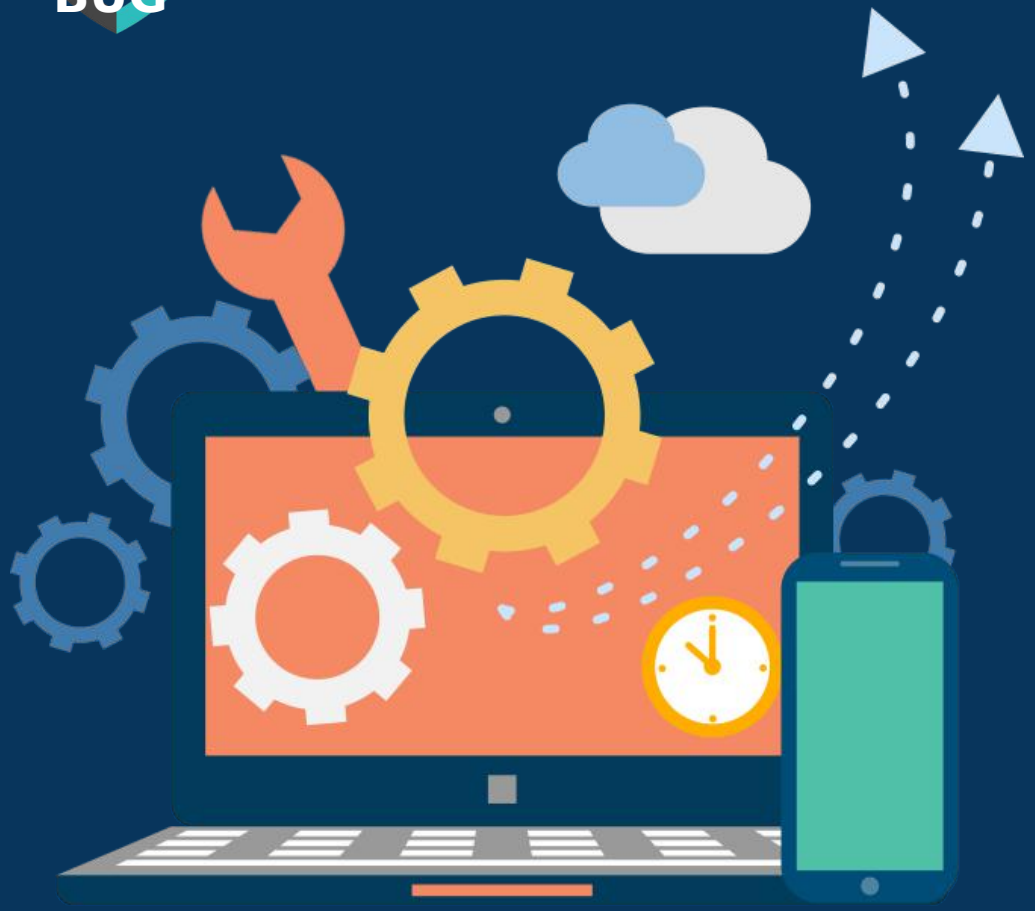
Testes de Compatibilidade

Atividade para avaliar a compatibilidade e
portabilidade do software a ambientes e dispositivos

Testes de Usabilidade

Recomendado para verificar o seus software é
fácil entendimento pelos usuários finais, intuitivo e
fácil de operar.

O que é teste de software?



O TESTE DE SOFTWARE é definido como uma atividade para verificar se os resultados reais correspondem aos resultados esperados e para garantir que o sistema de software esteja livre de defeitos.

Envolve a execução de um componente de software ou de sistema para avaliar uma ou mais propriedades de interesse.

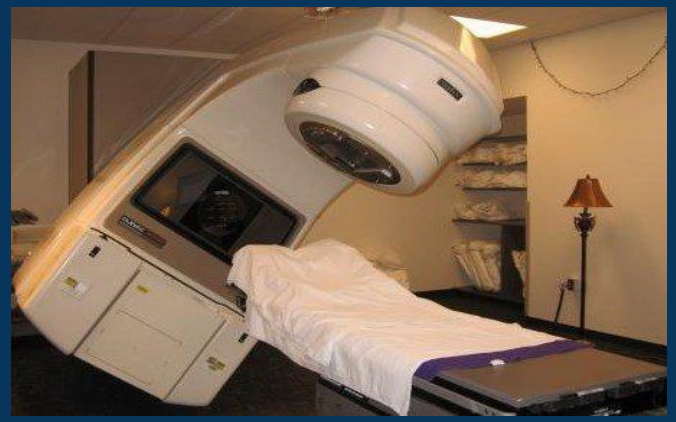
O teste de software também ajuda a identificar erros, lacunas ou falta de requisitos, ao contrário dos requisitos reais.

Pode ser feito manualmente ou usando ferramentas automatizadas.

O teste é importante porque os erros de software podem ser caros ou até perigosos. Os erros de software podem potencialmente causar perdas monetárias e humanas, e o histórico está cheio desses.

SYSTEM FAILURE

Por que testar é importante?



Do ano de 1985 até o ano de 1987, os hospitais dos EUA utilizavam uma máquina chamada Therac-25 para o tratamento com radiação contra o câncer.Eeste modelo apresentava um problema de programação no seu software, expondo os pacientes a uma intensidade de radiação 100 vezes maior do que a recomendada

Acidentalmente, em 2009, um programador da Google adicionou uma barra invertida em todas as URLs da gigante de Mountain View que eram direcionadas para o buscador da empresa. Dessa maneira, o site foi sinalizado como malware no mundo todo por cerca de uma hora, gerando um prejuízo total de quase US$ 3 milhões (quase R$ 7 milhões).

Em 1992, a Pepsi contava com uma promoção que quem tirasse a tampinha com o número 349 ganharia um prêmio considerável em dinheiro, resultou em 800 mil tampas com esta numeração nas Filipinas, gerando a mesma quantia de ganhadores.
A empres não pagou a “riqueza”

A empresa Knight Capital Group (ou apenas KCG) trabalha há algum tempo com investimentos. Porém, em 2012, gerou milhares de negociações que não poderia ser feitas. Desse modo, em apenas meia hora, a KCG perdeu US$ 440 milhões

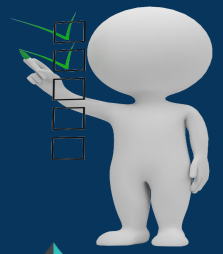
Por conta da falha de um sistema de alarme, em 2003 os Estados Unidos enfrentaram o seu maior apagão, conhecido como “The Great Northeast Blackout”. lançando cerca de 50 milhões de pessoas no escuro, o que resultou em 11 mortes e um prejuízo de US$ 6 bilhões (R$ 13,8 bilhões) ao governo.

https://www.tecmundo.com.br/tecnologia/49572-bugs-10-falhas-de-computadores-que-causaram-muito-prejuizo-e-confusao.htm

Quem faz o teste?

Software Testers

Lider do Time



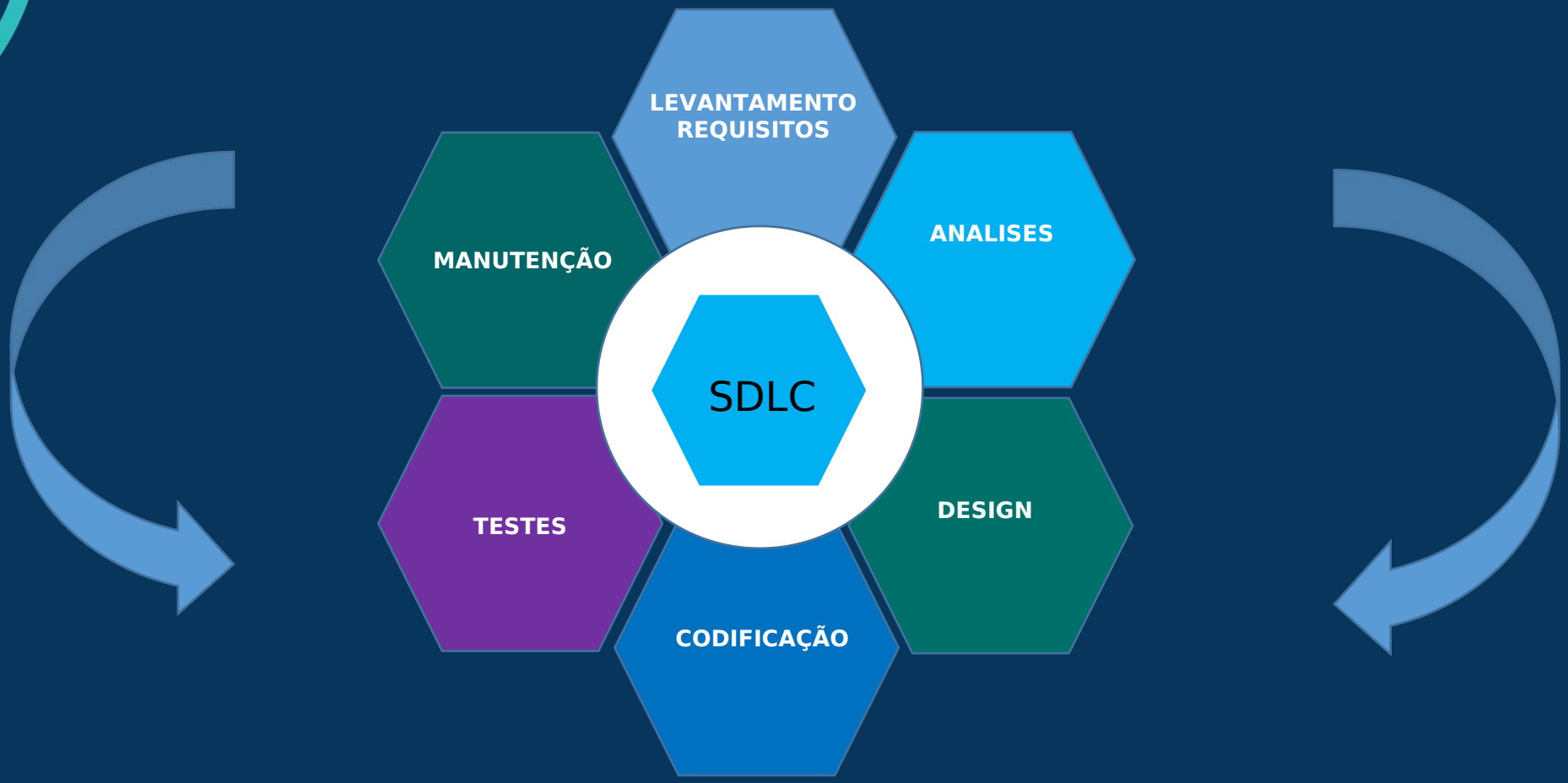
Desenvolvedores do Projeto

Lider do Projeto

Usuário Final

BUG

Ciclo de vida do desenvolvimento de software



O Modelo em Espiral de Boehm

Criado por Barry Boehm em 1988, o Modelo em Espiral é uma melhoria do Modelo Incremental e possui esse nome por causa de sua representação, onde cada volta no espiral percorre todas as fases do processo de software. As voltas devem ser repetidas quantas vezes forem necessárias até que o software possa ser completamente entregue.

Principios do Teste de Sofware

100% de qualidade

O teste sem erros
é um mito

teste é contexto
Dependencia

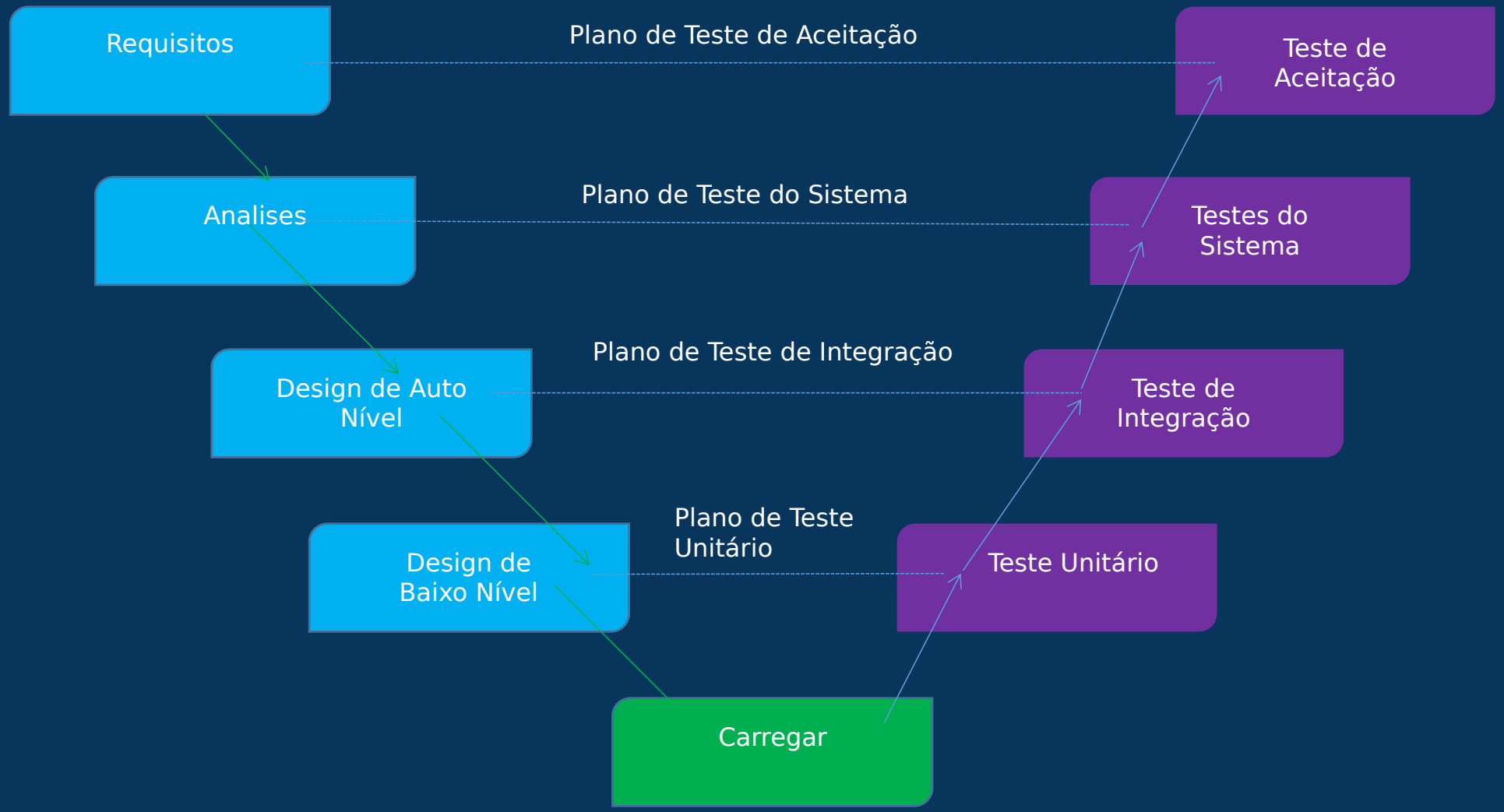
Defeito no cluster

Testes Antecipados

Teste de
Efetividade

detecção de Erros

Verificação e Validação Modelo Testes Software



Modelo em V

Analise de Requisitos

Plano de Testes

Caso de Teste

Desenvolvimento

Configurar Ambiente

Execução dos Testes

Fechamento Ciclo de Testes

O ciclo de vida consiste em uma série de etapas dependentes, consideradas como o esqueleto do Processo de Teste, que visam estruturar as atividades definindo como os testes serão conduzidos no projeto.

Ciclo de Vida Testes de Software

BUG

Testes de software
Tipos

Testes Manual

Teste manual significa testar um aplicativo manualmente por um ser humano. Um especialista em garantia de qualidade (testador) que executa testes manuais garante que um aplicativo esteja funcionando corretamente seguindo as condições descritas nos casos de teste. O testador avalia o design, a funcionalidade e o desempenho do aplicativo verificando vários elementos.
O teste manual é útil quando o teste automatizado não é possível.

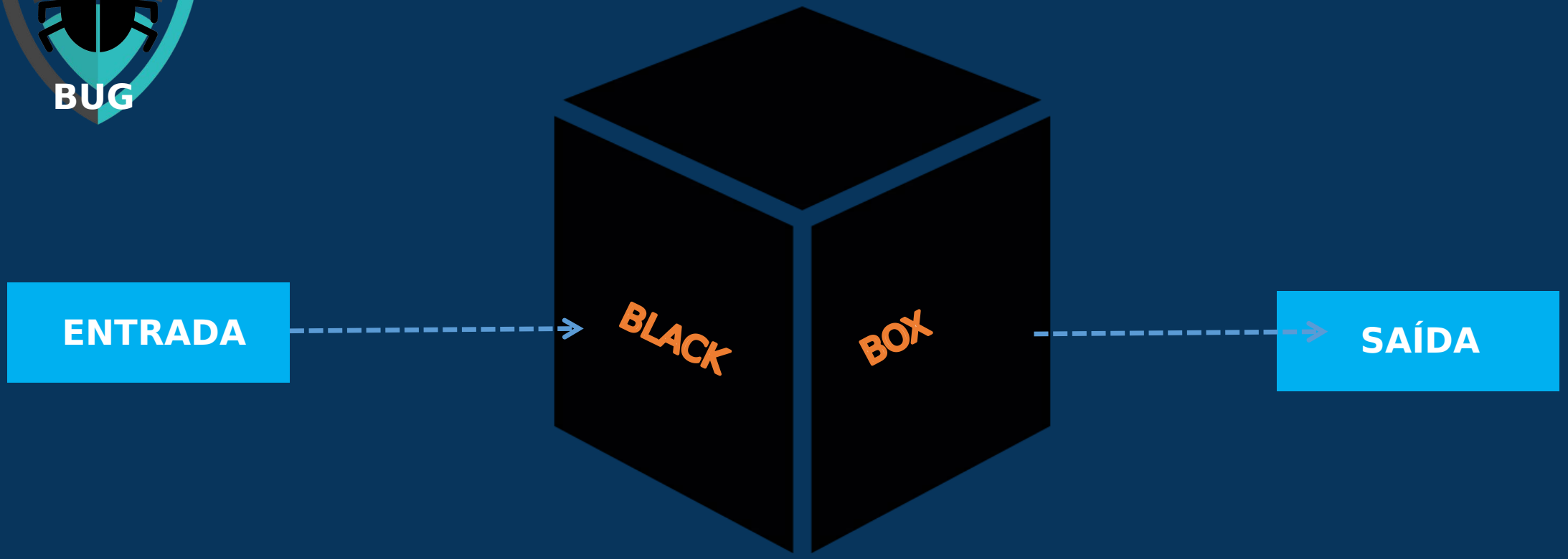
Testes Automáticos

Nos testes automatizados, existem testes pré-programados que são executados automaticamente. Os testes são executados para comparar os resultados reais com os resultados esperados.
Os testes automatizados ajudam a determinar se o aplicativo tem o desempenho esperado ou não.
Testes automatizados são úteis ao executar testes repetitivos e testes de regressão para garantir que um aplicativo funcione corretamente após a implementação de novas alterações.

Tipos de Testes de Software

Desafios dos Testes de Software

Testes de Software
Metódos



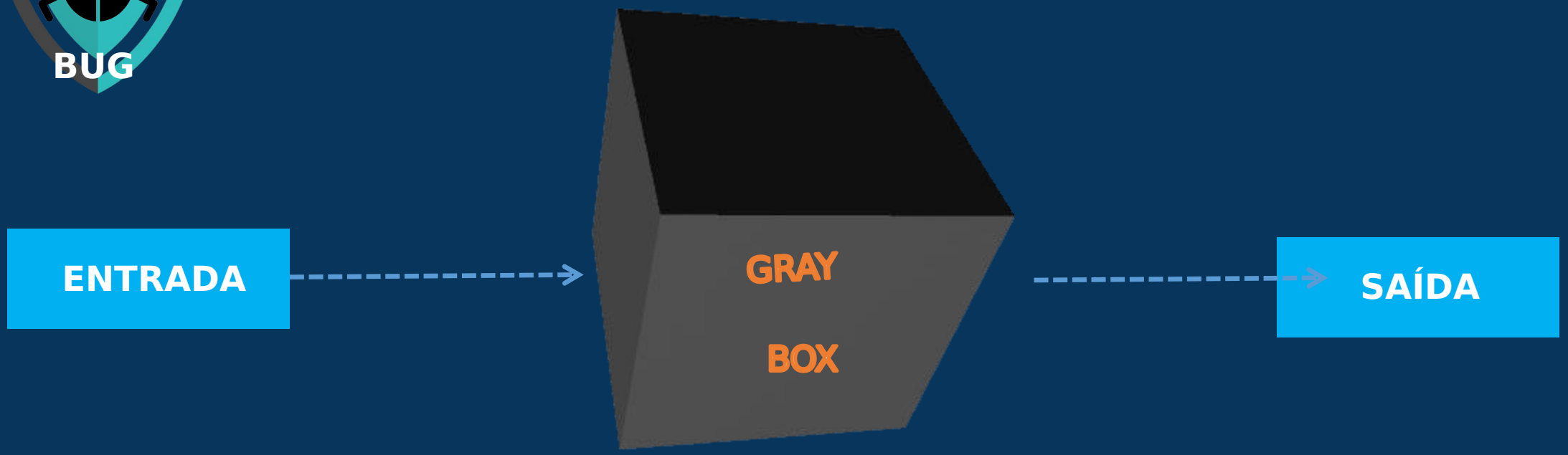
Também chamada de teste funcional, teste comportamental, orientado a dado ou orientado a entrada e saída, a técnica de caixa-preta avalia o comportamento externo do componente de software, sem se considerar o comportamento interno do mesmo.
Dados de entrada são fornecidos, o teste é executado e o resultado obtido é comparado a um resultado esperado previamente conhecido.
Como detalhes de implementação não são considerados, os casos de teste são todos derivados da especificação.

BUG

ENTRADA

SAÍDA

Também chamada de teste estrutural ou orientado à lógica, a técnica de caixa-branca avalia o comportamento interno do componente de software. Essa técnica trabalha diretamente sobre o código fonte do componente de software para avaliar aspectos tais como: teste de condição, teste de fluxo de dados, teste de ciclos, teste de caminhos lógicos, códigos nunca executados.



A técnica de teste de caixa-cinza é uma mescla do uso das técnicas de caixa-preta e de caixa-branca. Esta técnica analisa a parte lógica mais a funcionalidade do sistema, fazendo uma comparação do que foi especificado com o que esta sendo realizado. Usando esse método, o testador comunica-se com o desenvolvedor para entender melhor o sistema e otimizar os casos de teste que serão realizados.[8] Isso envolve ter acesso a estruturas de dados e algoritmos do componente a fim de desenvolver os casos de teste, que são executados como na técnica da caixa-preta.

FUNCIONAL VS NÃO FUNCIONAL

Realizado antes do teste não funcional

Baseados em requisitos de clientes

Descrevem o que os produtos fazem

Teste de Unidade. Teste de aceitação, Teste de fumaça, Teste de integração, Teste de regressão

Realizado após testes funcionais

Baseado nas expectativas dos clientes

Descreve como os produtos funcionam

Teste de desempenho, escalabilidade, Teste de volume, teste de carga, Teste de estresse

Testes de Software
Níveis

Níveis

Teste de Unidade

Teste de integração

Teste do sistema

Teste de Aceitação

Testes de Software
Documentação

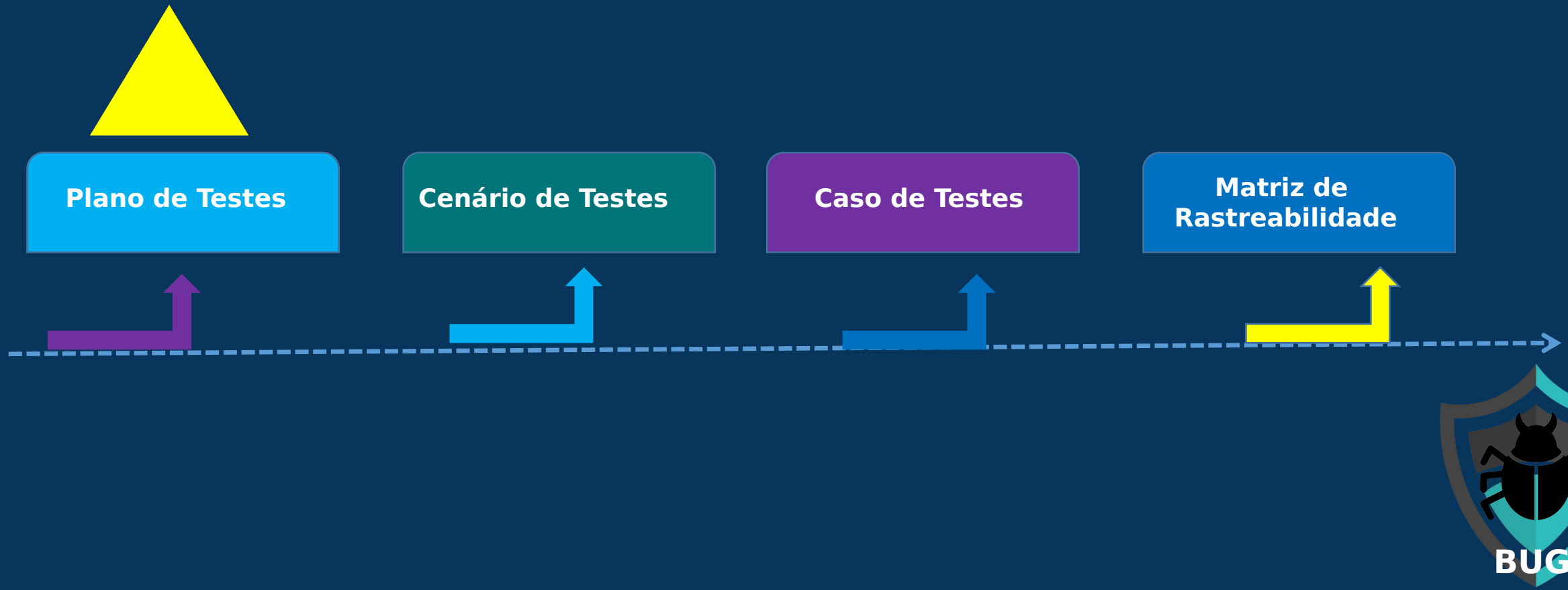
BUG

Artefatos da Documentação

Artefatos da Documentação

O **plano de teste** é um dos documentos produzidos na condução de um projeto. Ele funciona como:

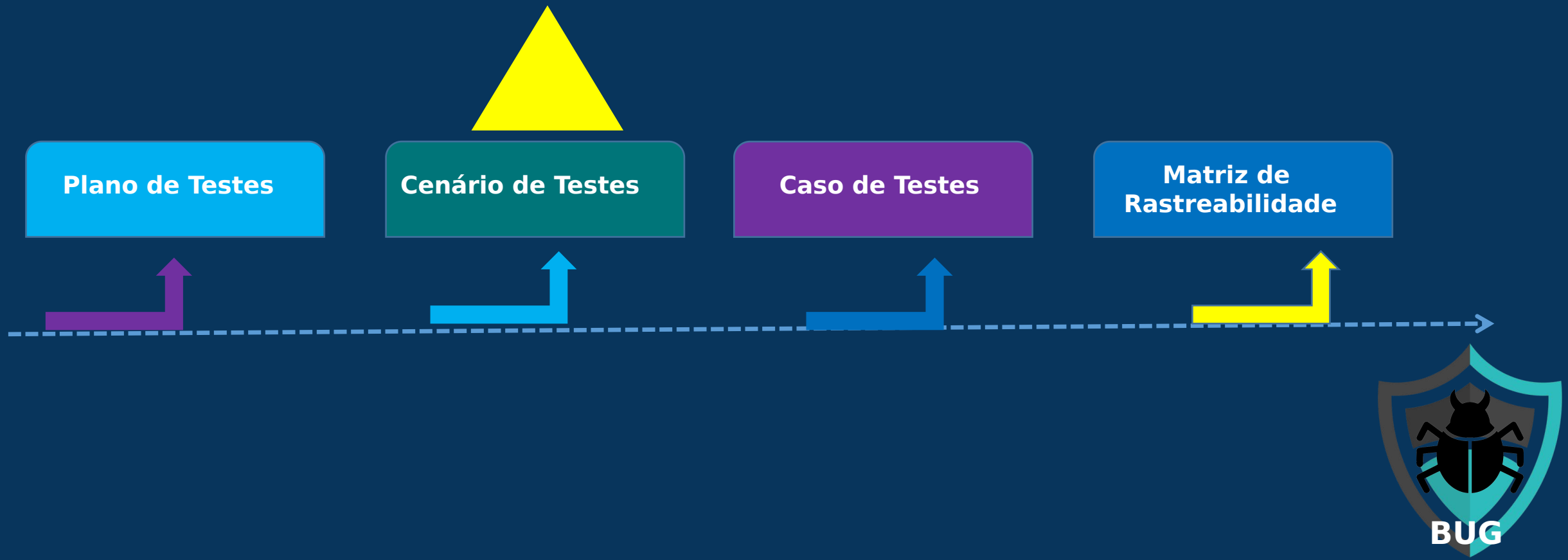
- Um ‘integrador’ entre diversas atividades de testes no projeto;
- Mecanismo de comunicação para os stakeholders (i.e. a equipe de testes e outros interessados);
- Guia para execução e controle das atividades de testes.



BUG

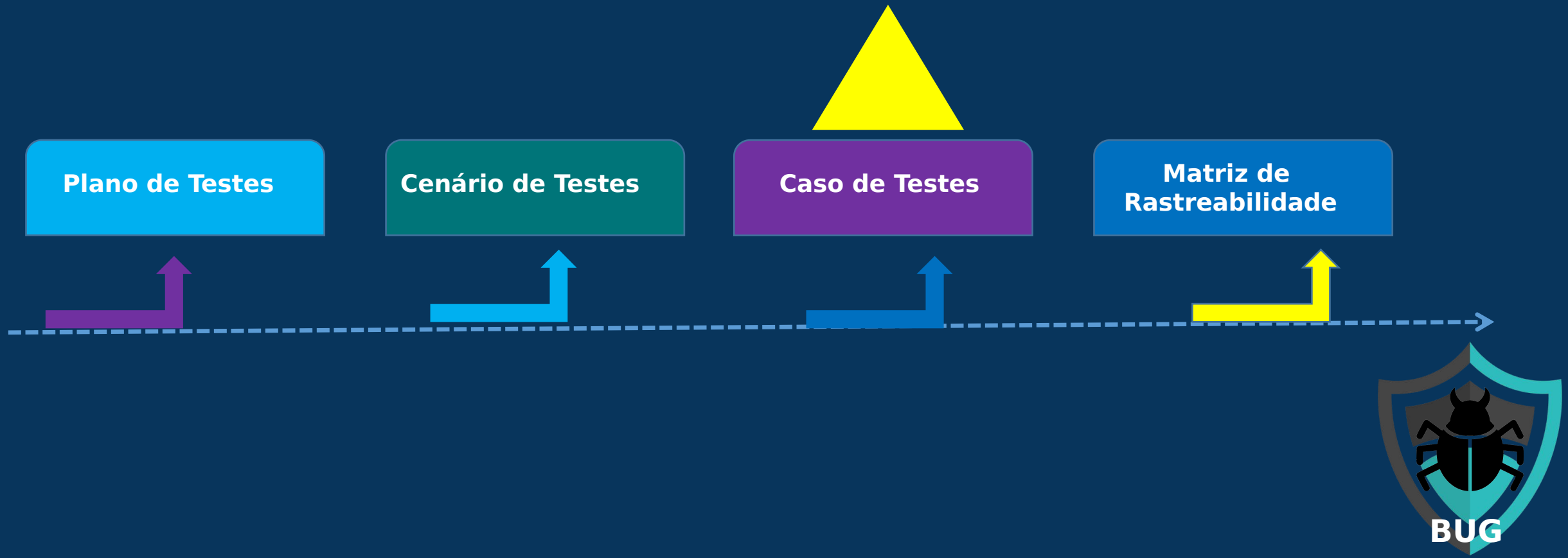
Artefatos da Documentação

O tipo menos detalhado de documentação é o **cenário de teste**. Um cenário de teste é uma descrição de um objetivo que o usuário pode encontrar ao utilizar o programa. Um exemplo seria “Testar se um usuário consegue deslogar do programa ao fechá-lo”. Tipicamente, um cenário de teste vai precisar de diferentes tipos de testes para garantir que o objetivo tenha sido bem testado.



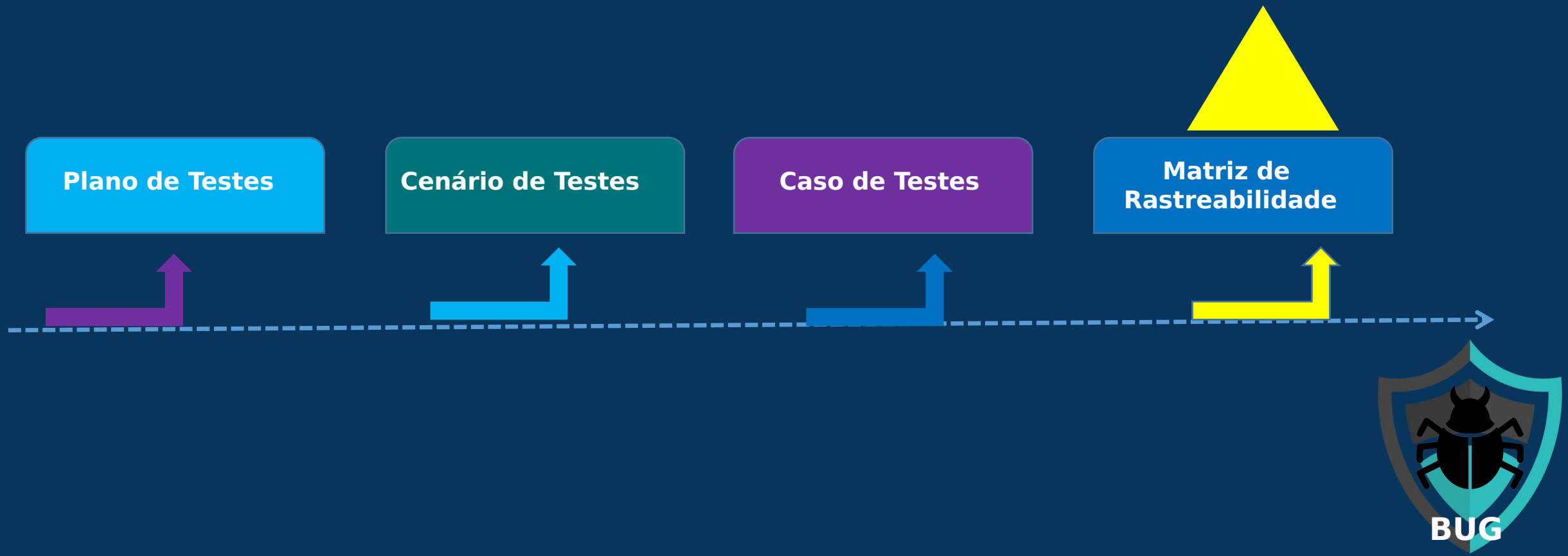
Artefatos da Documentação

A segunda forma mais detalhada de documentar o trabalho de um teste são os casos. Os **casos de teste** descrevem uma ideia específica a ser testada, sem detalhar os dados necessários e etapas exatas a serem executadas.
Por exemplo, um caso de teste poderia ser “Testar se um código de desconto pode ser aplicado em um produto em promoção”. Isso não descreve quantos vão ser códigos ou como serão utilizados. A forma de testar este caso pode variar de tempos em tempos.



Artefatos da Documentação

A **matriz de rastreabilidade** de requisitos é o modelo utilizado para visualizar a evolução de diferentes requisitos e seus pontos de interseção. Como é comum haver alteração nesses itens durante a execução do projeto, fica mais fácil enxergar onde será necessário promover mudanças sem deixar passar nenhum item importante.
Para que seja efetiva, a matriz de rastreabilidade deve ser feita após a determinação desses itens.
É natural que haja um refinamento dessas necessidades, porém a matriz não deixa a origem e o motivo do requisito se perderem



Gerenciamento de Defeitos

Processo de Gerenciamento de Defeitos(Bug)

01 Detecção de defeitos

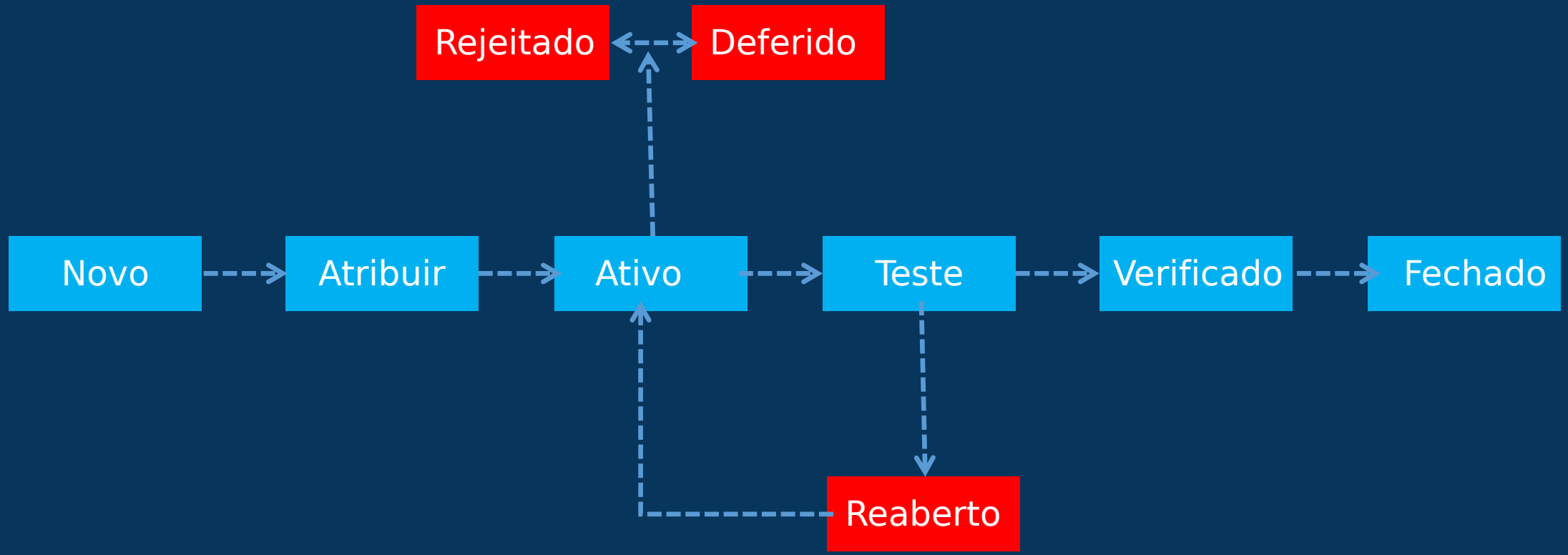
02 Formulação de relatórios de bugs

03 Corrigindo erro

04 Criação de lista de bugs

Um processo de gestão de defeitos tem o objetivo de definir práticas para prevenir os defeitos e minimizar os riscos de um projeto. A utilização de uma ferramenta automatizada, além de oferecer uma base comum para a entrada de informações, também oferece um meio para fomentar a integração entre o time de desenvolvimento e o time de testes.
Além disso, por meio dos relatórios de gestão e métricas geradas por essas ferramentas, os gestores do projeto poderão promover a melhoria contínua do processo estabelecido.

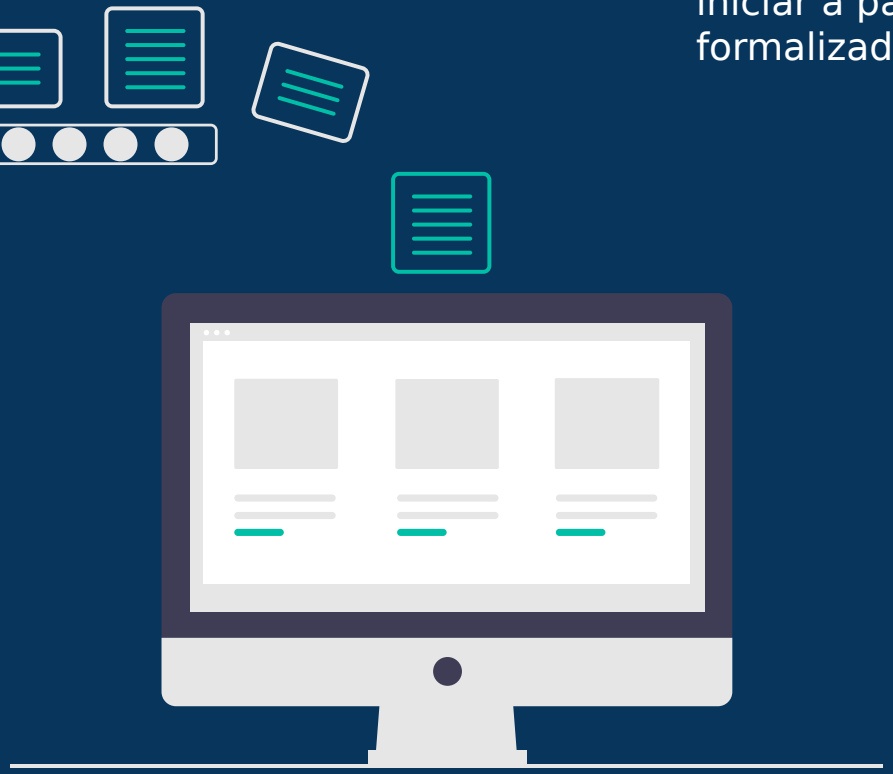
Defeitos(Bug) Ciclo de Vida



Testes Automatizados

O que é Testes Automatizados

Automação de teste é o uso de software para controlar a execução do teste de software, a comparação dos resultados esperados com os resultados reais, a configuração das pré-condições de teste e outras funções de controle e relatório de teste. De forma geral, a automação de teste pode iniciar a partir de um processo manual de teste já estabelecido e formalizado.



Ferramentas de Teste Automatizado

Selenium

Watir

TestComplete

HPE Unified Functional Testing

Telerik Test Studio

Ranorex

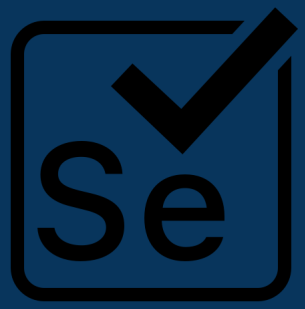
Robotium

Cucumber

TestingWhiz

O que é o Seleniun

Selenium é uma estrutura portátil para testar aplicativos da web. O Selenium fornece uma ferramenta de reprodução para criar testes funcionais sem a necessidade de aprender uma linguagem de script de teste (Selenium IDE).
Ele também fornece uma linguagem específica de domínio de teste (Selenese) para escrever testes em várias linguagens de programação populares, incluindo C #, Groovy, Java, Perl, PHP, Python, Ruby e Scala.
Os testes podem ser executados nos navegadores mais modernos.
O Selenium é executado no Windows, Linux e macOS.
É um software de código aberto lançado sob a Licença Apache 2.0.



THANK YOU