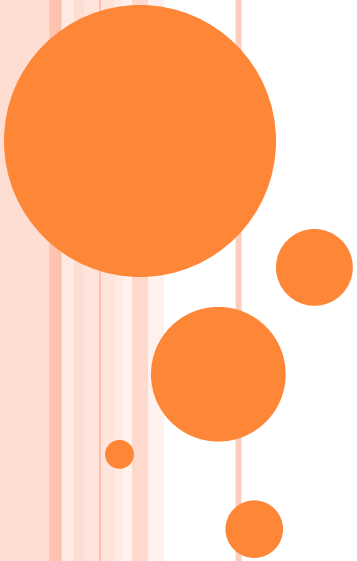
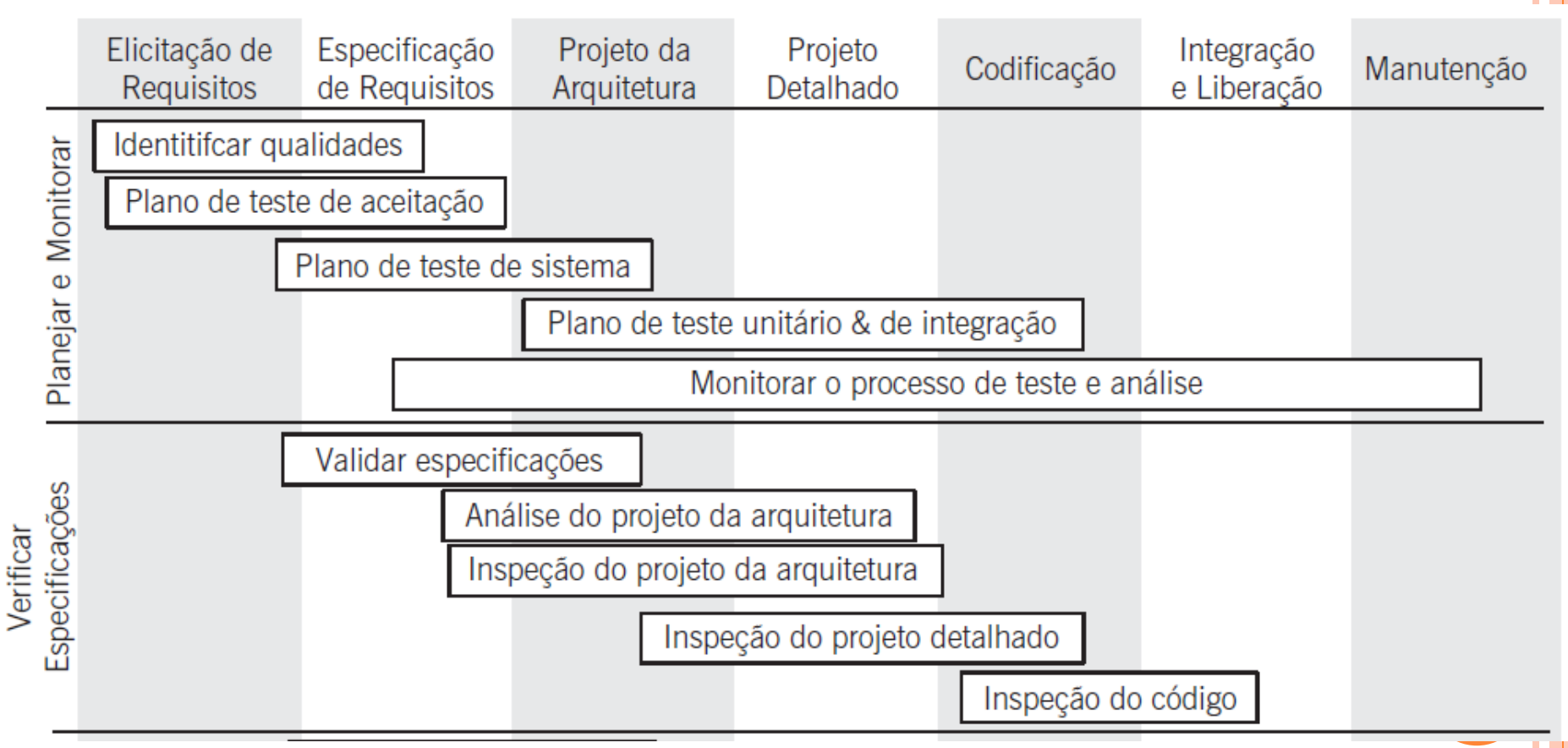


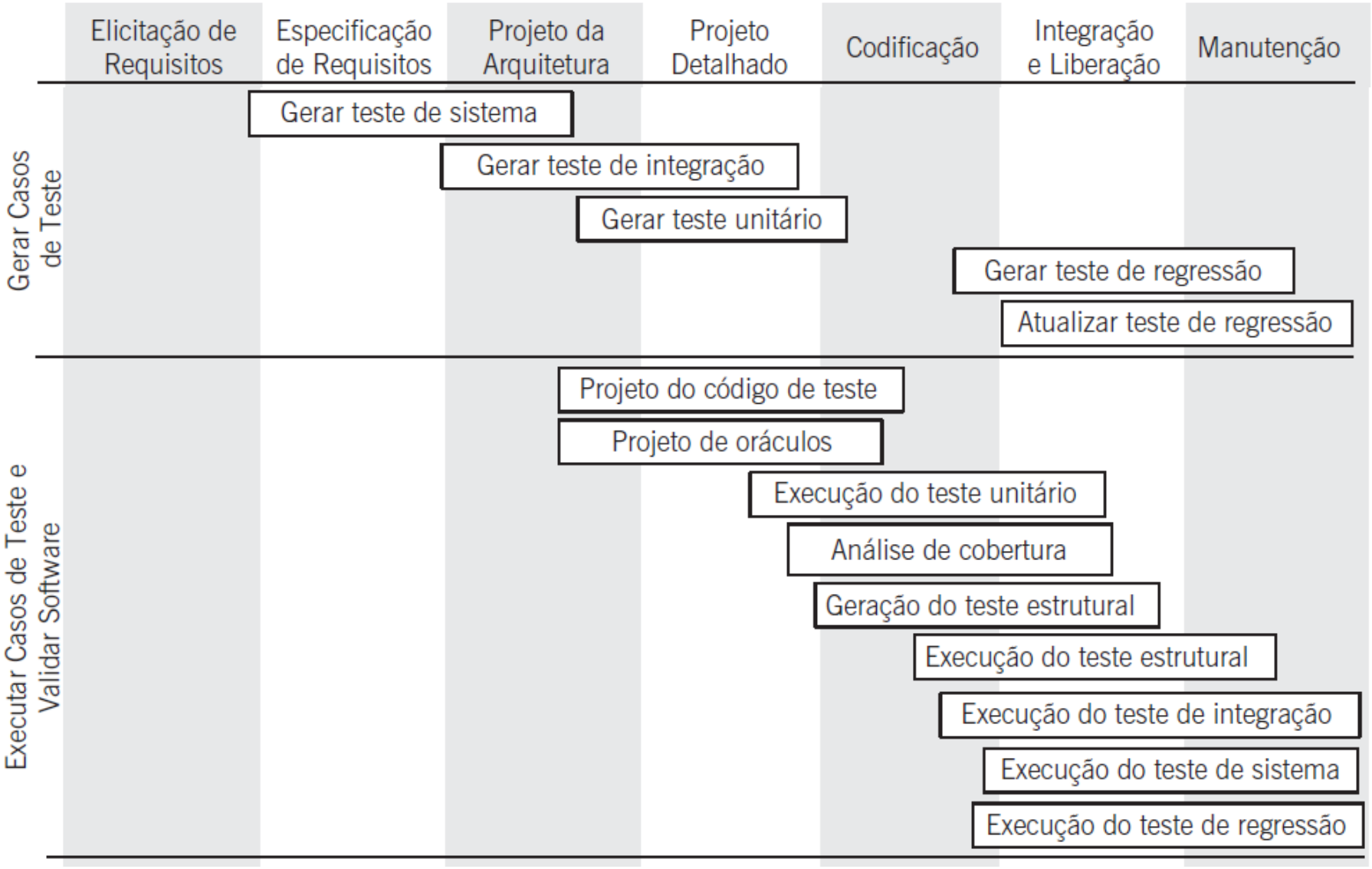
Processo de Teste

Paulyne Jucá (paulyne@ufc.br)

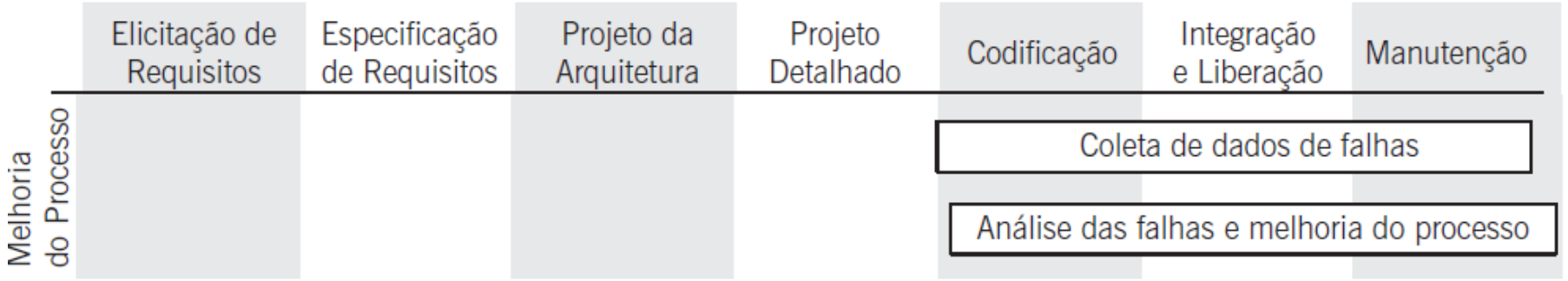
PRINCIPAIS ATIVIDADES DE TESTE E ANÁLISE AO LONGO DO CICLO DE VIDA DO SOFTWARE [1]



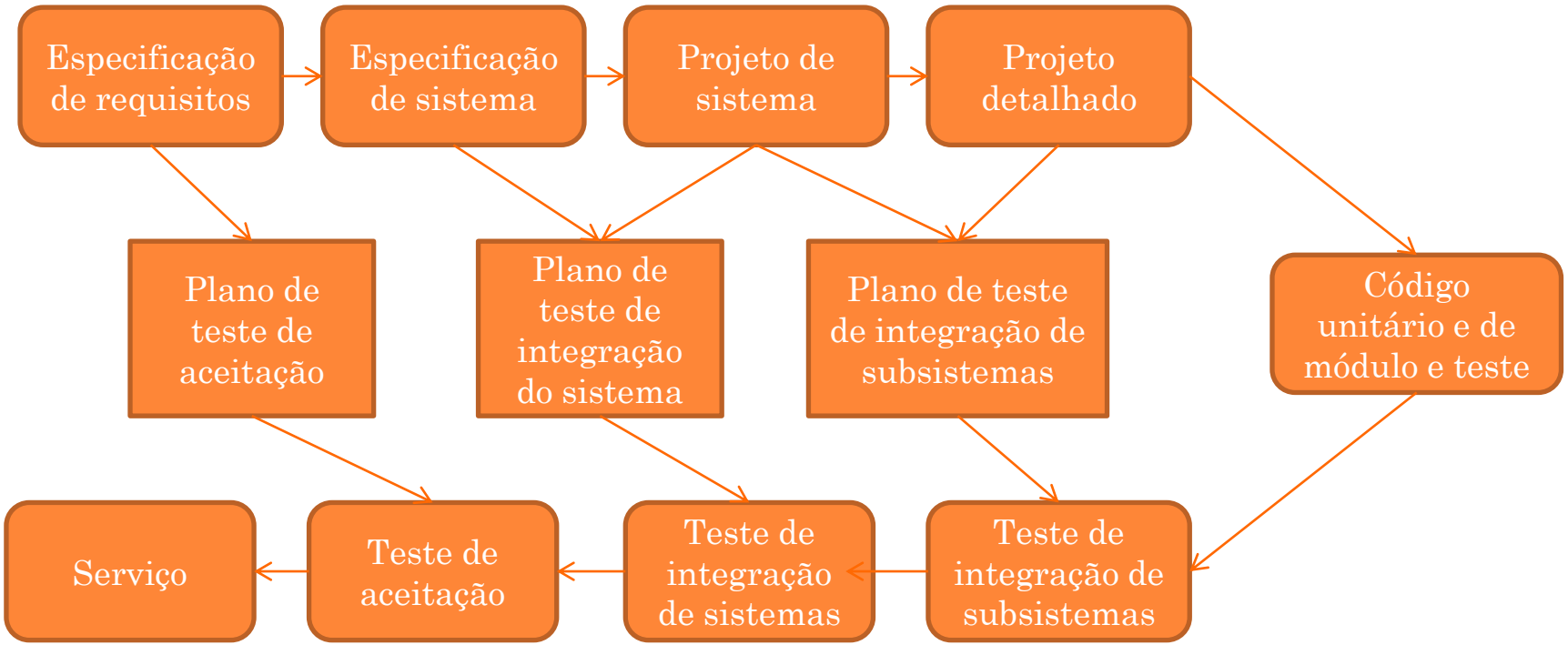
PRINCIPAIS ATIVIDADES DE TESTE E ANÁLISE AO LONGO DO CICLO DE VIDA DO SOFTWARE [1]



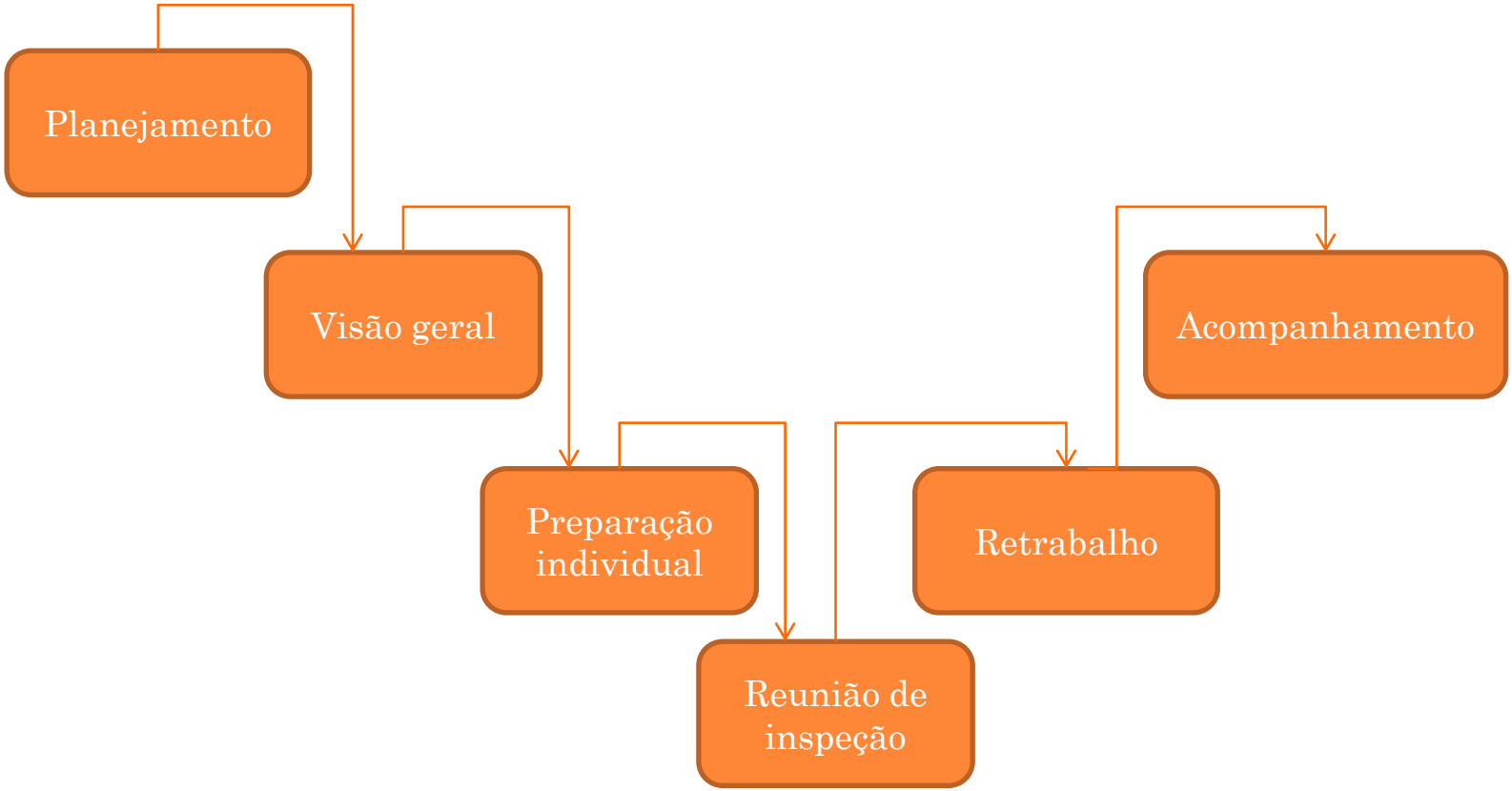
PRINCIPAIS ATIVIDADES DE TESTE E ANÁLISE AO LONGO DO CICLO DE VIDA DO SOFTWARE [1]



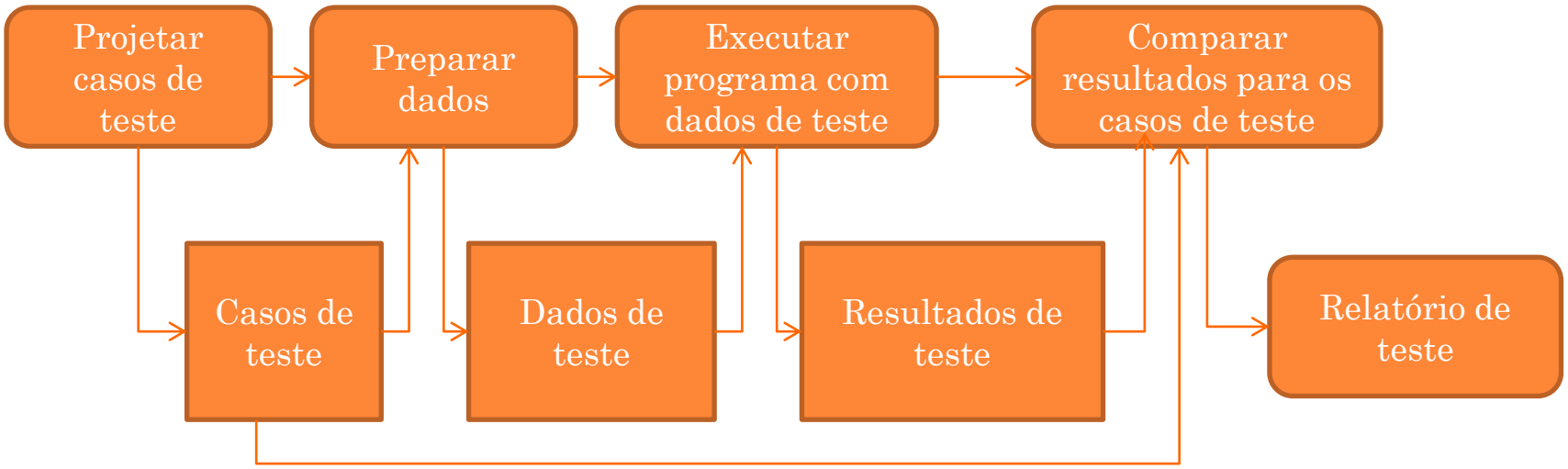
Plano de Teste [2]



Processo de Inspeção [2]



Processo de teste [2]



Quando saber se o software está pronto?

- Teste e análise têm o objetivo principal de revelar falhas para que estas sejam removidas
  - Tantas falhas quanto possível
- Encontrar todas é muito custoso e geralmente não é economicamente viável
  - Ex:
  - class Trivial {
    - static int sum (int a, int b) {return a+b;}
  - }

    - Inteiro em 32 bits 2^{32} possibilidades para cada inteiro
    - Possíveis entradas: 2^{32} x 2^{32}
    - Cada teste levar um nanossegundo: 10^{-9}
    - O teste completo levaria 10^{12} segundos que representa cerca de 30.000 anos

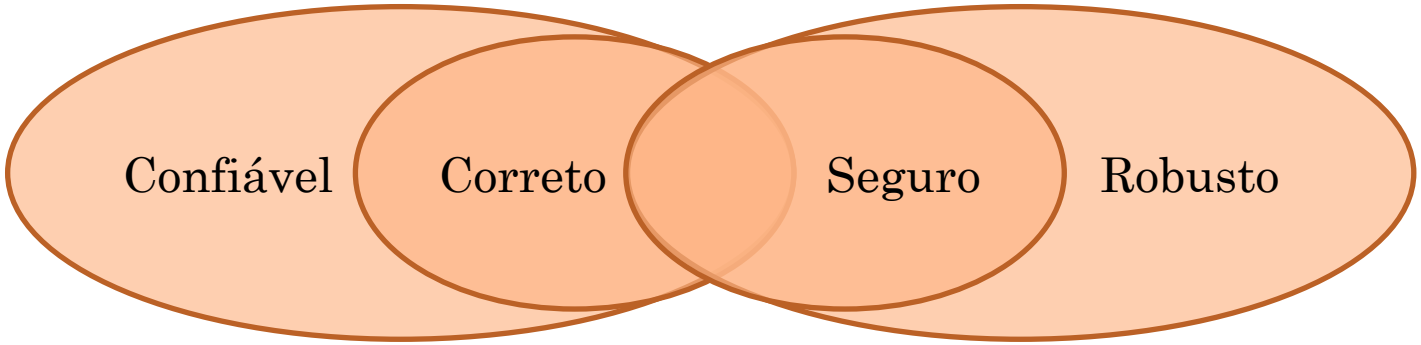
Quando saber se o software está pronto?

- Nível adequado de funcionalidade e qualidade
- Dependabilidade
  - Disponibilidade: qualidade do serviço em termos de tempo em serviço versus tempo fora de serviço (downtime)
  - Tempo médio entre falhas: qualidade do serviço em termos de tempo entre falhas
  - Confiabilidade: percentual de operações completadas com sucesso
- Ex: meta de disponibilidade de não mais de 30 minutos fora de serviço por mês e menos de uma falha a cada 1000 sessões de uso

Quando saber se o software está pronto?

- Confiança:
  - Corretude: consistência absoluta com a especificação. Quase nunca é possível com especificações não triviais.
  - Confiabilidade: aproximação estatística da corretude e é a probabilidade de comportamento correto durante o uso
  - Robustez: pondera as propriedades como mais ou menos críticas e define quais propriedades devem continuar funcionando mesmo em circunstâncias excepcionais
    - Falhar elegantemente
    - Ex: falha ao gravar arquivo em disco não para o software
  - Segurança: evitar comportamentos perigosos

Relação entre propriedades de confiança



Abordagens

- Nem sempre dá para realizar uma prova lógica do sistema
- O teste de todos os casos possíveis (um tipo de prova lógica) pode ser muito custoso
- Uso de técnicas imprecisas em uma das direções:
  - Pessimista: não aceitar um programa mesmo que satisfaça a propriedade analisada
    - Ex: verificações de sintaxe feitas por compiladores que verificam se a variável foi declarada antes de ser usada
  - Otimista: aceitar um programa mesmo que não satisfaça a propriedade analisada (não detectando todas as violações)
    - Ex: testes

Abordagens

- Pessimista e otimista podem ser complementares
- Podemos usar a dimensão de compromisso:
  - Utilizar uma propriedade que é mais fácil de verificar ou restringir a classe de programas a ser verificada
    - Ex: garantir que uma variável seja inicializada, testar tipo de uma variável

REFERÊNCIA

1. Pezzè, Mauro; Young Michal. Teste e análise de software – Processos, princípios e técnicas. Ed. Bookman. Capítulo 1. Figura 1.1
2. Sommeville, I. Engenharia de Software. Ed. Pearson. 8ª edição.