Verificação, Validação & Teste de Software

Jean Phelipe de Oliveira Lima

jpdol.eng16@uea.edu.br

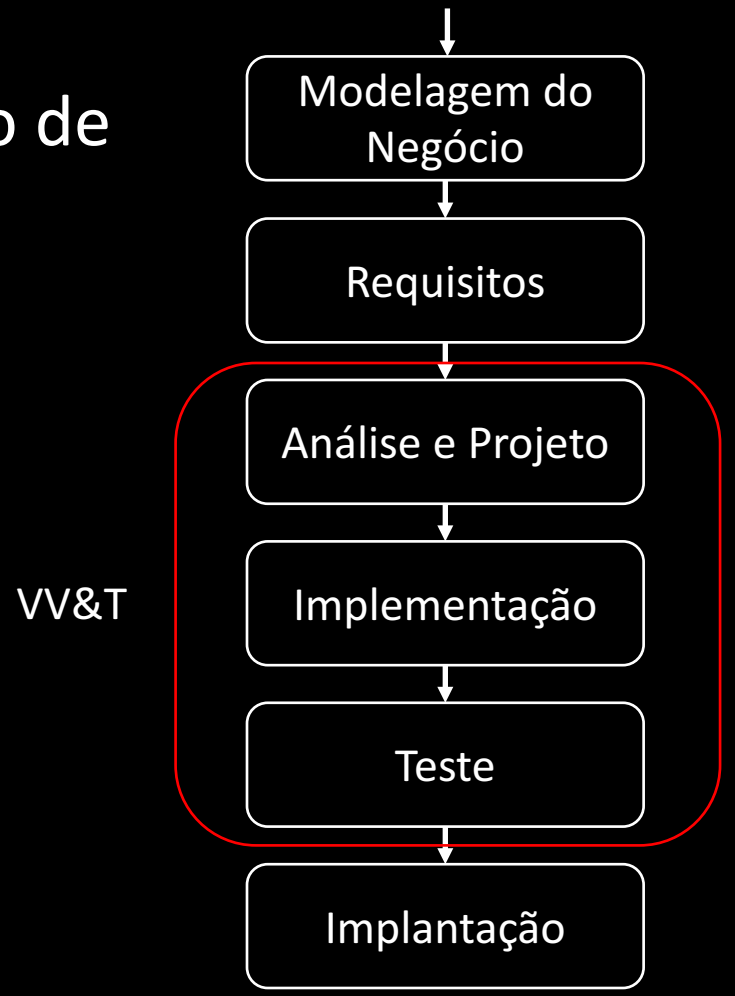
/jpdol

Engenharia de Software (Revisão)

- Fluxo de Referência para o desenvolvimento de Software.

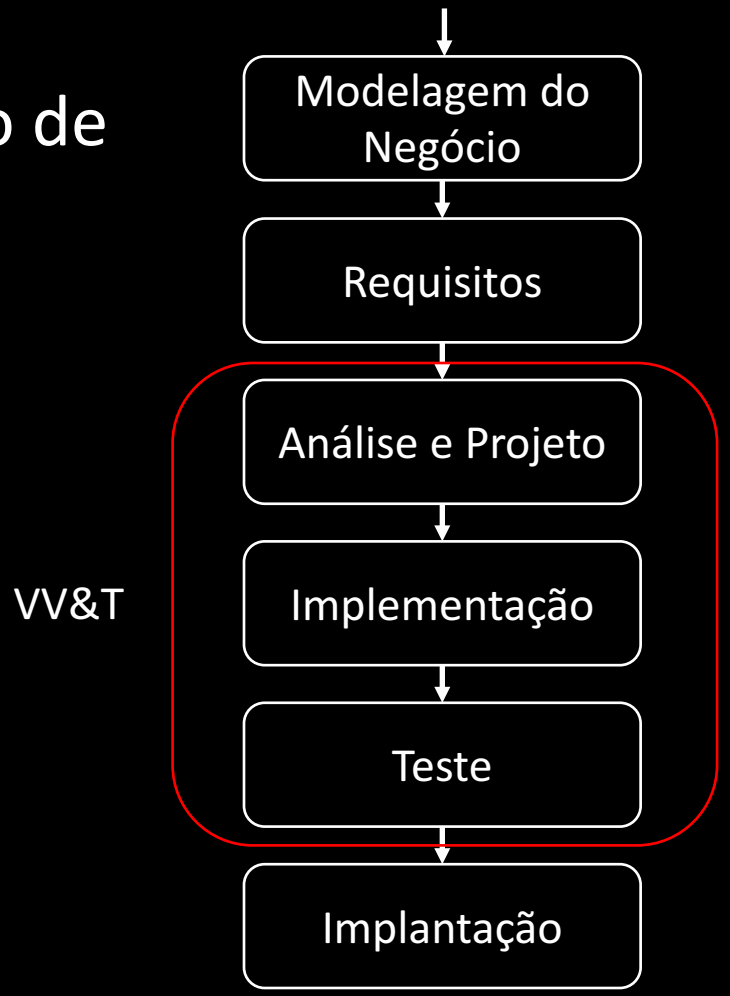
Engenharia de Software (Revisão)

- Fluxo de Referência para o desenvolvimento de Software.



Engenharia de Software (Revisão)

- Fluxo de Referência para o desenvolvimento de Software.



Validação, Verificação e Teste de Software

- Desafio: Construção de software é uma tarefa complexa.

Validação, Verificação e Teste de Software

- Desafio: Construção de software é uma tarefa complexa.

- Principal Problema: Solução Desenvolvida != Solução Esperada

Validação, Verificação e Teste de Software

- Desafio: Construção de software é uma tarefa complexa.
- Principal Problema: Solução Desenvolvida != Solução Esperada.
- Principal Motivo: Erro Humano.
    - Desenvolver Software depende principalmente da habilidade, da interpretação e da execução da equipe.
    - Erros surgem mesmo com a utilização de métodos e ferramentas da Engenharia de Software.

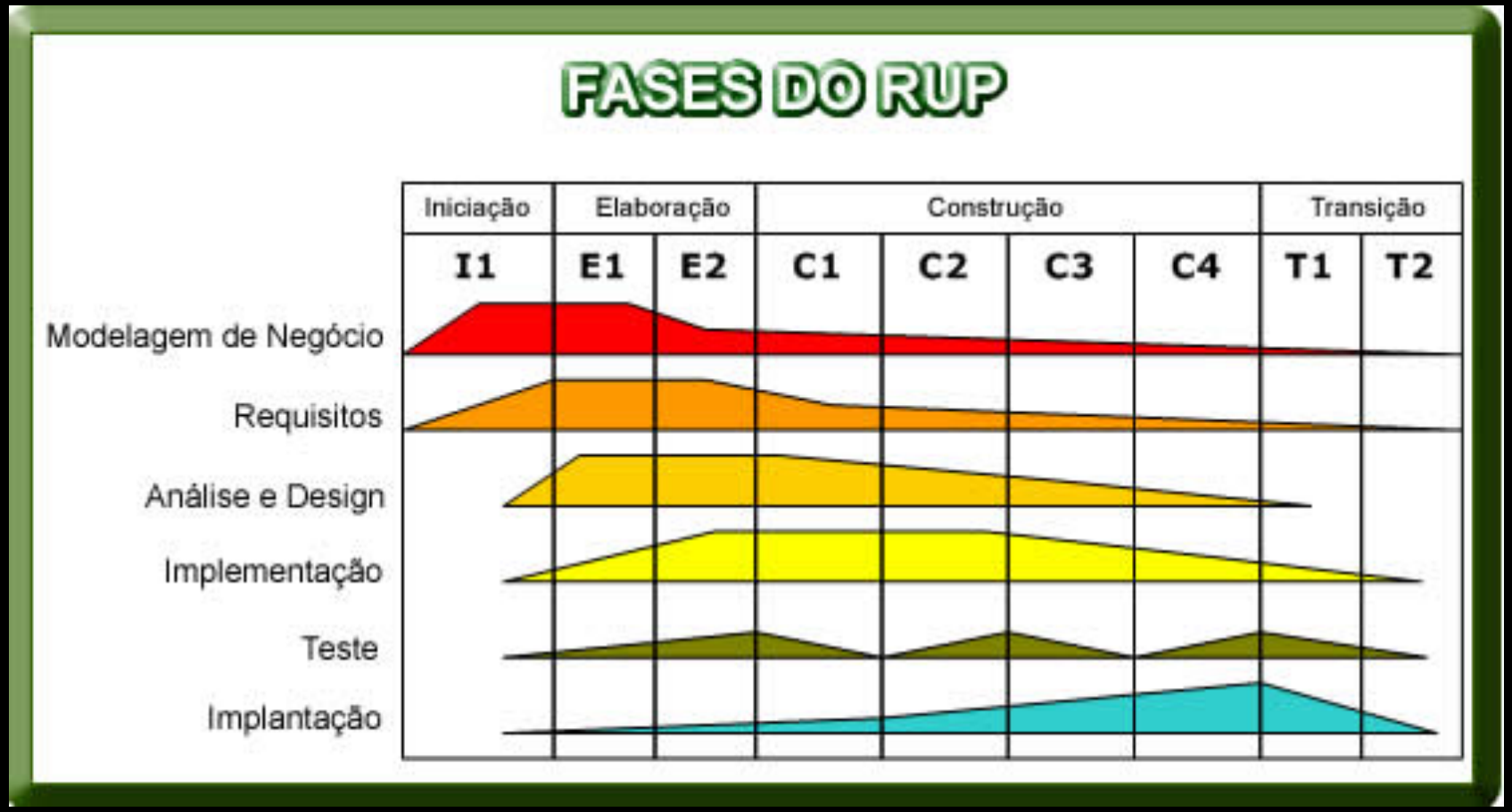
Validação, Verificação e Teste de Software

- Solução: Métodos avaliação que buscam garantir que o modo como o software vem sendo construído e que o produto esteja em conformidade com o esperado.
  - O conjunto de Atividades que realizam essa função é denominado VV&T.
- Resumo: VV&T servem para que qualquer tipo de erro seja encontrado antes do software ser liberado para utilização.

”Teste consiste em executar o programa com intenção de encontrar erros” – Meyers

”Testes nunca podem demonstrar a ausência de erros em softwares, apenas a presença” – Dijkstra

Validação, Verificação e Teste de Software



Validação, Verificação e Teste de Software

- Atividades VV&T:
  - Estáticas: Não requerem execução ou mesmo existência de um programa ou modelo executável para serem conduzidas.
  - Dinâmicas: Se baseiam na execução de um programa ou de um modelo.

Validação, Verificação e Teste de Software

- Validação: Assegurar que o produto final corresponda aos requisitos do usuário. (Estática)

”Estamos construindo o produto certo?”

- Verificação: Assegurar consistência, completitude e corretude do produto em cada fase e entre fases consecutivas do ciclo de vida do software. (Estática e Dinâmica)

”Estamos construindo o produto certo?”

- Teste: Examina o comportamento do produto por meio de sua execução. (Dinâmica)

”O software está funcionando como deveria?”

Terminologia

- A literatura tradicional estabelece significados específicos para termos no universo de teste de software, como: falha, defeito, erro, engano.

Terminologia

- **Defeito** *(fault)*: passo, processo ou definição de dados incorretos;
- **Engano** *(mistake)*: ação humana que produz um defeito.
- OBS: esses dois conceitos são estáticos, pois estão associados a um determinado programa ou modelo e não dependem de uma execução particular.

Terminologia

- O **estado** *(state)* de um programa ou, mais precisamente, da execução de um programa em determinado instante, é dado pelo valor da memória (ou das variáveis do programa) e do apontador de instruções.
- A existência de um defeito pode ocasionar um **erro** *(error)* durante uma execução do programa, que se caracteriza por um estado inconsistente ou inesperado.
- Esse estado pode levar a uma **falha** *(failure)*, ou seja, pode fazer com que o resultado produzido pela execução seja diferente do resultado esperado.

Terminologia

- O **domínio de entrada** de um programa **P**, denotado por **D(P)**, é o conjunto de todos os possíveis valores que podem ser utilizados para executar **P**.

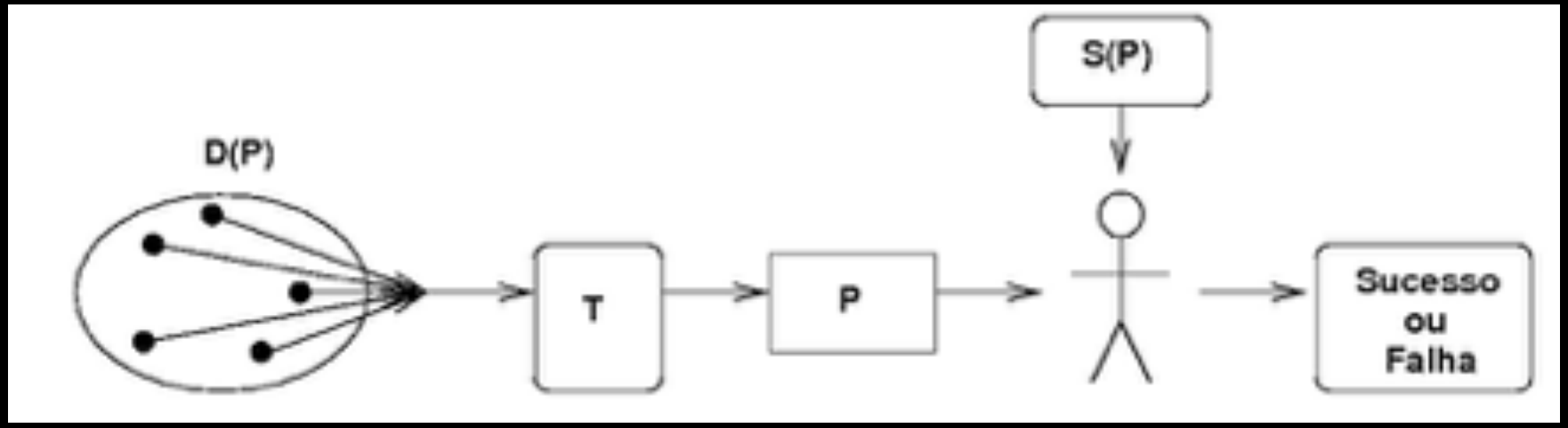
- O **domínio de saída** de **P** é o conjunto de possíveis resultados produzidos pelo programa.

Terminologia

- Um **dado de teste** para um programa **P** é um elemento do domínio de entrada de **P**.
- Um **caso de teste** é uma 2-tupla formada por um **dado de teste** e seu respectivo resultado esperado.
- O **conjunto de teste** ou **conjunto de casos de teste** é um conjunto formado por todos os **casos de teste**.

Terminologia

- Cenário típico da atividade de teste:



D(P) → Domínio de Entrada de P
T → Subconjunto de D(P)
P → Programa
S(P) → Especificação de Software

Fases da Atividade de Teste

- Teste de Unidade: Tem como foco menores unidades de um programa
  - Ex: Funções, Procedimentos, métodos ou classes.

- Teste de Integração: deve ser realizado após serem testadas as unidades individualmente, a ênfase é dada na construção da estrutura do sistema.

- Teste de Sistemas: realizado quando se tem o sistema completo. O objetivo é verificar se as funcionalidades especificadas nos documentos de requisitos estão todas corretamente implementadas.
    - Aspectos de correção, completitude e coerência são explorados, além dos requisitos não funcionais.

Fases da Atividade de Teste

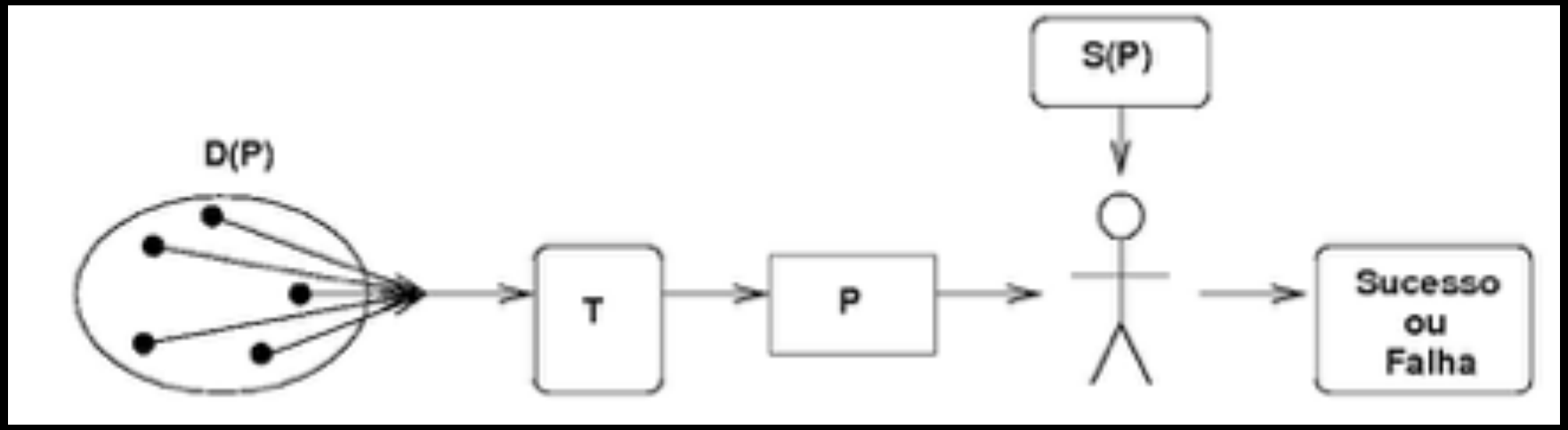
- Teste de Regressão: Realizado após a implantação do sistema, durante a fase de manutenção. é necessário, após a manutenção, realizar testes que mostrem que as modificações efetuadas estão corretas, ou seja, que os novos requisitos implementados (se for esse o caso) funcionam como o esperado e que os requisitos anteriormente testados continuam válidos.

Fases da Atividade de Teste

- Independente da fase de teste, algumas etapas são bem definidas para a execução da atividade:

1. Planejamento;
2. Projeto de Casos de Teste;
3. Execução;
4. Análise.

Técnicas e Critérios de Teste



- Idealmente, deveríamos ter T = D(P) para garantir a inexistência de erros, o que é infactível.
- Dessa forma, é importante encontrar uma forma de utilizar sabiamente um subconjunto de D(P).

Técnicas e Critérios de Teste

- A ideia central para isso é a de subdomínios de teste.
- Um subdomínio de D(P) é um subconjunto do domínio de entrada que contém dados de teste ”semelhantes”.

Técnicas e Critérios de Teste

- Métodos:
  - **Teste aleatório**: um grande número de casos de teste é selecionado aleatoriamente, de modo que, probabilisticamente, se tenha uma boa chance de que todos os subdomínios estejam representados no conjunto de teste T.
  - **Teste de Partição** (ou Teste de Subdomínios): procura-se estabelecer quais são os subdomínios a serem utilizados e, então, selecionam-se os casos de teste em cada subdomínio.
    - **Questão:** como identificar os subdomínios para, então, fazer a seleção de casos de teste?
    - **Solução:** Definição de requisitos de testes baseados em regras (ou **critérios de teste**).

Técnicas e Critérios de Teste

- Tipos de Critérios de Teste:
  - Funcionais;
  - Estruturais;
  - Baseados em Defeitos.

- OBS: Um conjunto de teste que satisfaz todos os requisitos de critério de teste **C**, ou seja, que tem pelo menos um caso de teste para cada subdomínio determinado por **C**, é dito **C-adequado.**

Características e Limitações

- É impossível mostrar que um programa está correto por meio de teste.
- Com a escolha de um subconjunto de dados de teste sempre corre-se o risco de não incluir um caso que revele a presença de um defeito.
- Objetivo da atividade de teste não é mostrar que um programa está correto, mas sim mostrar a presença de erros caso existam.
- Quando um software é testado com métodos embasados cientificamente, agrega-se **confiança** ao sistema.

Características e Limitações

- Limitações:
  - É indecidível se uma determinada sequência de comandos (um caminho) de um programa é executável ou não.
  - É indecidível se duas rotinas ou dois caminhos computam a mesma função.

Características e Limitações

- Não existe, a princípio, nenhuma restrição sobre o tipo de critérios de teste que pode ser definido e utilizado.
- A fim de se obter um nível mínimo de qualidade para os conjuntos adequados a um critério C para um programa P , deve-se requerer que:
    - exista um conjunto C-adequado que seja finito e, de preferência, de baixa cardinalidade;
    - do ponto de vista do fluxo de execução, um conjunto que seja C-adequado execute cada comando de P pelo menos uma vez;
    - do ponto de vista da utilização das variáveis, cada atribuição de valor a uma variável tenha seu valor verificado por um caso de teste que execute o trecho do programa que vai do ponto em que a variável recebe esse valor até um ponto do programa em que esse valor é utilizado.

Verificação, Validação & Teste de Software

Jean Phelipe de Oliveira Lima

jpdol.eng16@uea.edu.br

/jpdol