Estratégias de Teste, Modelos de Maturidade e Plano de Teste

Prof. Fabiano Cutigi Ferrari
1º semestre 2015
fabiano@dc.ufscar.br

LaPES – Laboratório de Pesquisa em Engenharia de Software
Departamento de Computação – UFSCar

Recados Iniciais

- Formação dos grupos
- Correção da P1

Roteiro

- Estratégias de Teste
- Processo de Teste
- Modelos de Maturidade
- Plano de Teste

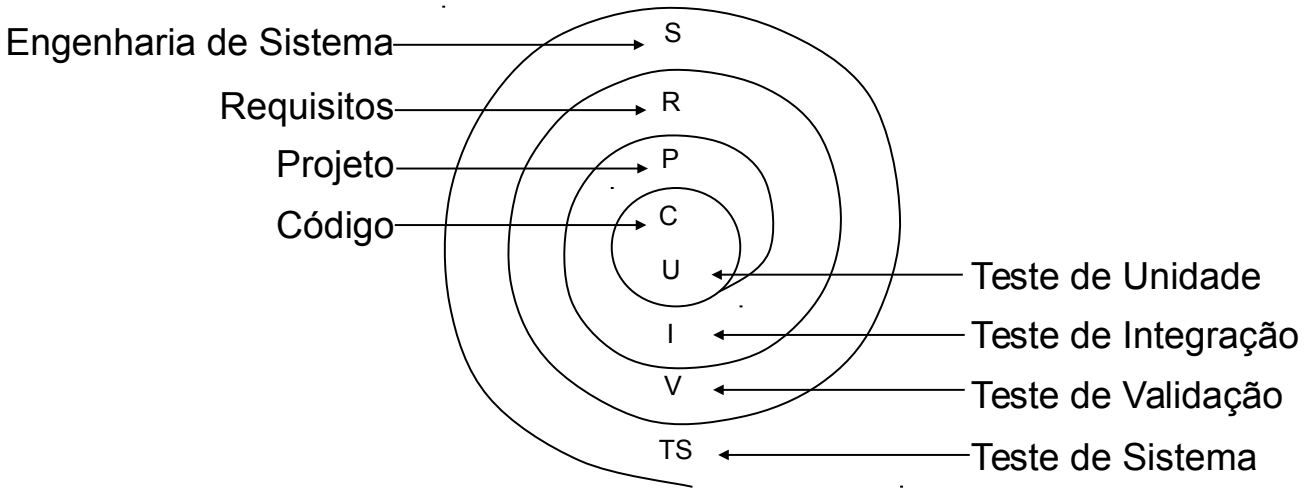
Teste de Software: Estratégias

Estratégias de Teste

- Aspectos genéricos das Estratégias de Teste:
  - A atividade de teste inicia-se no nível de módulos e caminha na direção da integração de todo o sistema.
  - Diferentes técnicas de teste são apropriadas para diferentes situações.
  - A atividade de teste, em geral, é realizada pela equipe de desenvolvimento e, no caso de grandes projetos, por um grupo de teste independente.
  - As atividades de teste e depuração são atividades diferentes, mas a depuração é necessária em qualquer estratégia de teste.

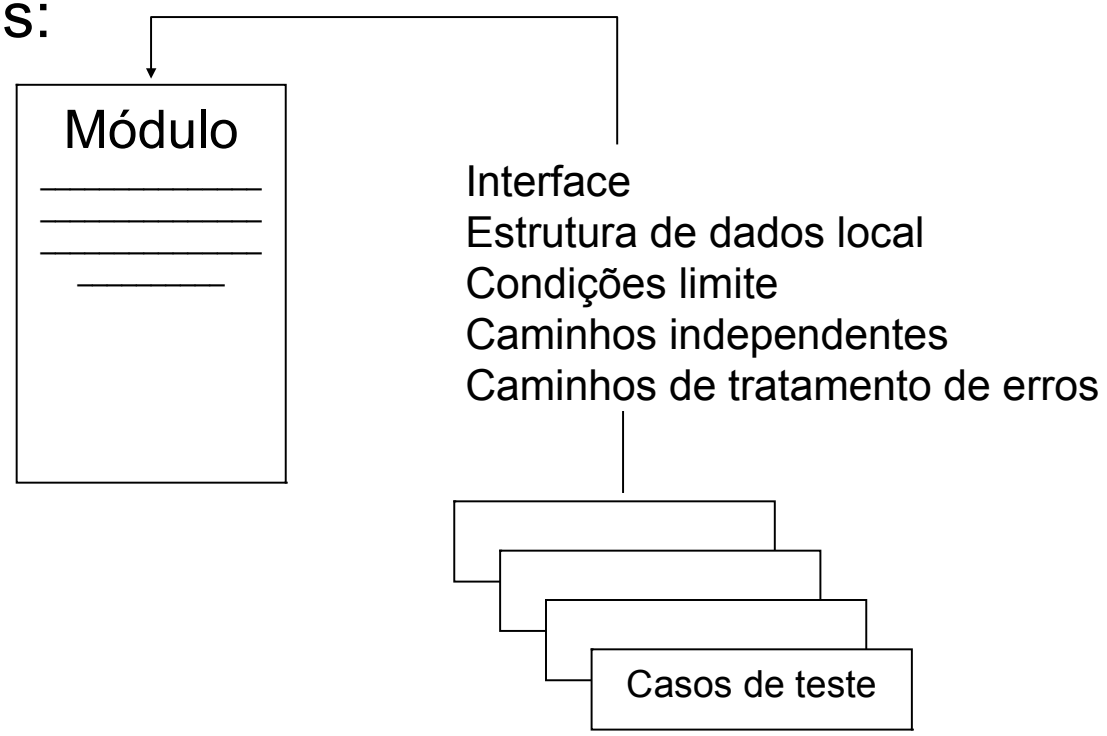
Estratégias de Teste

Relação entre o processo de desenvolvimento e uma estratégia de teste



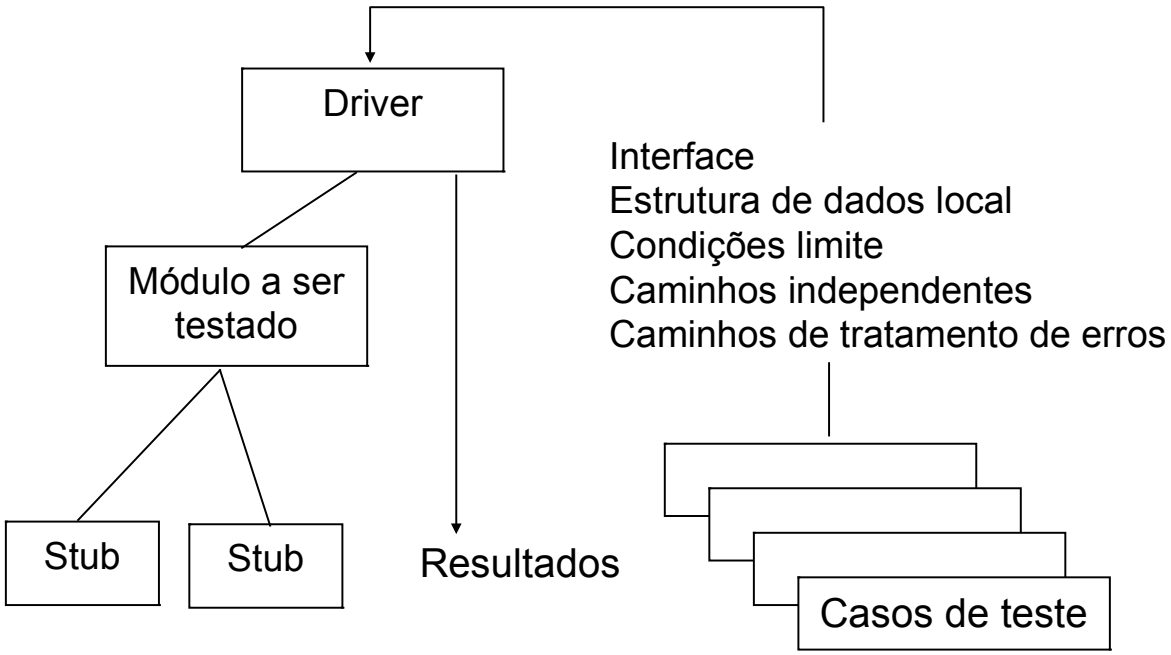
Teste de Unidade

- Concentra-se no módulo
- Utiliza a técnica de teste estrutural
- Pode ser realizado em paralelo para vários módulos
- Aspectos considerados:



Teste de Unidade

- Geralmente, um programa não é um módulo único, mas formado de diversos módulos que, para efeito do teste de unidade, devem ser testados separadamente.



driver : é um "programa principal" que aceita dados de casos de teste, passa esses dados para o módulo a ser testado e imprime os dados relevantes que ele recebe do módulo

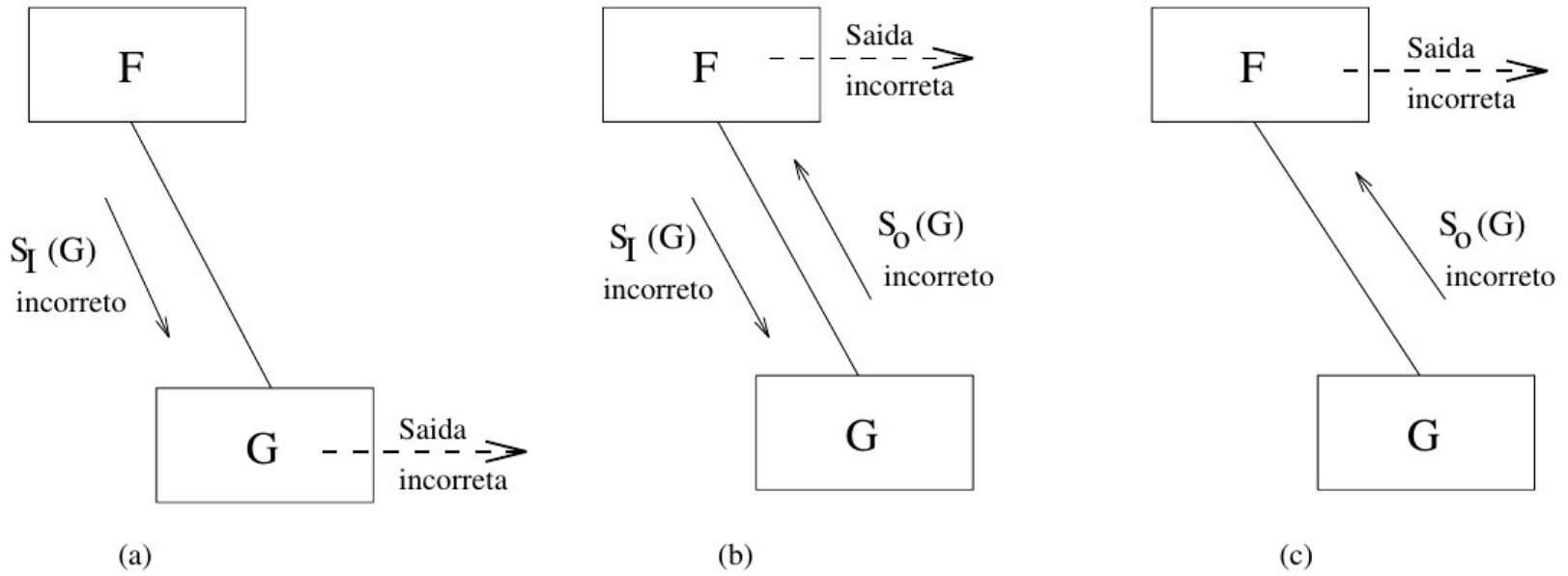
stub : são módulos que servem para substituir outros módulos que estejam subordinados, isto é, que são chamados pelo módulo testado; ele tem a interface do módulo subordinado, faz o mínimo de manipulação de dados, imprime uma verificação da entrada e retorna

Teste de Integração

- Constroi-se, de uma forma sistemática, a estrutura do programa. Realiza-se, ao mesmo tempo, testes para detectar defeitos de interface.
- Embora os módulos, depois do teste de unidade, funcionem corretamente de forma isolada, o teste de integração é necessário pois quando colocados juntos, várias situações inesperadas podem acontecer.

Teste de Integração
Exemplo de Modelo de Defeitos

- Exemplo de modelo de defeitos [1] relacionados à integração de dois módulos:



Legenda:
- F e G são funções (F invoca G)
- S_I(G): dados de entrada de G
- S_O(G): dados de saída de G

---

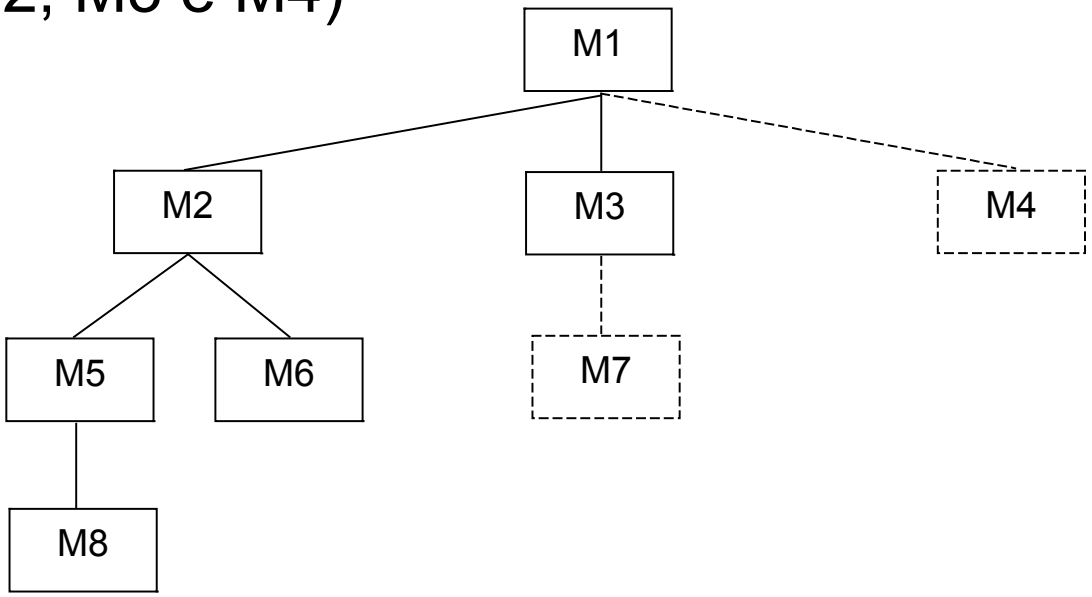
[1] Delamaro, M. E.: Mutação de Interface: Um Critério de Adequação Interprocedimental para o Teste de Integração. Tese (Doutorado) IFSC/USP, São Carlos, SP - Brasil, 1997.

Teste de Integração

- Integração dos módulos é feita através de uma abordagem incremental:
  - integração top-down
  - integração bottom-up

Teste de Integração

- **Integração Top-down**:
  - A integração dos módulos é feita de cima para baixo.
  - Pode ser realizada de duas maneiras:
    - por profundidade (M2, M5 e M8 ou M2, M5, M6 e M8)
    - por largura (M2, M3 e M4)



Teste de Integração

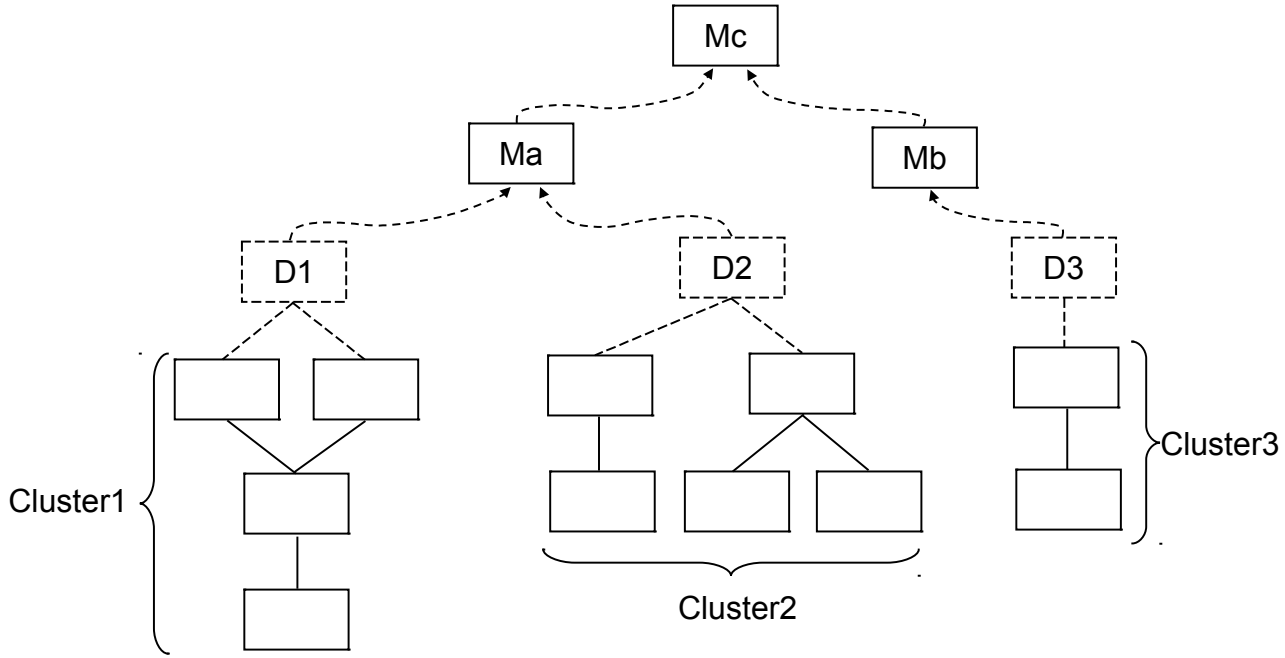
- **Integração Top-down**:
  - Por profundidade: permite que uma função específica do módulo principal possa ser testada por completo.
  - Nem sempre a construção de um stub é uma tarefa fácil, pois se a função do módulo real que ele representa for complexa, o stub tem que tratar os aspectos principais desse módulo para que o teste seja significativo.

Teste de Integração

- **Integração Top-down**: Processo em 5 passos:
  1. O módulo de controle principal é usado como um driver e substitui-se por stubs todos os módulos reais diretamente subordinados ao módulo principal;
  2. Dependendo da abordagem de integração a ser utilizada – por profundidade ou largura – os stubs são substituídos pelos módulos reais, um de cada vez;
  3. São realizados testes para cada módulo que seja integrado;
  4. Quando um teste é concluído, outro stub é substituído pelo módulo real;
  5. Teste de regressão (isto é, repetição de todos ou alguns dos testes já realizados) pode ser aplicado novamente para garantir que novos defeitos não tenham sido introduzidos.

Teste de Integração

- **Integração Bottom-up**:
  - A integração dos módulos é feita de baixo para cima.
  - Quando os módulos de níveis superiores vão ser testados, os módulos subordinados já estão prontos e, portanto, não se torna necessária a construção de stubs.



Teste de Integração

- A escolha por uma dessas duas abordagens de integração depende do tipo de software e, às vezes, do cronograma do projeto
- Em geral, uma integração combinada - sanduíche - é mais aconselhável:
  - módulos superiores → abordagem top-down
  - módulos mais inferiores → abordagem bottom-up

Teste de Integração

- Top-down:
  - Vantagem: testar logo no início as funções principais do software
  - Desvantagem: os stubs e a dificuldade de teste quando eles são usados
- Bottom-up:
  - Vantagem: não se precisa de stubs
  - Desvantagem: o módulo principal não existe enquanto todos módulos não estiverem testados

    → Um ou mais drivers devem ser criados

Teste de Validação

- O software está montado como um pacote e a validação do mesmo é realizada através de uma série de testes caixa preta.
- Finalidade:
  - Demonstrar a conformidade aos requisitos funcionais e de qualidade (ou seja, requisitos não funcionais).
  - Verificar se a documentação está correta.
- Duas possibilidades:
  - Aceito
  - Não está totalmente de acordo com os requisitos: negociar com o usuário

Teste de Validação

- Engloba o **Teste de Aceitação**:
  - Realizado pelo próprio usuário

- No caso de software desenvolvido para vários usuários:
  - Teste alfa: realizado pelo usuário no ambiente do desenvolvedor
  - Teste beta: realizado pelo usuário em seu próprio ambiente

Teste de Sistema

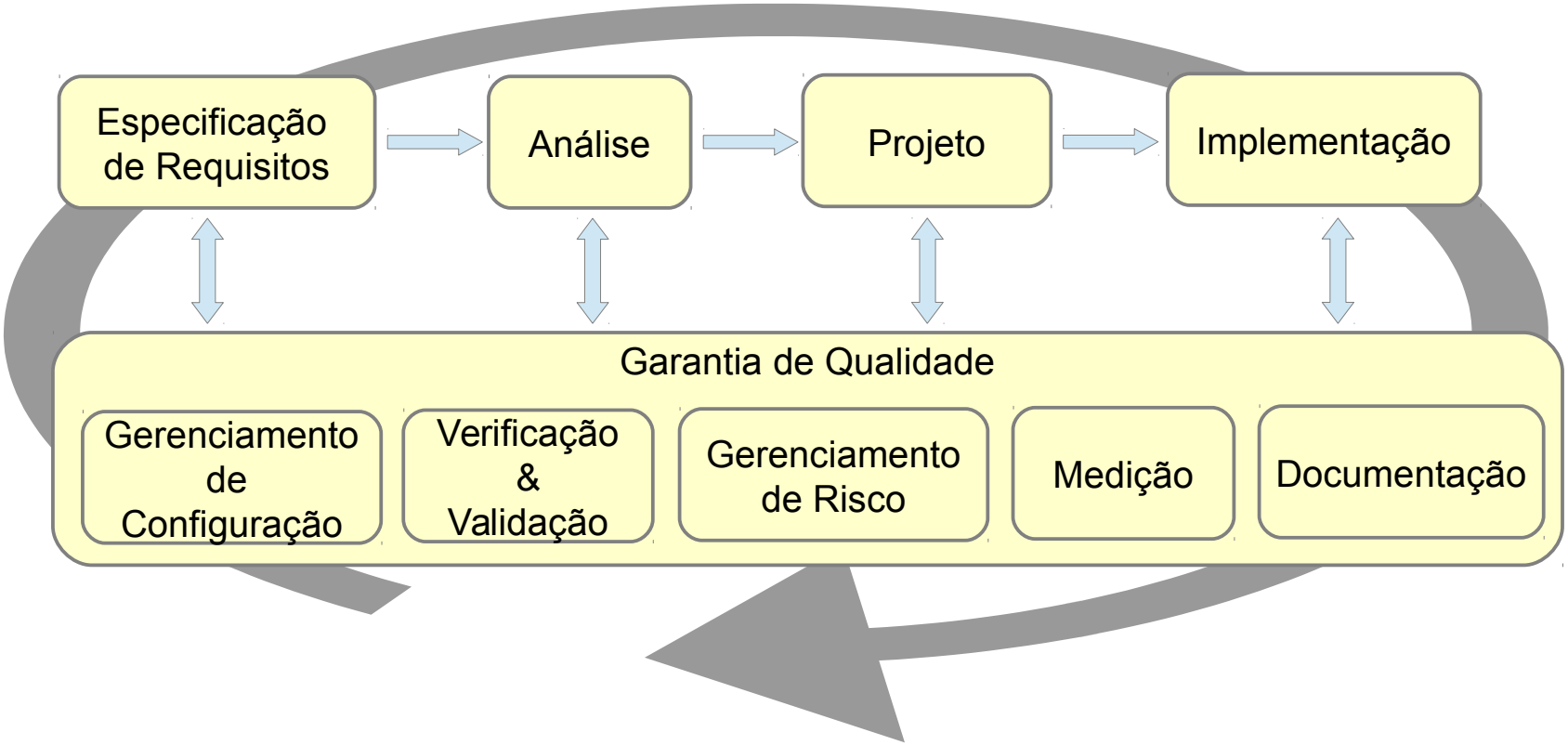
- Considera o software dentro do seu ambiente mais amplo (todos os aspectos de interação com ele, como outro hardware, software, pessoas, etc.)
- Corresponde a uma série de testes que tem por objetivo verificar se todos os elementos do sistema foram integrados adequadamente e realizam corretamente suas funções.

Teste de Sistema

- **Teste de segurança**: tem por objetivo verificar se todos os mecanismos de proteção protegem realmente o software de acessos indevidos.
- **Teste de estresse**: tem por objetivo confrontar os programas com situações anormais de frequência, volume ou recursos em quantidade.
- **Teste de desempenho**: tem por objetivo testar o tempo de resposta do sistema e é aplicado, geralmente, para sistemas de tempo real

Processo de Teste

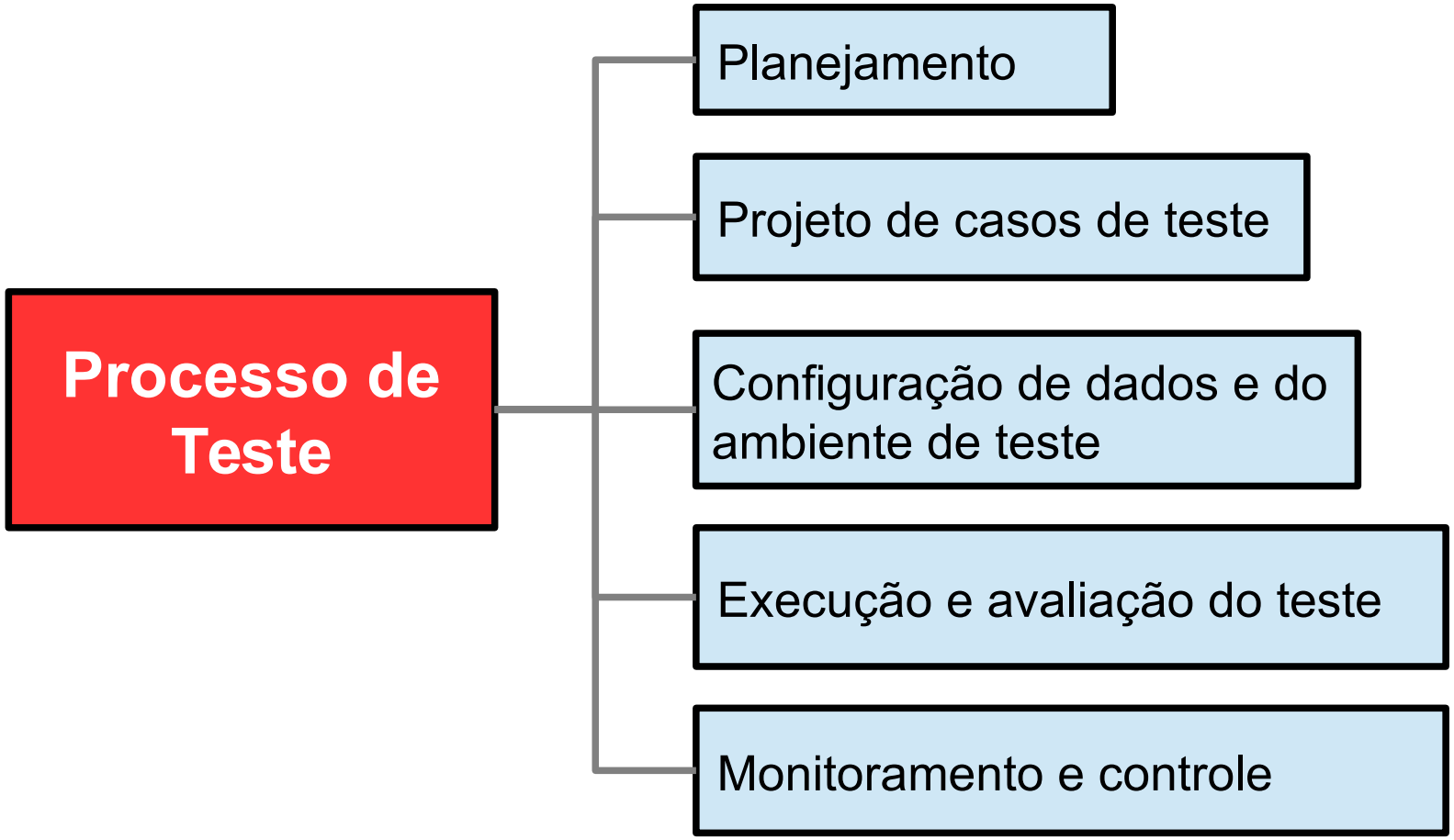
Processo de Desenvolvimento de Software



Processo de Desenvolvimento de Software

- Cada uma das etapas possui seu próprio processo.
  - Engenharia de Requisitos
  - Análise
  - Projeto
  - Implementação
  - Gerenciamento de Configuração
  - V & V
    - Inspeção
    - Análise Estática
    - Teste
  - etc...

Processo de Teste de Software [1]



[1] Höhn, E. N.: KITest: Um Arcabouço de Conhecimento e Melhoria de Processo de Teste. Tese (Doutorado) ICMC/USP, São Carlos, SP - Brasil, junho 2011.

Processo de Teste de Software

- **Planejamento**:
  - Análise de risco, definição de recursos e ambientes necessários, critérios de parada, cronograma, resultados esperados, equipe etc.

- **Projeto de casos de teste**:
  - Definição dos cenários, identificação e priorização de casos de teste, criação de casos de teste, identificar dados de teste específicos etc.

Processo de Teste de Software

- **Configuração de dados e do ambiente de teste**:
  - Desenvolver e priorizar procedimentos de teste, implementar o ambiente de teste, realizar pré-teste etc.
- **Execução e avaliação do teste**:
  - Executar casos de teste, relatar incidentes de teste, escrever log de teste, decidir sobre incidentes de teste, acompanhar o status dos incidentes etc.
- **Monitoramento e controle**:
  - Conduzir revisões do progresso do teste, monitorar defeitos, conduzir revisões de qualidade do produto, analisar problemas, tomar ações corretivas etc.

Modelos de Maturidade para Processo de Teste

TMMi – Test Maturity Model integration [1]

- Inspirado no CMMI
- Modelo de maturidade especializado na melhoria do processo de teste
- 5 níveis de maturidade

[1] TMMi Foundation: Test Maturity Model integration Version 3.1, 2010.

TMMi
Níveis de Maturidade

- **Nível 1: Inicial**
  - processo não definido (caótico)
  - teste mistura-se com depuração
  - ambiente organizacional instável para apoiar o processo de teste
  - sucesso depende de empreitadas “heroicas” da equipe
- **Nível 2: Gerenciado**
  - teste é uma atividade bem definida e separada das demais
  - práticas de teste são mantidas em épocas de sobrecarga
  - planos de teste são criados
  - a etapa de teste ainda é percebida (por alguns) como uma atividade que sucede a codificação

TMMi
Níveis de Maturidade

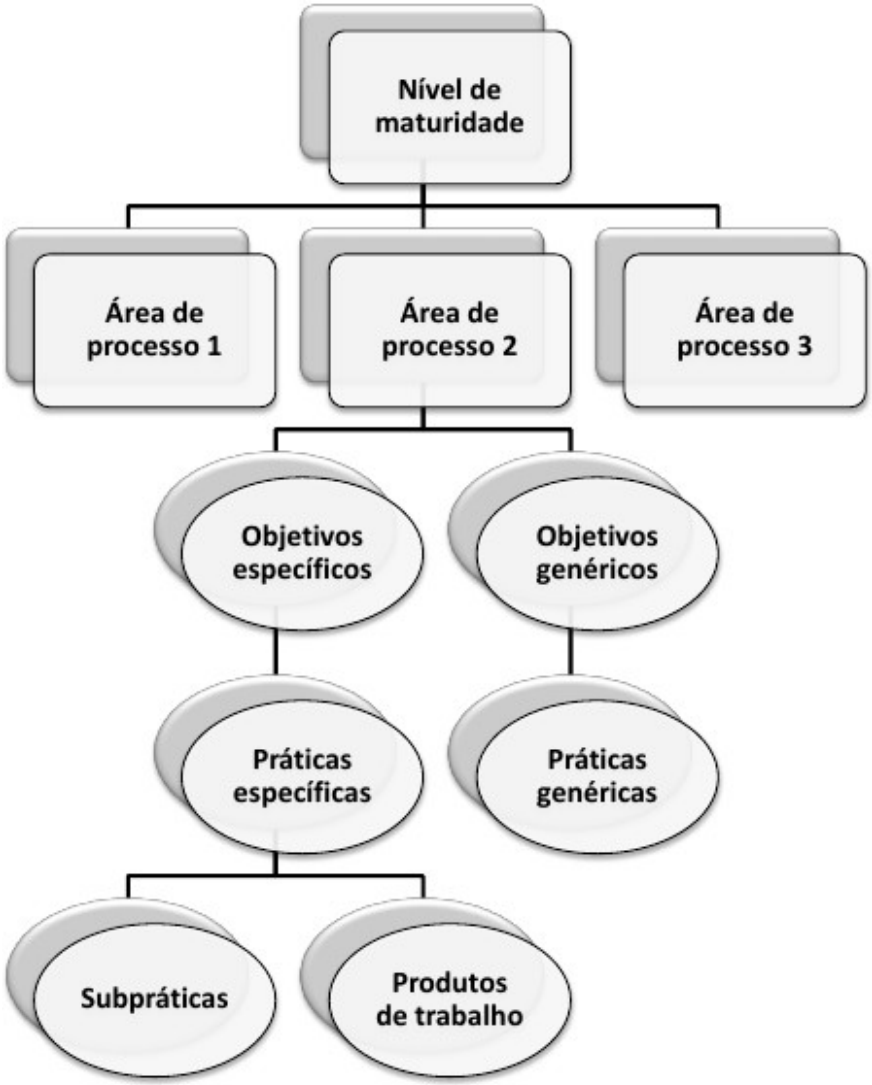
- **Nível 3: Definido**
  - teste não é mais uma etapa que sucede a codificação
    → é totalmente integrado ao ciclo de vida e *milestones*
  - planejamento é feito em fases iniciais do desenvolvimento
  - *expertise* na atividade embasa a definição dos planos de teste
  - a organização já possui um padrão de processo de teste
  - a equipe passa por processos de qualificação
- **Nível 4: Gerenciado Quantitativamente**
  - o processo de teste é mensurável quantitativamente
  - métricas de processo de teste são incorporadas ao repositório de métricas da organização, e são empregadas em estimativas
  - medições apoiam a definição de processo de avaliação da qualidade de produtos

TMMi
Níveis de Maturidade

- **Nível 5: Em Otimização**
  - já existe uma infraestrutura organizacional para apoiar um processo de teste totalmente definido e mensurável
  - o processo de teste é melhorado continuamente com base em medidas avaliadas estatisticamente
  - melhorias incrementais e inovativas, com apoio tecnológico, embasam a melhoria do processo de teste

TMMi
Áreas de Processo, Objetivos e Práticas

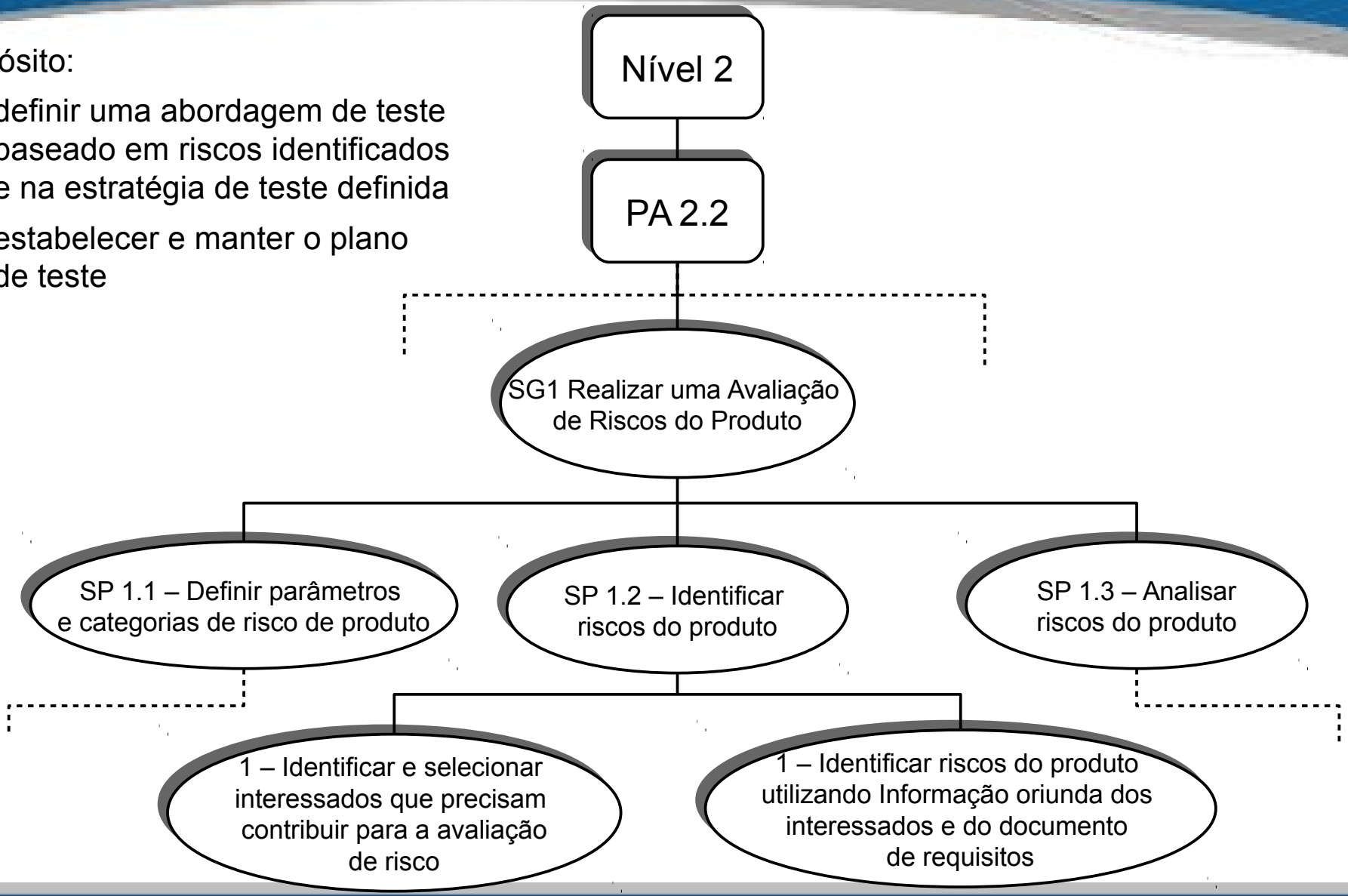
- Cada nível de maturidade é composto por Áreas de Processo (*Process Area – PA*)
  - PAs identificam questões que precisam ser tratadas para atingir melhorias do processo pretendidas
- Cada Área de Processo é formada por Objetivos Específicos (*Specific Goals - SG*)
- Cada Objetivo Específico é formado por Práticas Específicas (Specific Practices - SP)
- Há também Objetivos Genéricos (e respectivas Práticas Genéricas), que podem se relacionar com mais de uma PA.



Exemplo: Área de Processo 2.2 – Planejamento de Teste (Nível 2)

Propósito:

- definir uma abordagem de teste baseado em riscos identificados e na estratégia de teste definida
- estabelecer e manter o plano de teste



Opções ao TMMi

- **MPT.Br[1] – Melhoria de Processo de Teste**
  - estrutura em níveis similar ao TMMi
    - 1 – Parcialmente Gerenciado
    - 2 – Gerenciado
    - 3 – Definido
    - 4 – Prevenção de Defeitos
    - 5 – Automação e Otimização
  - voltado para o cenário brasileiro (análogo ao MR-MPS[2])
- **ISO/IEC/IEEE 29119[3] – Software Testing**
  - conjunto de padrões para teste de software que podem ser empregados durante o ciclo de desenvolvimento
  - visa a prover à organização padrão internacional de qualidade em teste de software
  - contempla Processo de teste, Documentação de teste, e Técnicas de teste

[1] http://mpt.org.br/mpt/ - acessado em 12/05/2015
[2] http://www.softex.br/mpsbr/ - acessado em 12/05/2015
[3] http://www.softwaretestingstandard.org/ - acessado em 12/05/2015

Plano de Teste

Perguntas Iniciais

- Por que precisamos planejar?

O tempo pode acabar

O dinheiro pode acabar

O Plano de Teste

- Documento que contém todas as informações sobre a atividade de teste pretendida:
  - Equipe de teste
  - Software de suporte
  - Cronograma das atividades
  - Técnicas e critérios a serem seguidos
  - Forma de execução dos casos de teste
  - Forma de avaliação dos resultados

  - Ver exemplo...