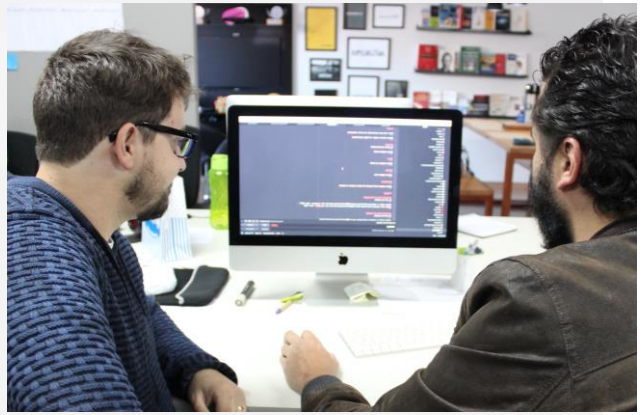
Engenharia de Software I

- Testes de software (parte 1)

Elicitação/
Especificação
de requisitos

Projeto de software

Plano de testes



Codificação

Testes de software

Implantação

Manutenção
Evolução

GERÊNCIA DE PROJETOS

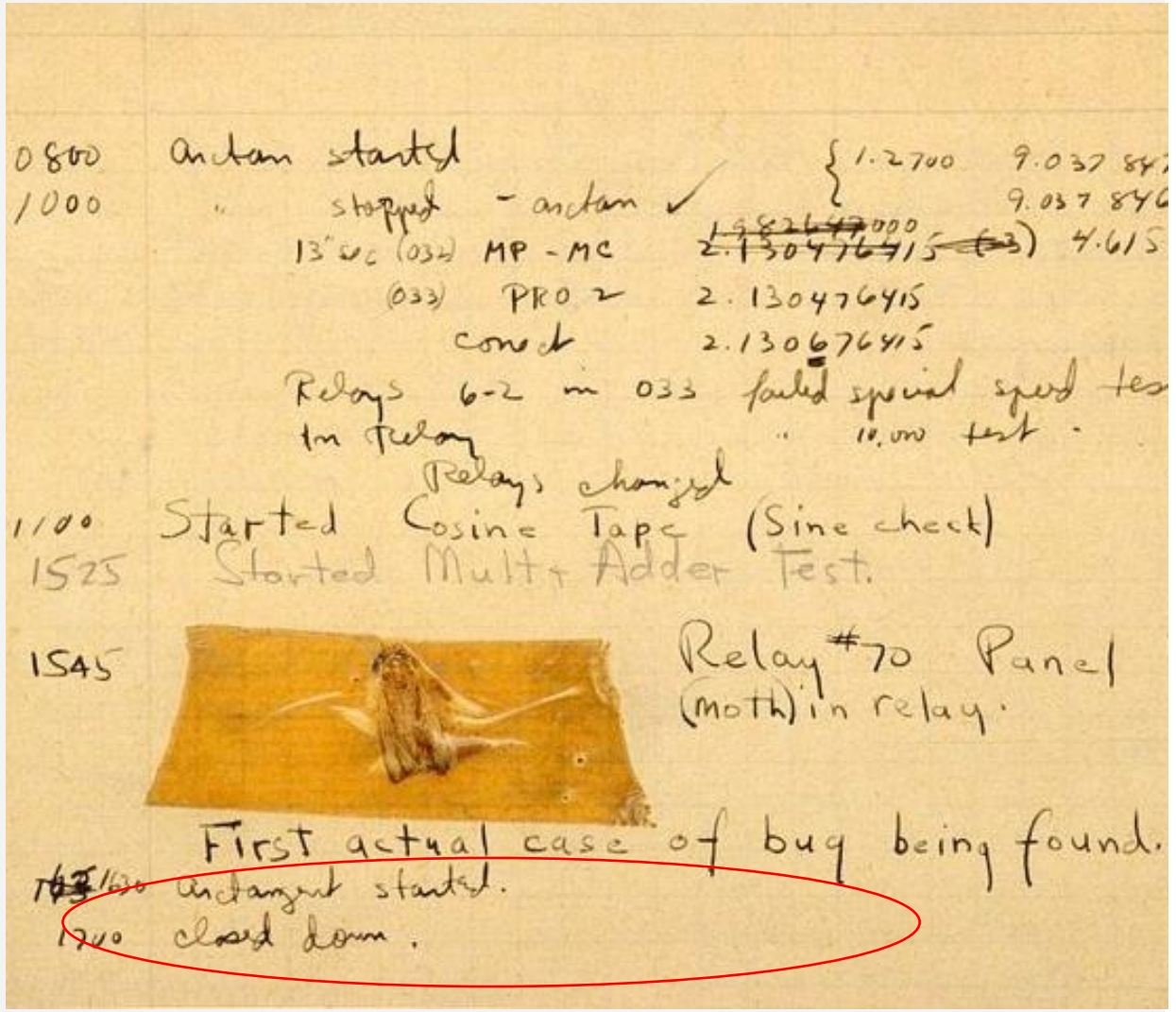
Bugs podem causar catástrofes

Primeiro bug da história

Computador Mark II

Universidade de Harvard em 1945

- O inseto foi descoberto por Grace Hoper. Ao verificar o motivo da pane no computador identificou uma mariposa nos contatos de um relê era a causa do problema.
- O fato ocorreu em 1945 e acredita-se que foi ele que deu a origem do temo “bug” como erro do computador.
- Grace tirou o inseto e colocou em seu caderno de anotações e escreveu: “primeiro caso de bug realmente encontrado”



Bugs famosos

Desastre: Um foguete com uma sonda espacial para Vênus, foi desviado de seu percurso de voo logo após o lançamento. O controle da missão destruiu o foguete 293 segundos após a decolagem (1962).

Custo: 18,5 milhões dólares

Causa: Um programador, ao passar para o computador uma fórmula que haviam lhe entregado escrita manualmente, se esqueceu de uma barra. Sem ela, o software tratava variações normais de velocidade como se fossem sérios problemas, causando falhas por tentativas de correções que acabaram por enviar o foguete fora do curso.

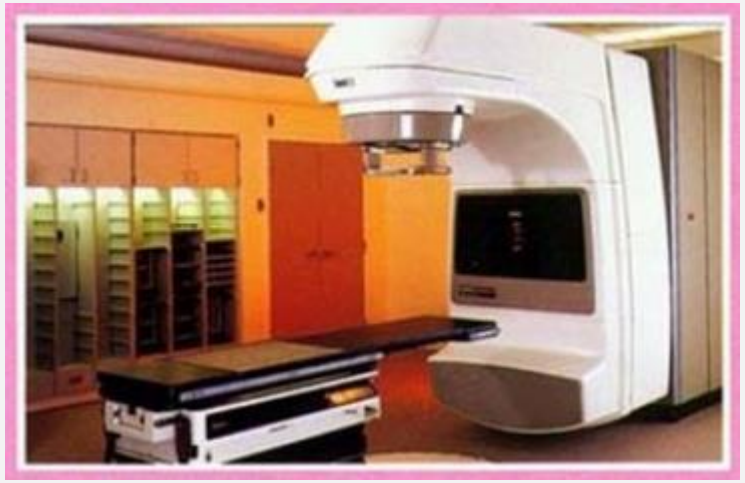
https://pt.wikipedia.org/wiki/Mariner_1

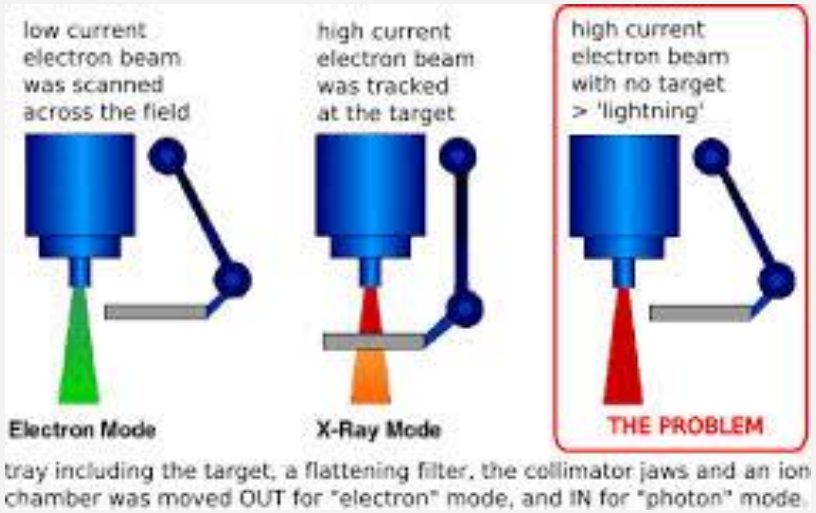
Bugs famosos

Desastre: A máquina de radiação Therac-25 irradiou doses letais em pacientes (1985)

Custo: Três mortos e três seriamente feridos

Causa: O projeto continha travas de hardware para prevenir que o feixe de elétrons de alta intensidade fosse aplicado sem o filtro estar em seu lugar. *Overflows* faziam o software não executar os procedimentos de segurança





tray including the target, a flattening filter, the collimator jaws and an ion chamber was moved OUT for "electron" mode, and IN for "photon" mode.

https://pt.wikipedia.org/wiki/Therac-25

Bugs famosos

Desastre: Durante a primeira Guerra do Golfo, um sistema americano de mísseis na Arábia Saudita falhou ao interceptar um míssil vindo do Iraque. O míssil destruiu acampamentos americanos (1991)

Custo: 28 soldados mortos e mais de 100 feridos.

Causa: Um erro de arredondamento no software calculou incorretamente o tempo, fazendo com que o sistema ignorasse os mísseis de entrada. A cada 100 horas o relógio interno do sistema desviava 1/3 de segundo

Novo contexto de uso de software

- Internet das coisas: mais “coisas” conectando as pessoas
- Sensores para monitoramento contínuo, roupas, aparelhos integrados, celulares, PCs, tablets
- Integração inteligente
- Big data (dados gerados em grade escola)
- Aplicativos em smartphones que acessam todos os tipos de aplicação (inclusive sistema críticos)
- Automóveis (recursos de direção, segurança, usabilidade, comunicação com outros dispositivos, carros autônomos
- Corpo humano/saúde: equipamentos médicos, auxilio a pessoas com deficiência, monitoramento (batimento cardíaco, temperatura), implantes

O que vocês entendem por

qualidade de software?

Como deve ser realizada esta atividade na prática?

Garantia da Qualidade X Testes

A **atividades de garantia da qualidade** de um **produto de software** é o **teste**, para certificar se de sua aderência aos requisitos especificados:

- Eliminar erros
- Errar é humano
- Aumentar a qualidade
- Reduzir os custos

O que vocês entendem por

teste de software?

O que é testar?

Testar é o processo de **executar um programa** ou sistema com a intenção de **encontrar defeitos** *(Myer, 1979)*

- *Objetivo:* Demonstrar que o software atende aos requisitos

Testar é **verificar se o software está fazendo o que deveria fazer**, de acordo com seus requisitos *(Rios e Moreira, 2002)*

- *Objetivo:* Descobrir situações em que o software se comporta de maneira incorreta, indesejável ou de forma diferente das especificações

O objetivo não é mostrar que o programa está correto, mas sim revelar a presença de defeitos caso estes existam

Garantia e Controle da Qualidade

Validação → **Estamos construindo o produto certo?** *(avaliação do atendimento aos requisitos)*

Verificação → **Estamos construindo o produto de forma correta?** *(avaliação da aderência aos padrões da empresa e sem falhas)*

Testes → Atividades de V&V dinâmica

Teste de software

1. **Executar** um programa utilizando algumas entradas de dados

2. Após se **verificar** se o **comportamento** está de acordo com o **esperado**.

Se os **resultados obtidos** coincidem com os **resultados esperados**, então nenhum defeito foi identificado

→"***O software passou no teste***"

Se o **resultado** obtido for **diferente do esperado**, então um defeito foi detectado

→ "***O software não passou no teste***"

Teste de software

A **idéia básica dos testes** é que os **defeitos podem se manifestar** por **meio de falhas observadas durante** a **execução do software**.

- As falhas podem ser resultado de:
  - **uma especificação errada ou falta de requisito,**
  - **o projeto pode conter defeitos ou**
  - **o código pode estar errado.**

Importância dos testes

- Investir em **testes** é uma **boa estratégia** para as empresas de desenvolvimento **diminuírem os custos diretos** (manutenção, suporte e retrabalho

- Contribui no **aumento da qualidade** dos produtos

- Melhora a **satisfação dos clientes**

Teste de Software

Do **ponto de vista psicológico**, o teste de software é uma atividade com um certo **viés destrutivo**, ao contrário de outras atividades do processo de software.

Mitos a serem eliminados

O testador é inimigo do desenvolvedor

A equipe de testes pode ser montada com os desenvolvedores menos qualificados

Quando o software estiver pronto deverá ser testado pela equipe de testes

Lei de Murphy

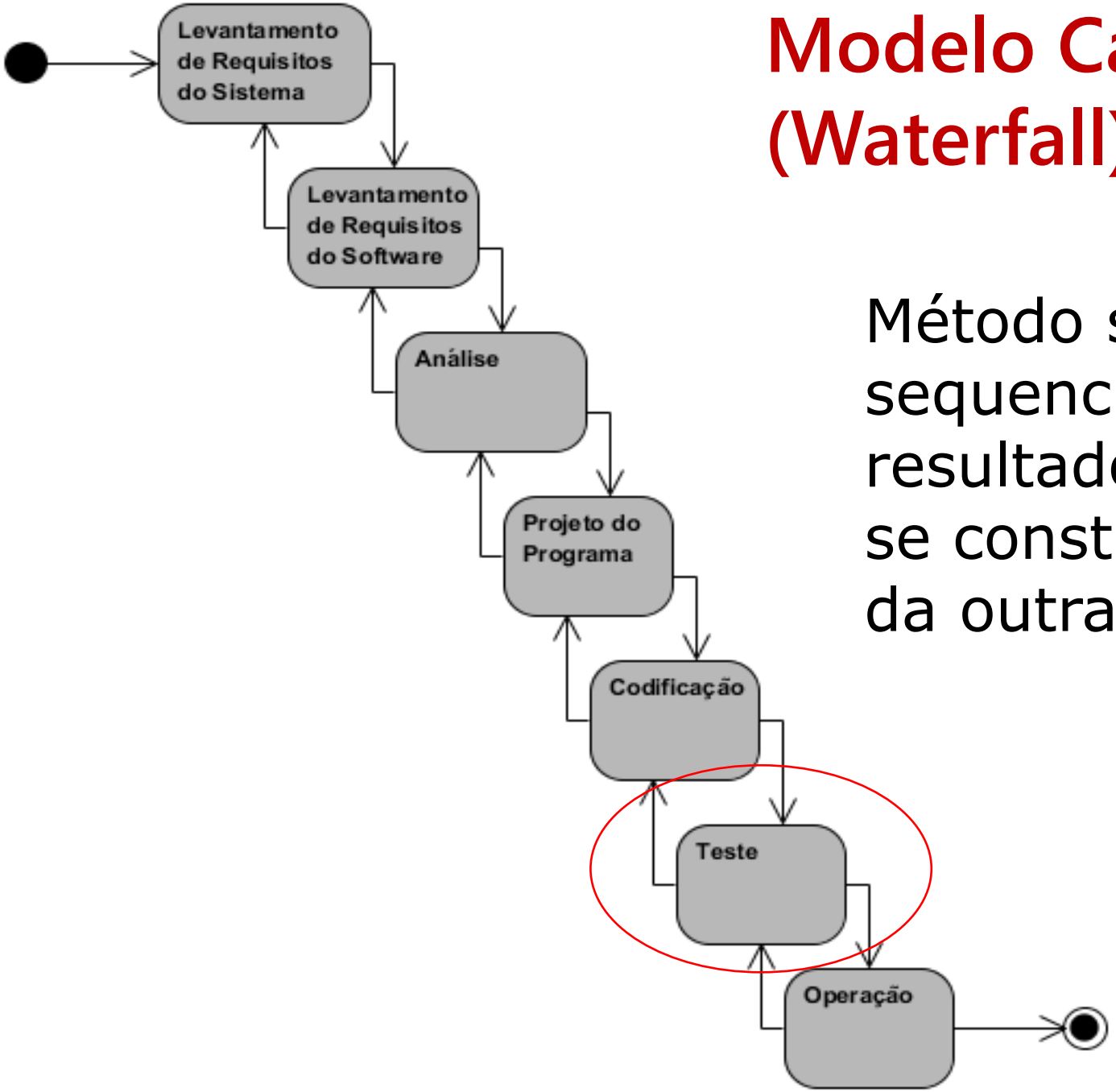
Se alguma coisa pode dar errado, dará.
E mais, dará errado da pior maneira, no pior momento e de modo que cause o maior dano possível

Se tudo parece ir indo muito bem é porque você não olhou direito

A natureza sempre está a favor da falha oculta

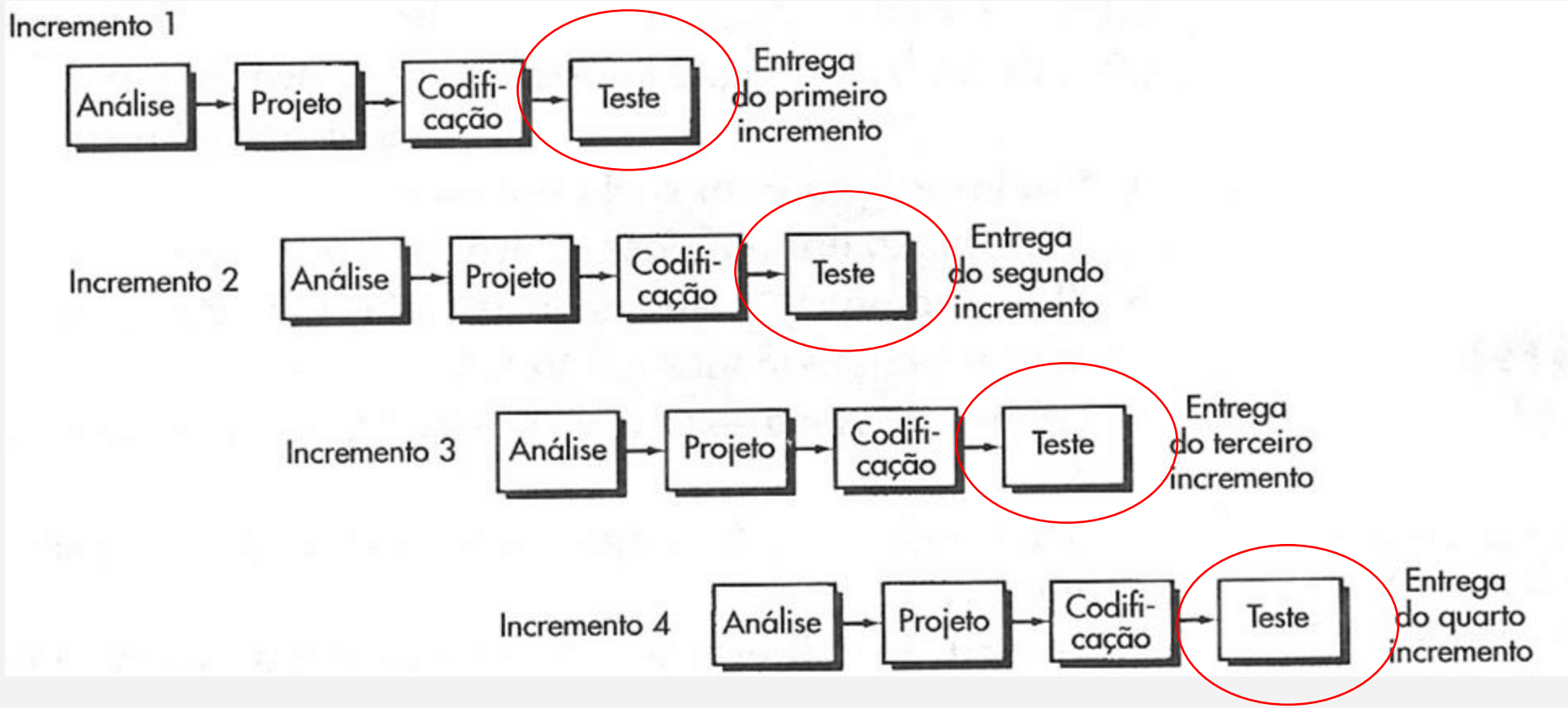
Você sempre acha algo no último lugar que procura

Modelo Cascata (Waterfall)



Método sistemático e sequencial, em que o resultado de uma fase se constitui na entrada da outra fase.

Modelo incremental e iterativo



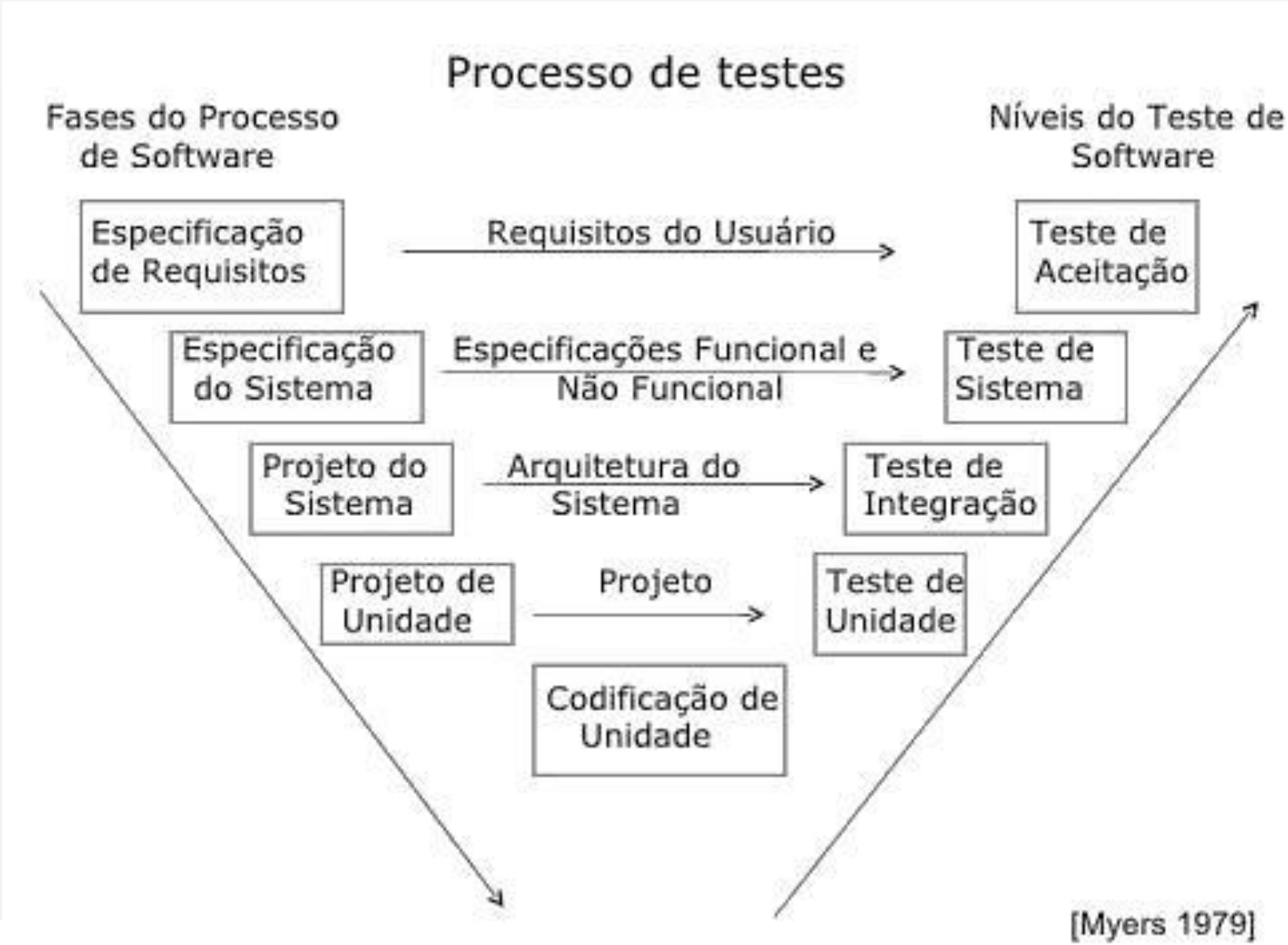
Testar no final de cada iteração

é o suficiente?

Desafio:

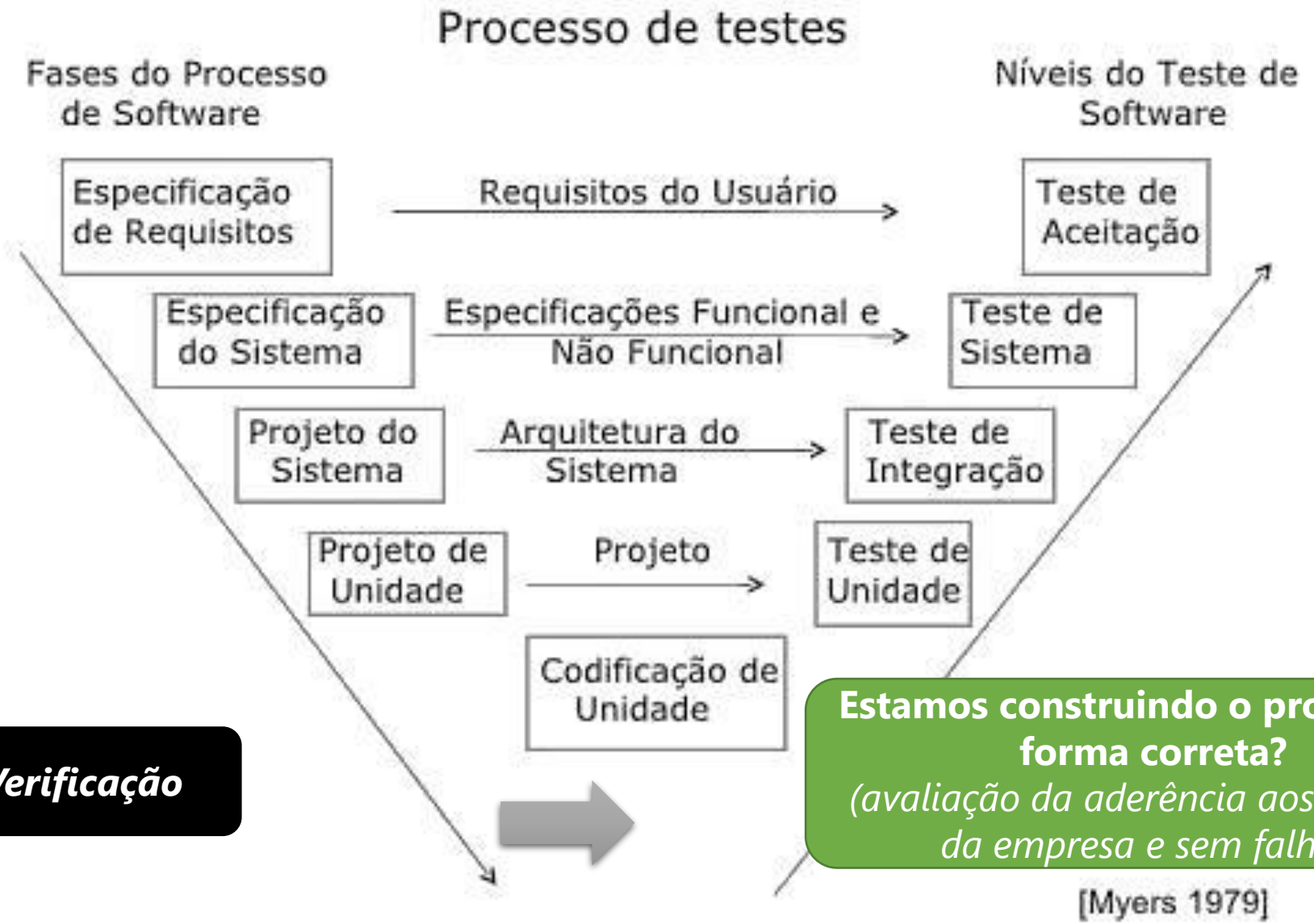
Como integrar o processo de teste ao longo de todo o ciclo de vida do sistema, não sendo apenas uma atividade a ser executada após o desenvolvimento?

Modelo em V


Processo de testes
Fases do Processo de Software
Níveis do Teste de Software
Especificação de Requisitos
Requisitos do Usuário
Teste de Aceitação
Especificação do Sistema
Especificações Funcional e Não Funcional
Teste de Sistema
Projeto do Sistema
Arquitetura do Sistema
Teste de Integração
Projeto de Unidade
Projeto
Teste de Unidade
Codificação de Unidade
[Myers 1979]

Validação

Estamos construindo o produto certo?
(avaliação do atendimento aos requisitos)

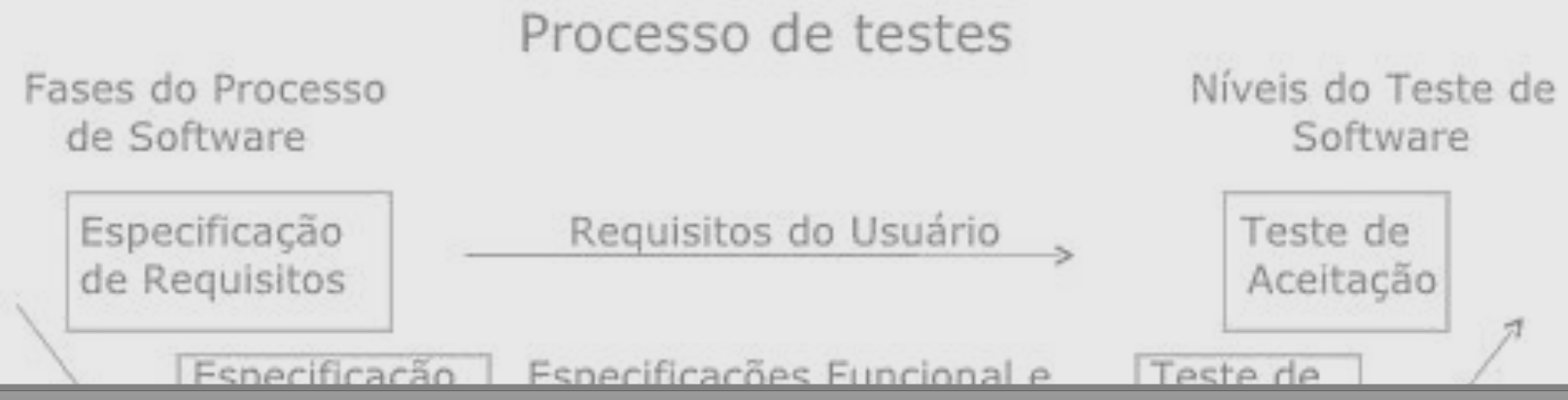


Verificação

Estamos construindo o produto de forma correta?
(avaliação da aderência aos padrões da empresa e sem falhas)

Validação

Estamos construindo o produto certo?
(avaliação do atendimento aos requisitos)



Teste de unidade e integração: perspectiva dos projetistas e desenvolvedores

Verificação

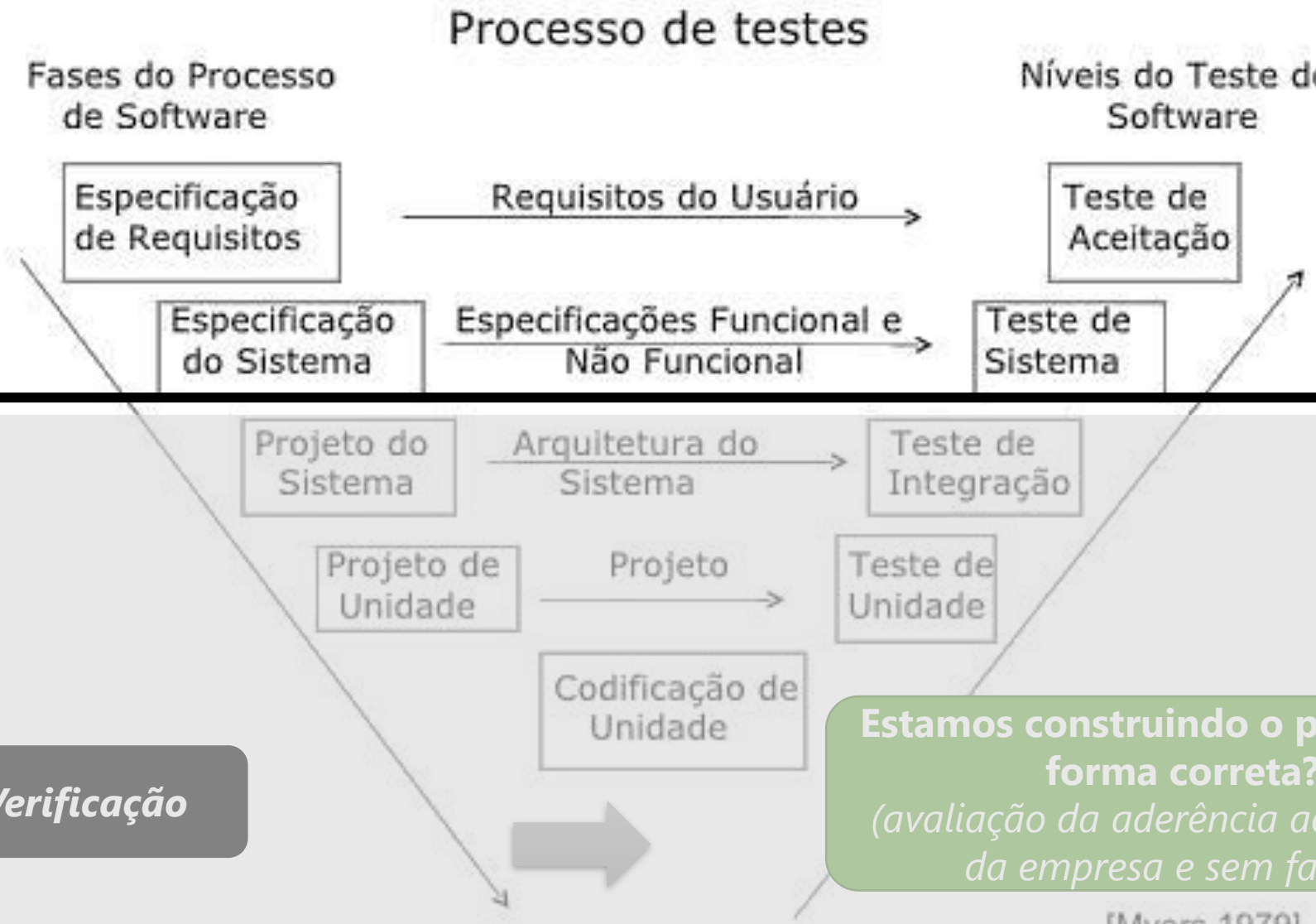
Estamos construindo o produto de forma correta?
(avaliação da aderência aos padrões da empresa e sem falhas)

[Myers 1979]

Testes de sistemas e aceitação: perspectiva do cliente e usuários

Validação

Estamos construindo o produto certo?
(avaliação do atendimento aos requisitos)

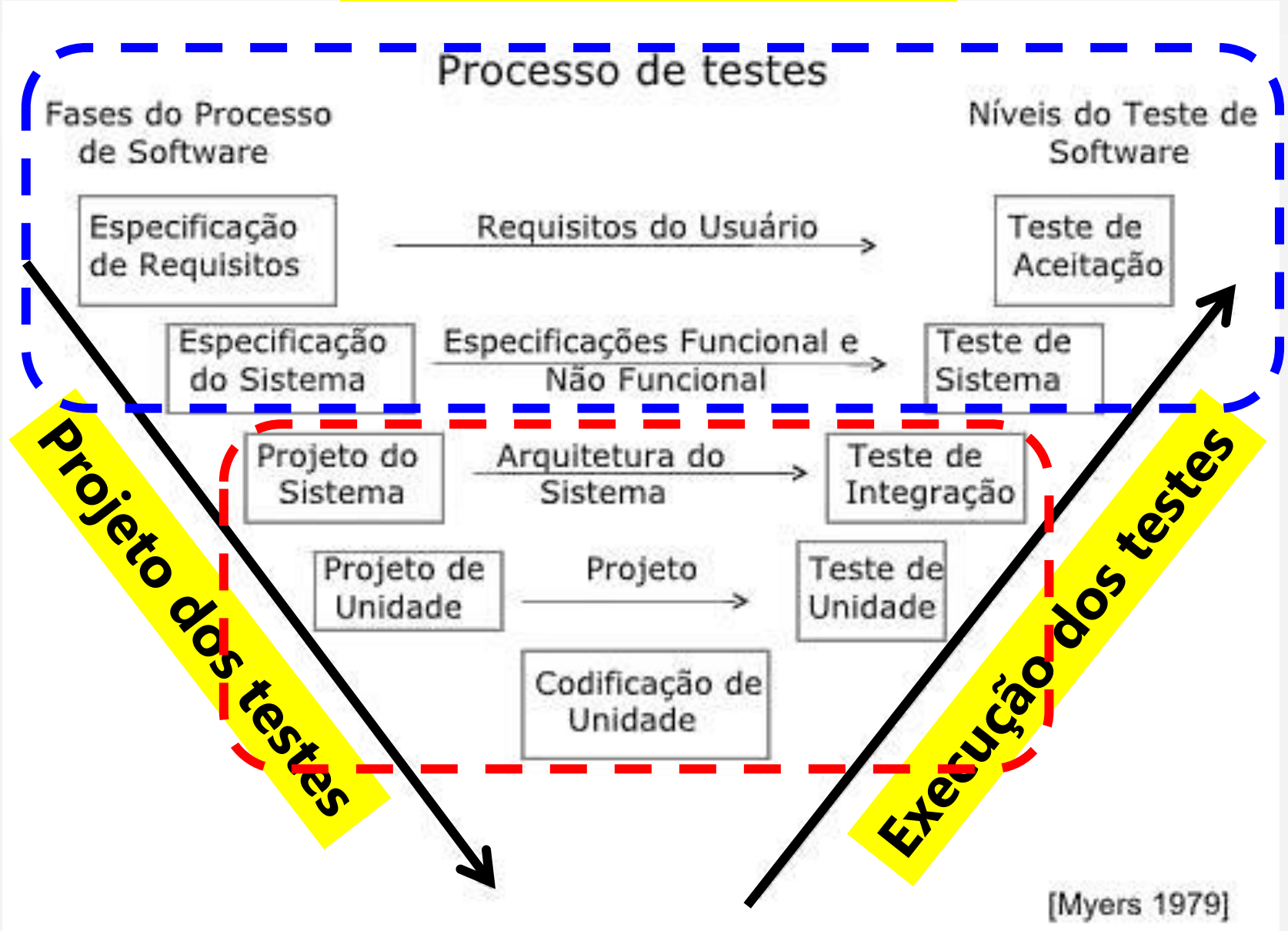


Verificação

Estamos construindo o produto de forma correta?
(avaliação da aderência aos padrões da empresa e sem falhas)

[Myers 1979]

PLANO DE TESTES

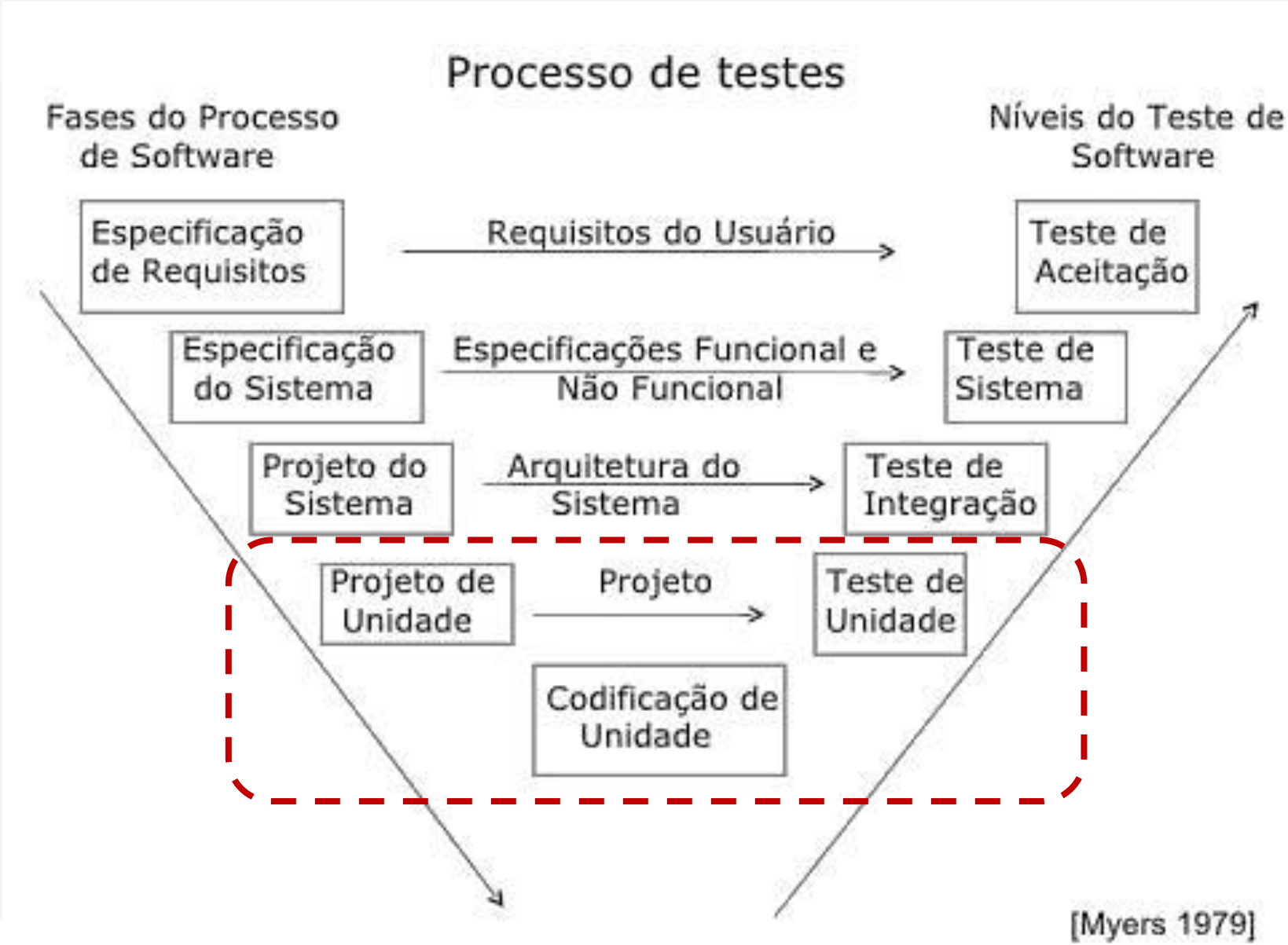


Modelo em V

A importância em estudar o Modelo em V: **associação dos testes em todas as fases do processo de desenvolvimento**, relacionado as **atividades que devem ser testadas** para **garantir a entrega de um produto de qualidade ao cliente.**

1. NÍVEIS DE TESTES
(Quando testar)

Modelo em V


Processo de testes
Fases do Processo de Software
Níveis do Teste de Software
Especificação de Requisitos
Requisitos do Usuário
Teste de Aceitação
Especificação do Sistema
Especificações Funcional e Não Funcional
Teste de Sistema
Projeto do Sistema
Arquitetura do Sistema
Teste de Integração
Projeto de Unidade
Projeto
Teste de Unidade
Codificação de Unidade
[Myers 1979]

Testes de Unidade (testes unitários)

Também conhecido como **testes unitários**

Objetivo: **explorar a menor unidade do projeto procurando** provocar falhas ocasionadas por **defeitos** de lógica e de implementação em cada módulo, separadamente

Alvo do teste: métodos dos objetos ou mesmo pequenos trechos de código

São aplicados de maneira individual a cada unidade do sistema

Cada unidade do sistema é verificada (testada) de forma isolada

Normalmente é realizado pelo próprio programador

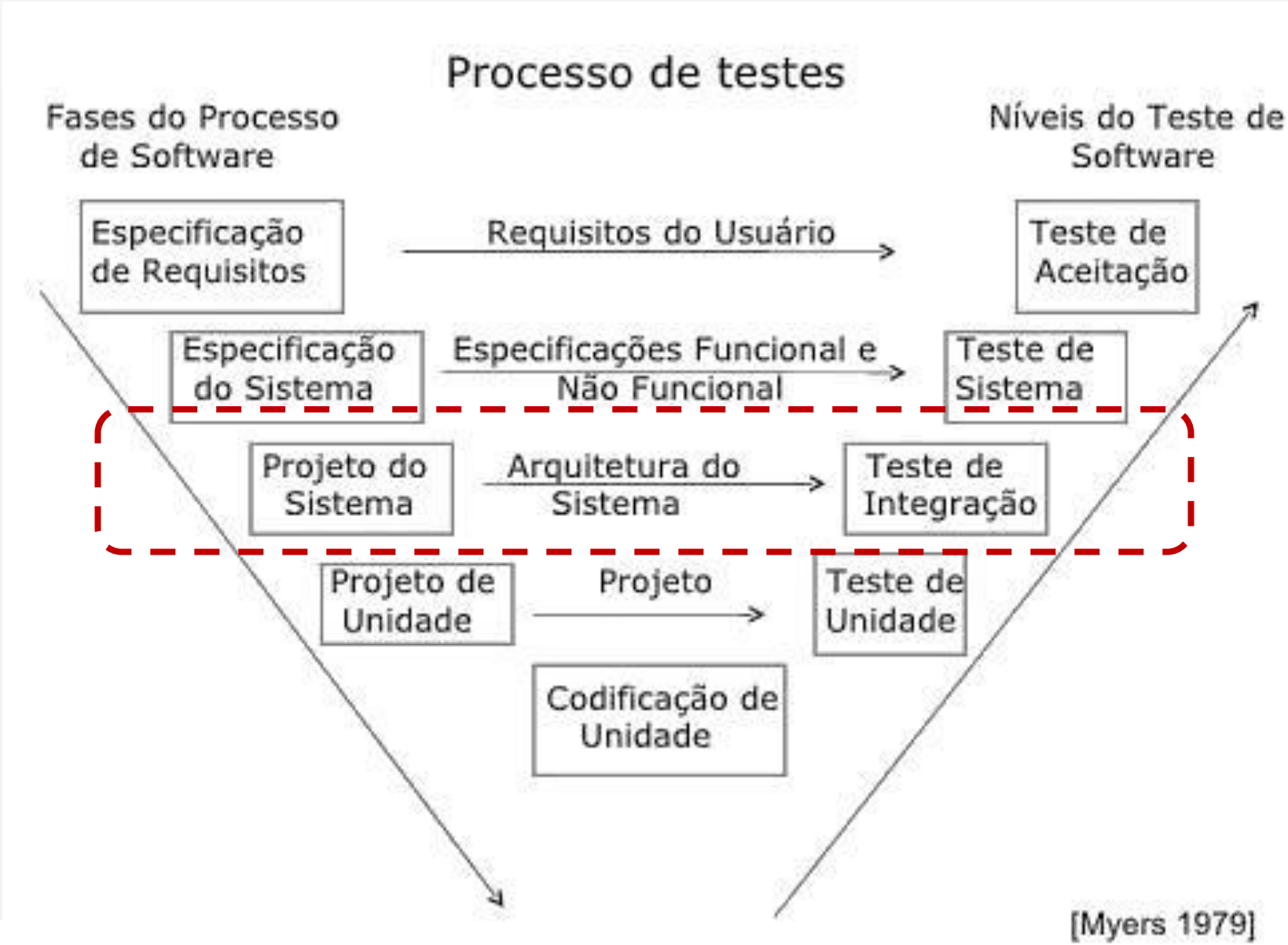
Testes de Unidade (testes unitários)

- **Testes manuais**
  - Necessidade casos de testes bem especificados, com definição de informação de objetivo do teste, ação a ser realizada, saída ou comportamento esperado
  - Realização de code review: inspeção de código de forma não automatizada
  - Code review é uma boa prática para auxiliar na refatoração de código, propriedade coletiva e gestão de conhecimento
  - Testado uma vez
- **Testes automatizados**
  - Executa uma rotina que irá testar o seu (programa que testa um programa)
  - Pode ser executado várias vezes
  - Arcabouço de testes automatizados (ferramenta que facilita em escrever e executar os testes unitários)
    (Exemplo de ferramentas: Python: PyTest, PHP: PHPUnit, Java: Junit, ...)

Testes de Unidade

Cuidar para não testar apenas o cenário feliz

Modelo em V


Processo de testes
Fases do Processo de Software
Níveis do Teste de Software
Especificação de Requisitos
Requisitos do Usuário
Teste de Aceitação
Especificação do Sistema
Especificações Funcional e Não Funcional
Teste de Sistema
Projeto do Sistema
Arquitetura do Sistema
Teste de Integração
Projeto de Unidade
Projeto
Teste de Unidade
Codificação de Unidade
[Myers 1979]

Teste de Integração

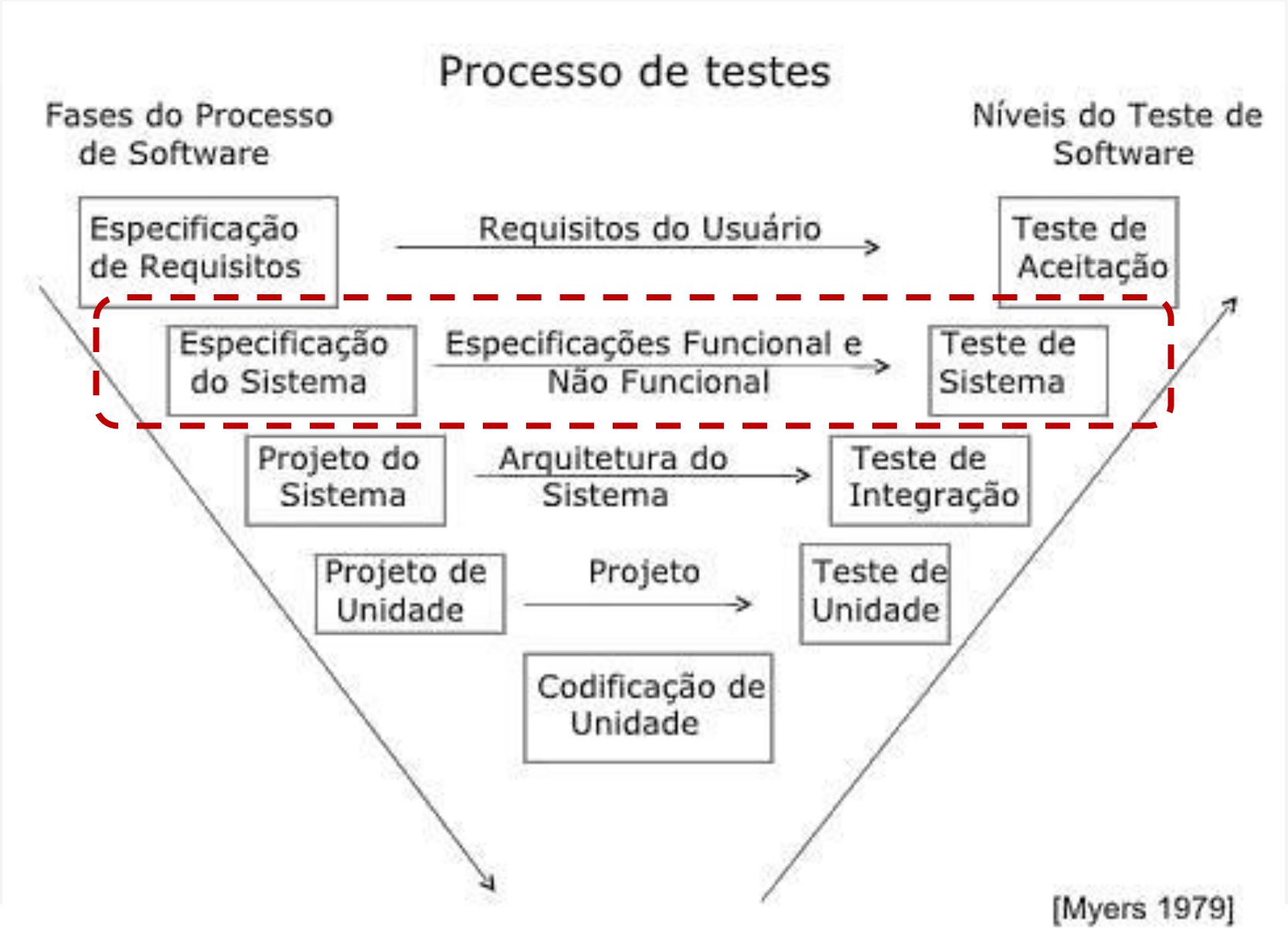
Objetivo é **apontar as falhas decorrentes da integração entre as unidades**.

É o processo de verificar a interação entre os componentes.

Visa provocar falhas associadas às interfaces entre os módulos.

Para que esta fase seja executada, os módulos já devem ter passado pelos testes unitários.

Modelo em V


Processo de testes
Fases do Processo de Software
Níveis do Teste de Software
Especificação de Requisitos
Requisitos do Usuário
Teste de Aceitação
Especificação do Sistema
Especificações Funcional e Não Funcional
Teste de Sistema
Projeto do Sistema
Arquitetura do Sistema
Teste de Integração
Projeto de Unidade
Projeto
Teste de Unidade
Codificação de Unidade
[Myers 1979]

Teste de Sistema

Verificação de cada uma das funcionalidades do sistema (**funcionais e não funcionais**).

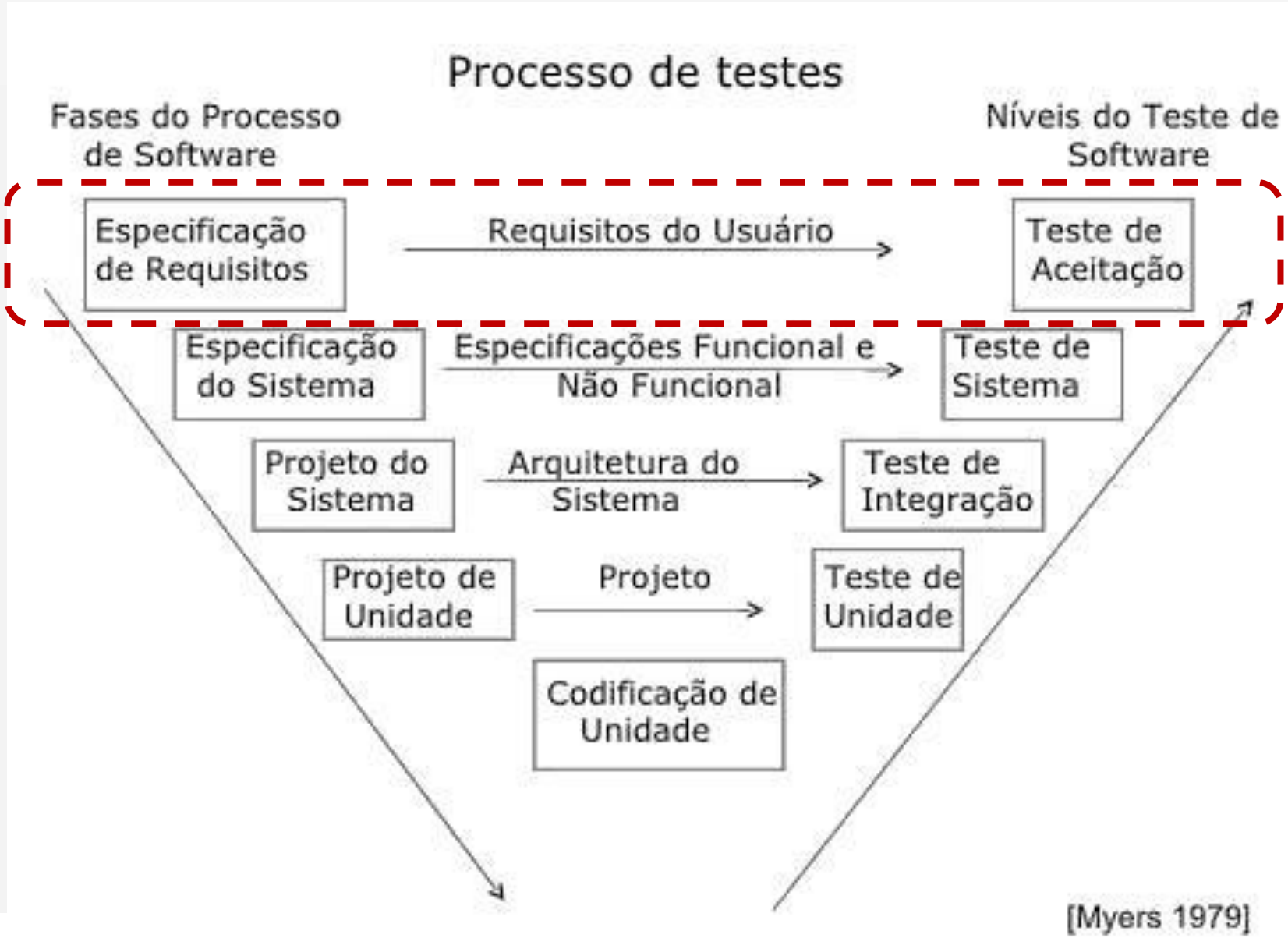
Avalia o software em busca de falhas por meio da utilização do mesmo, como se fosse um usuário final.

Dessa maneira:

- **os testes são executados nos mesmos ambientes**
- **com as mesmas condições**
- **com os mesmos dados de entrada que um usuário utilizaria no seu dia-a-dia**

Nesta etapa o software é testado por completo.

Modelo em V


Processo de testes
Fases do Processo de Software
Níveis do Teste de Software
Especificação de Requisitos
Requisitos do Usuário
Teste de Aceitação
Especificação do Sistema
Especificações Funcional e Não Funcional
Teste de Sistema
Projeto do Sistema
Arquitetura do Sistema
Teste de Integração
Projeto de Unidade
Projeto
Teste de Unidade
Codificação de Unidade
[Myers 1979]

Teste de Aceitação

O software é testado pelo usuário final

Objetivo demonstrar a conformidade com os requisitos definidos pelo usuário

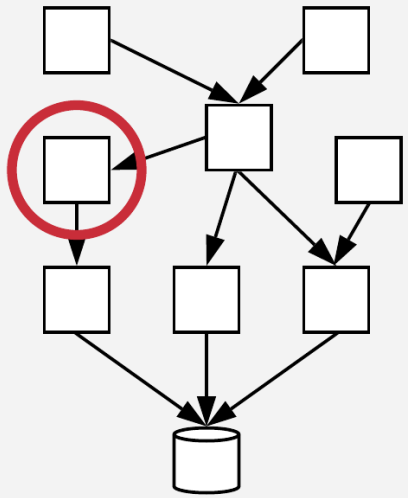
Testa as funcionalidades do sistema

Realizados por um restrito grupo de usuários finais do sistema

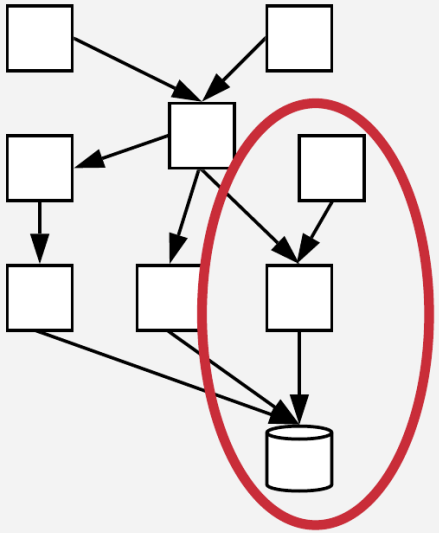
Simulam operações de rotina do sistema de modo a verificar se seu comportamento está de acordo com o solicitado.

Nesta fase o cliente confirma se todas as suas necessidades foram atendidas pelo sistema.

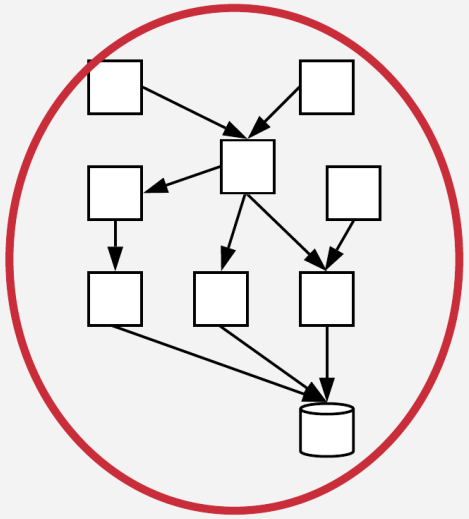
Tipos de testes



Testes de Unidade



Testes de Integração



Testes de Sistema

Testes de Aceitação

2. TÉCNICAS DE TESTES
(Como testar)

Técnicas de Teste

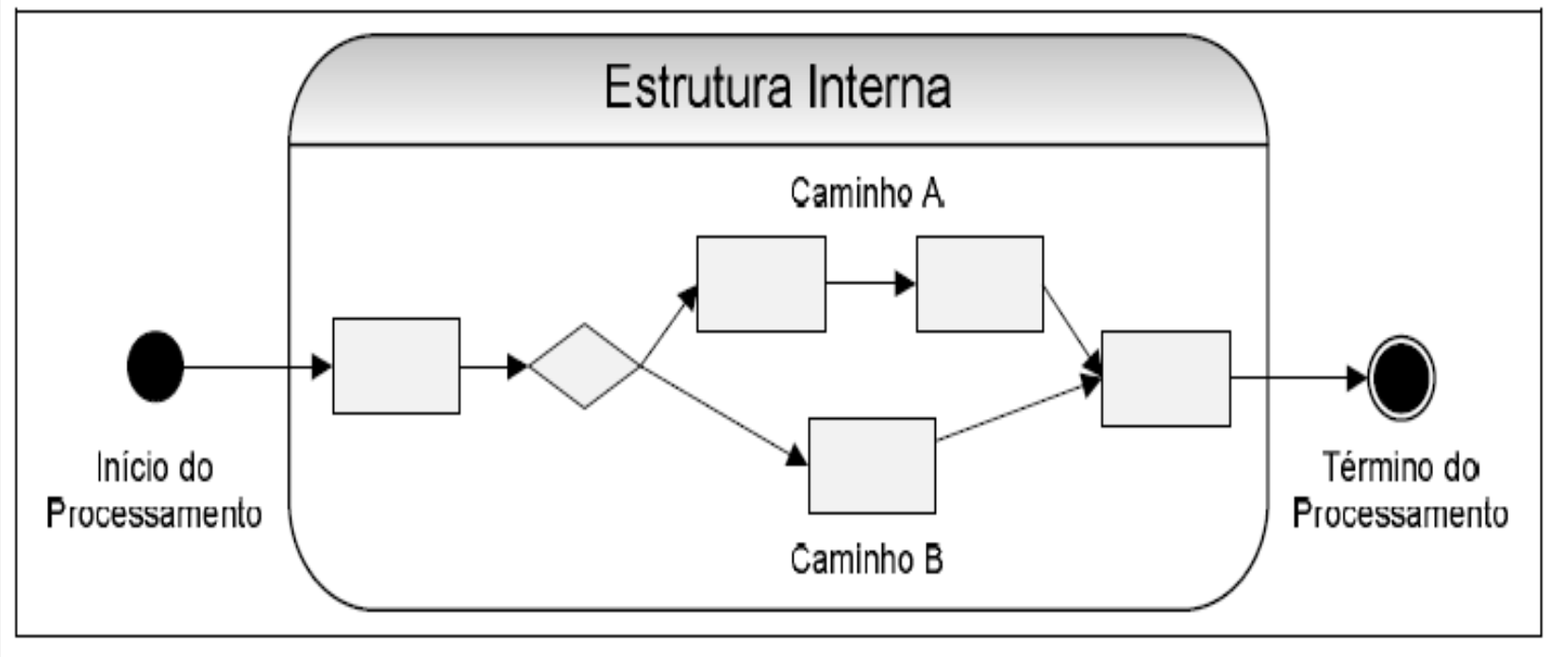
As técnicas de teste de software fornecem ao desenvolvedor uma abordagem sistemática de como fazer o teste, ou seja, um conjunto de métodos para ajudar a descobrir a maior quantidade de defeitos possível.

As técnicas de teste são classificadas de acordo com a origem das informações utilizadas para estabelecer os requisitos de teste.

As técnicas existentes são:

- Funcionais ou caixa preta
- Estruturais ou caixa branca

Técnicas de Teste - Caixa Branca



- Garante que todos os caminhos lógicos e loops foram percorridos pelo menos uma vez
- Exercita estruturas de dados e sua validade
- Testes unitário e de integração

Técnicas de Teste - Caixa Branca

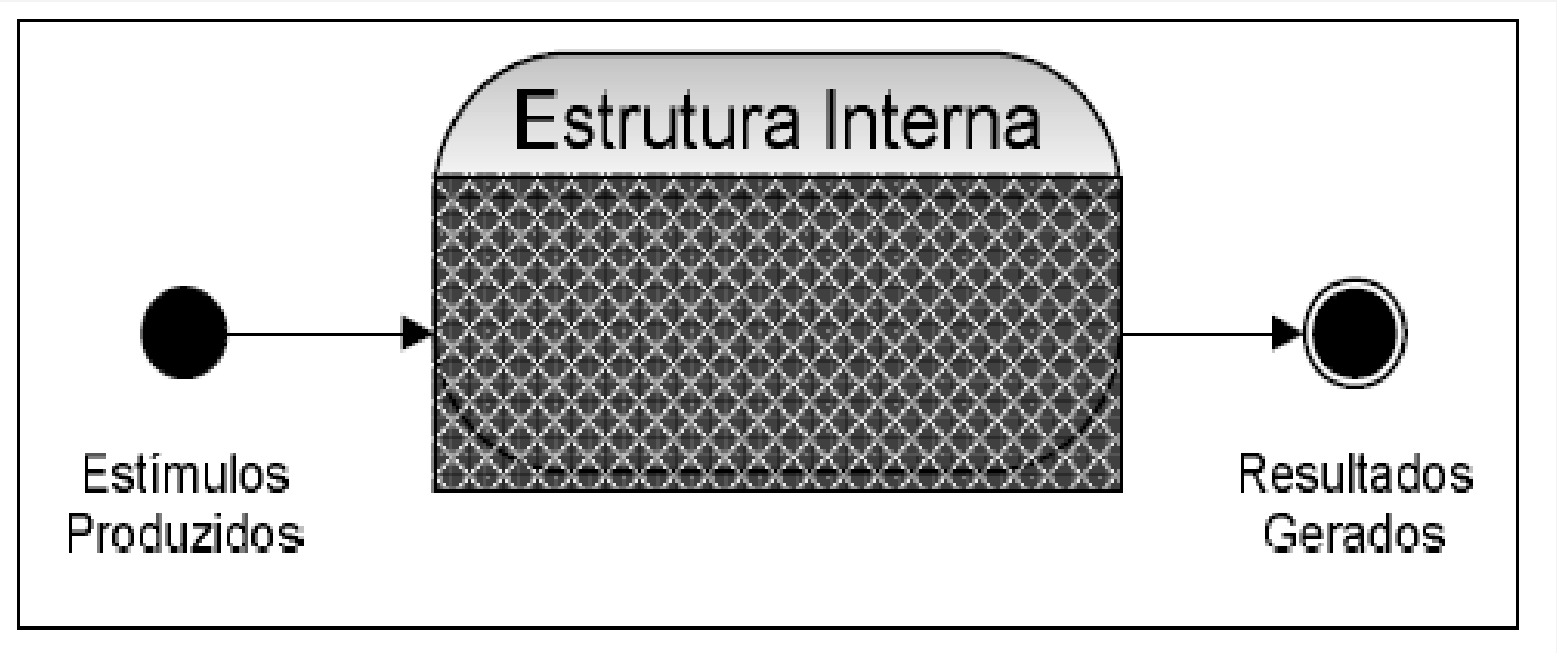
Avalia o comportamento interno do componente de software

Trabalha diretamente sobre o **código fonte** do componente de software para avaliar aspectos tais como: condições, fluxo de dados, ciclos ou caminhos lógicos

O testador tem acesso ao código fonte da aplicação

O objetivo é **testar todas as estruturas internas** do sistema, de forma a garantir que o software implementado **possui o comportamento desejado**.

Técnica de Teste - Caixa Preta



Técnica de Teste - Caixa Preta

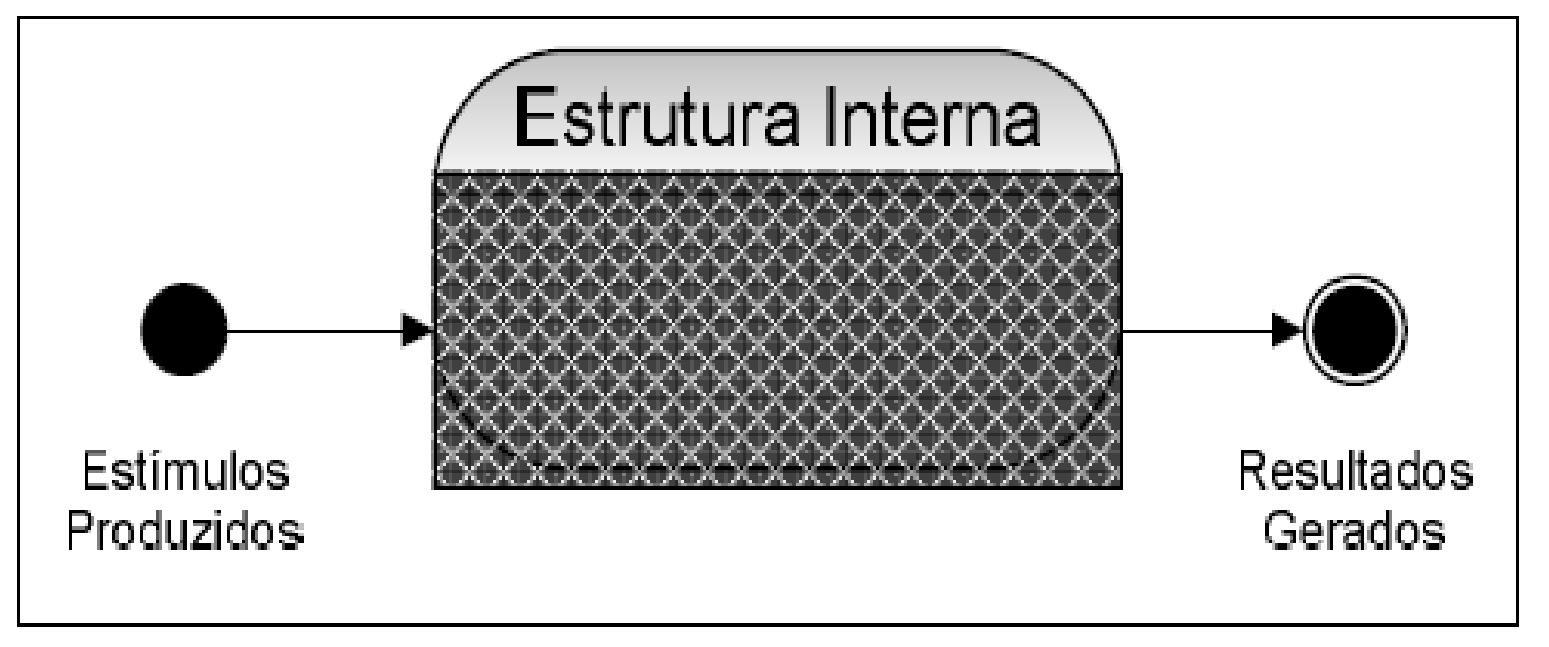
Objetivo testar o software numa perspectiva externa.

Dados de entrada são fornecidos, o teste é executado e o resultado obtido é comparado a um resultado esperado.

Haverá sucesso no teste se o resultado obtido for igual ao resultado esperado.

Não exige conhecimento da estrutura interna do desenvolvimento do produto

Técnica de Teste - Caixa Preta



- Testa os requisitos funcionais e não funcionais, funções incorretas ou faltantes
- Identifica erros de interface, performance e acesso as estruturas de dados
- Testes de sistema e de aceitação