SSC721 – Teste e Inspeção de Software

Inspeção de Software – Código

Profa. Ellen Francine Barbosa
francine@icmc.usp.br

Instituto de Ciências Matemáticas e de Computação — ICMC/USP

Roteiro

- Revisões de Software
- Revisões Técnicas Formais
- Reunião de Revisão Técnica
- Inspeção de Software
- Inspeção em Código
- Exercício de Fixação

Aula Anterior...

- Qualidade de software.
- V&V envolvem atividades de análise estática e de análise dinâmica.
- Análise Estática
  - As revisões são o exemplo mais clássico de análise estática.

Revisões de Software I

- Meio efetivo para melhorar a qualidade de software.
  - Filtro para o processo de Engenharia de Software.
- Podem ser aplicadas em vários pontos durante o desenvolvimento do software.
- Maneira de usar a diversidade de um grupo de pessoas para:
  - Apontar melhorias necessárias ao produto.
  - Confirmar as partes de um produto em que uma melhoria não é desejada ou não é necessária.
  - Realizar um trabalho técnico de qualidade mais uniforme de forma a torná-lo mais administrável.

Objetivo das Revisões de Software I

- Encontrar erros durante o processo de desenvolvimento, de forma que eles não se transformem em defeitos depois da entrega do software.
- Descoberta precoce dos erros.
  - Melhoria da qualidade já nas primeiras fases do processo de desenvolvimento.
  - Aumento da produtividade e diminuição dos custos.
    - Erros são detectados quando sua correção é mais barata.

Tipos de Revisões

- Discussão informal de um problema técnico.
- Apresentação do projeto de software para uma audiência de clientes, administradores e pessoal técnico.
- Revisões Técnicas Formais (RTF), as quais incluem avaliações técnicas do software realizadas em pequenos grupos.
  - Inspeção
  - Walkthrough
  - Peer-Review

Revisões Técnicas Formais

- Métodos de RTF:
  - Inspeção
  - Walkthrough
  - Peer-Review

Inspeção

- Método de análise estática para verificar a qualidade de um produto de software.
- Como detectar defeitos?
  - Técnicas de leitura

Walkthrough

- Procedimento similar ao procedimento para condução de uma inspeção.
- A diferença fundamental está na maneira como a sessão de revisão é conduzida.
  - Em vez de ler o programa ou checar os erros por meio de um *checklist*, os participantes simulam sua execução.
  - Papel adicional: testador.
    - Elaborar um pequeno conjunto de casos de teste (em papel).
    - Monitorar e controlar os resultados obtidos.

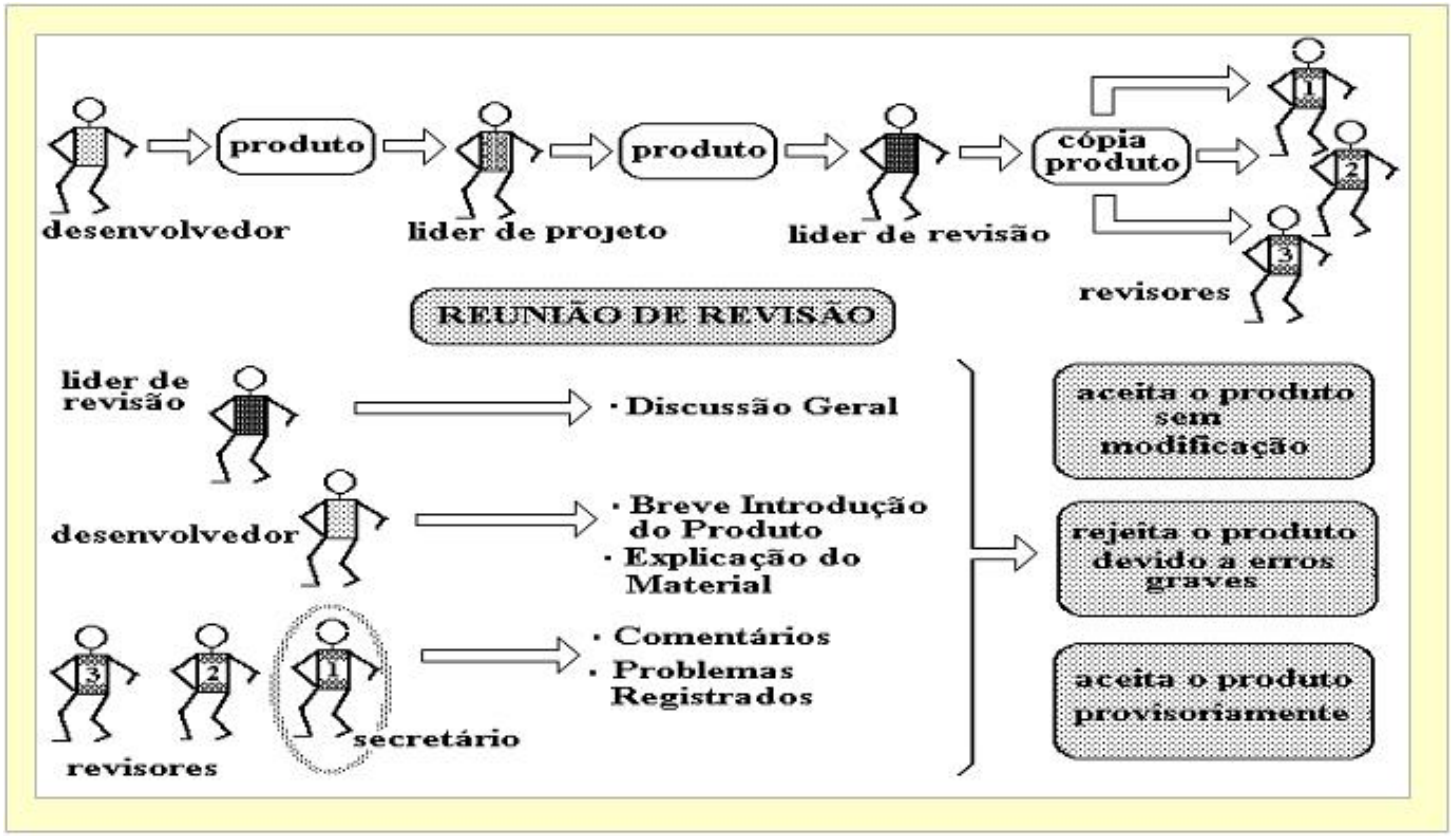
Peer-Review

- Conduzida por pares de programadores.
  - Mesmo nível de conhecimento.
- Aplicada ao código.
- Reuniões com duração de 1 a 2 horas.
  - Somente um programa ou parte dele (rotinas) deve ser revisado.
- Resultados são publicados em um relatório informal.
  - Não faz parte da documentação oficial do projeto.

Reunião de Revisão Técnica I

- Independentemente do formato da RTF, toda reunião de revisão deve seguir as seguintes recomendações:
  - Envolver de 2 a 5 pessoas.
  - Deve haver uma preparação para a reunião.
    - A preparação não deve exigir mais de 2 horas de trabalho de cada pessoa.
  - A reunião deve durar menos de 2 horas.
  - Deve-se focalizar uma parte específica do produto.
    - Maior probabilidade de descobrir erros.

Reunião de Revisão Técnica II



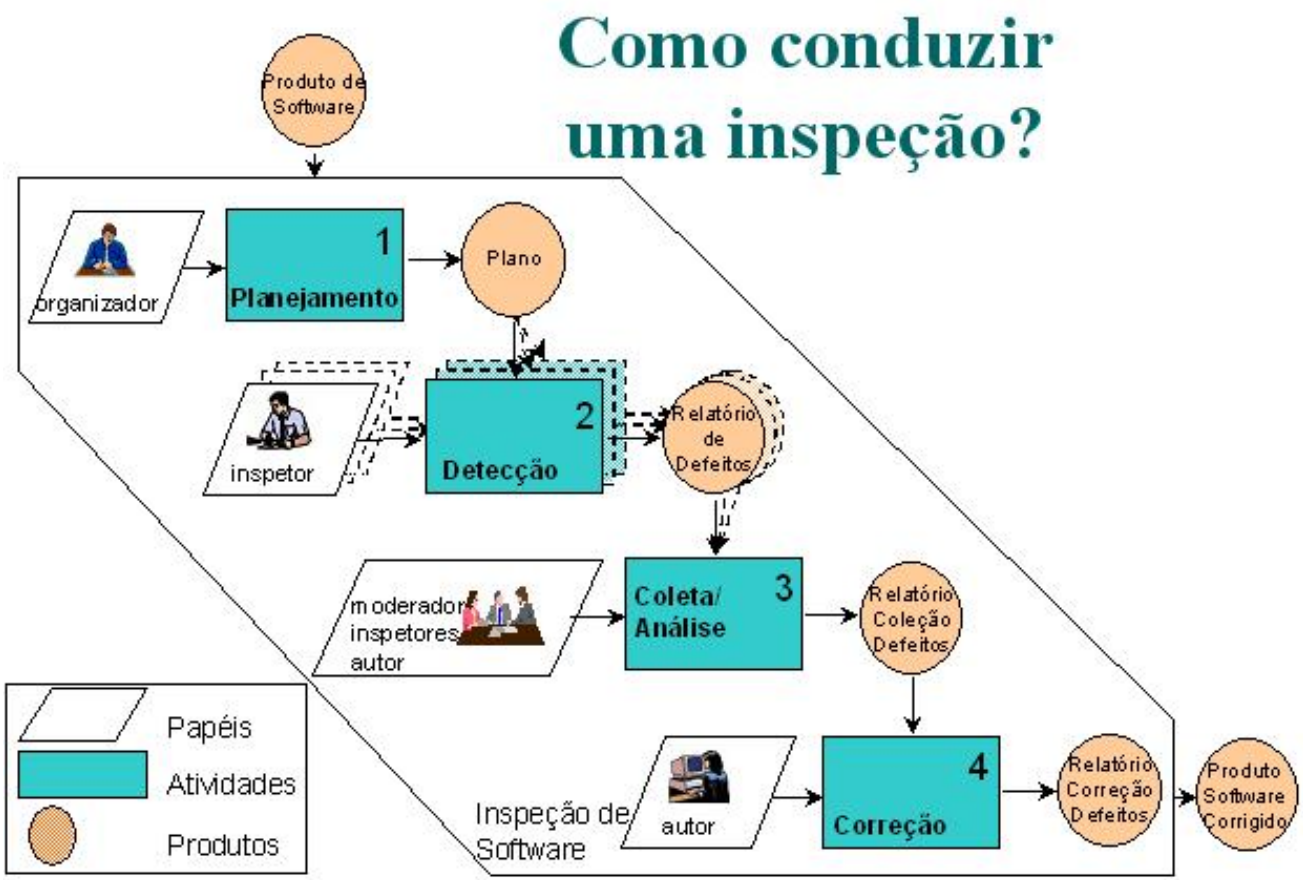
Diretrizes para Revisão

- Revise o produto, não o produtor.
- Fixe e mantenha uma agenda.
- Limite o debate e a refutação.
- Relacione as áreas problemáticas.
- Faça anotações por escrito.
- Limite o número de participantes e insista em uma preparação antecipada.
- Desenvolva uma lista de conferência (checklist) para cada produto que provavelmente será revisto.
- Atribua recursos e uma programação de tempo para as revisões.
- Realize um treinamento significativo para todos os revisores.
- Reveja suas antigas revisões.

Inspeção de Software

- Método de análise estática para verificar a qualidade de um produto de software.
- Pode-se inspecionar tanto produtos de software como também projetos de software.
  - Diferencial está na seleção dos aspectos que devem ser considerados durante a revisão.
- **Inspeção em Código**.
- **Inspeção em Documentos de Requisitos**.

Etapas da Inspeção



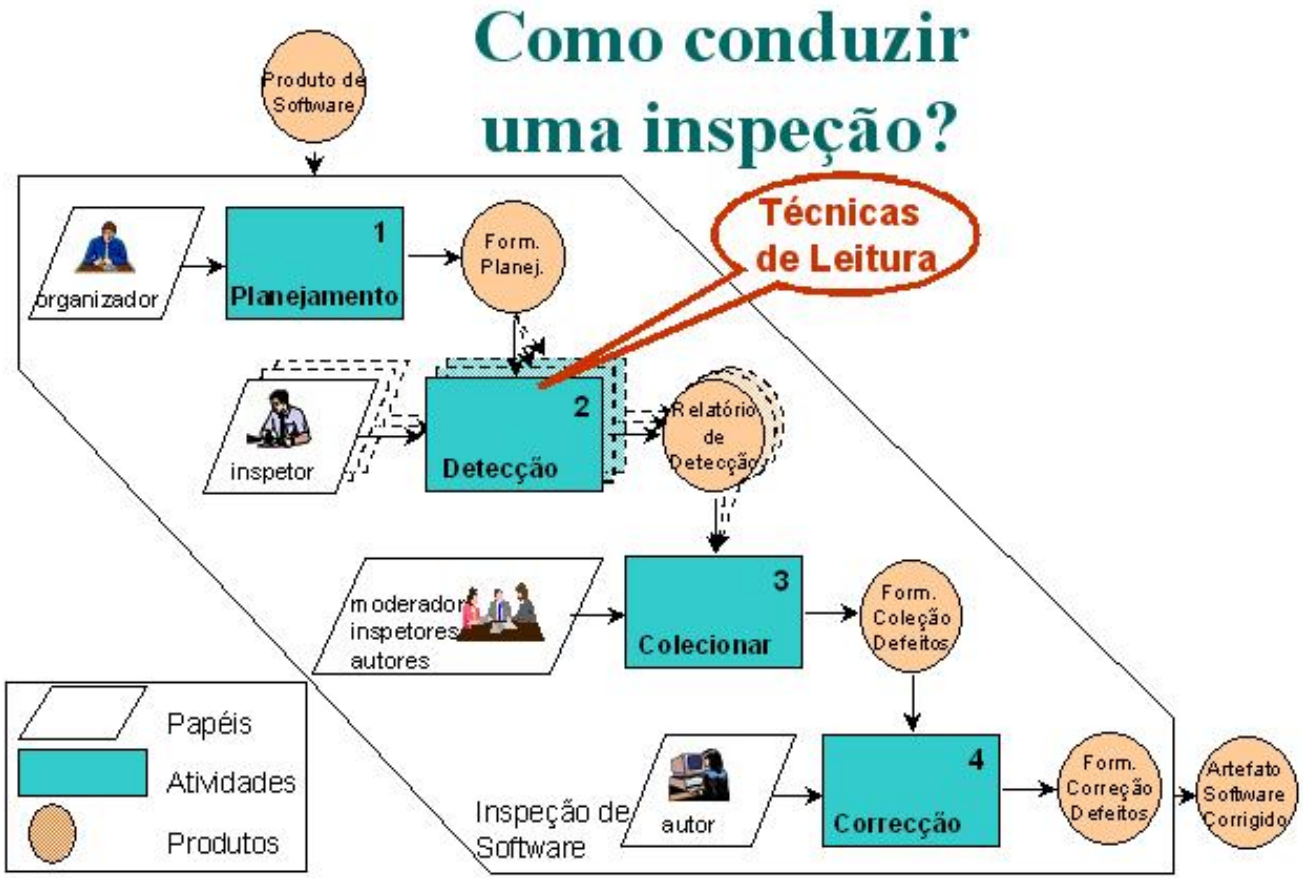
Técnicas de Leitura para Inspeção I

- Como detectar defeitos?
  - Lendo o documento.
  - Entendendo o que o documento descreve.
  - Verificando as propriedades de qualidade requeridas.
- Problema:
  - **Em geral não se sabe como fazer a leitura de um documento!!!**
- Razão:
  - Em geral, os desenvolvedores aprendem a escrever documento de requisitos, código, projeto, mas não aprendem a fazer uma leitura adequada dos mesmos.

Técnicas de Leitura para Inspeção II

- Solução:
  - Fornecer **técnicas de leitura** bem definidas.
- Benefícios:
  - Aumenta a relação custo/benefício das inspeções.
  - Fornece modelos para escrever documentos com maior qualidade.
  - Reduz a subjetividade nos resultados da inspeção.

Técnicas de Leitura para Inspeção III



Técnicas de Leitura para Inspeção IV

- O que é uma técnica de leitura?
  - Conjunto de instruções fornecido ao revisor dizendo como ler e o quê procurar no produto de software.
- Algumas técnicas de leitura:
  - Ad-hoc
  - Checklist
  - Code Reading - **código**
  - Leitura Baseada em Perspectiva (PBR) - **DR**
  - Object Oriented Reading Techniques (OORTs) - **modelos de APOO**

Ad-hoc

- Os revisores não utilizam nenhuma técnica sistemática de leitura.
- Cada revisor adota sua própria maneira de ler o documento.
- Desvantagens:
  - Depende da experiência do revisor.
  - Não é repetível.
  - Não é passível de melhoria pois não existe um procedimento a ser seguido.

Checklist

- Similar ao ad-hoc, mas cada revisor recebe um **checklist**.
  - Os itens do checklist capturam lições importantes que foram aprendidas em inspeções anteriores no ambiente de desenvolvimento.
  - Itens do checklist podem explorar defeitos característicos, priorizar defeitos diferentes e estabelecer questões que ajudam o revisor a encontrar defeitos.

Inspeção em Código

- Visa a encontrar defeitos no código, realizando uma análise estática.
- *Mas... que tipo de defeito???*

Taxonomia de Defeitos em Código

- Duas classificações de defeitos em código:
  - Classificação I:
    - Defeitos de Omissão
    - Defeitos de Comissão
  - Classificação II:
    - Defeitos de Inicialização
    - Defeitos de Computação
    - Defeitos de Controle
    - Defeitos de Interface
    - Defeitos de Dados
    - Defeitos Cosméticos

Taxonomia de Defeitos em Código - I

- Defeitos de Omissão: Esquecimento de algum elemento no programa.
- Exemplo: Falta de um comando que iria atribuir um valor a uma variável.

- Defeitos de Comissão: Segmento de código incorreto.
- Exemplo: Um operador aritmético errado é usado em uma expressão.

Taxonomia de Defeitos em Código - II I

- Defeito de Inicialização: Inicialização incorreta de uma estrutura de dados.
- Exemplo: Atribuir um valor errado a uma variável.

- Defeitos de Computação: Qualquer computação incorreta para geração do valor de uma variável.
- Exemplo: Um operador aritmético errado é usado em uma expressão de atribuição de valor para variável.

Taxonomia de Defeitos em Código - II II

- Defeito de Controle: Causa a execução de um caminho de controle errado para um valor de entrada.
- Exemplo: Predicado incorreto em um comando IF-THEN-ELSE.

- Defeito de Interface: Quando um módulo usa ou faz suposições sobre dados que não fazem parte do seu escopo.
- Exemplo: Passagem de um argumento incorreto para um procedimento, ou quando um módulo assume que irá receber um array inicializado com zero.

Taxonomia de Defeitos em Código - II III

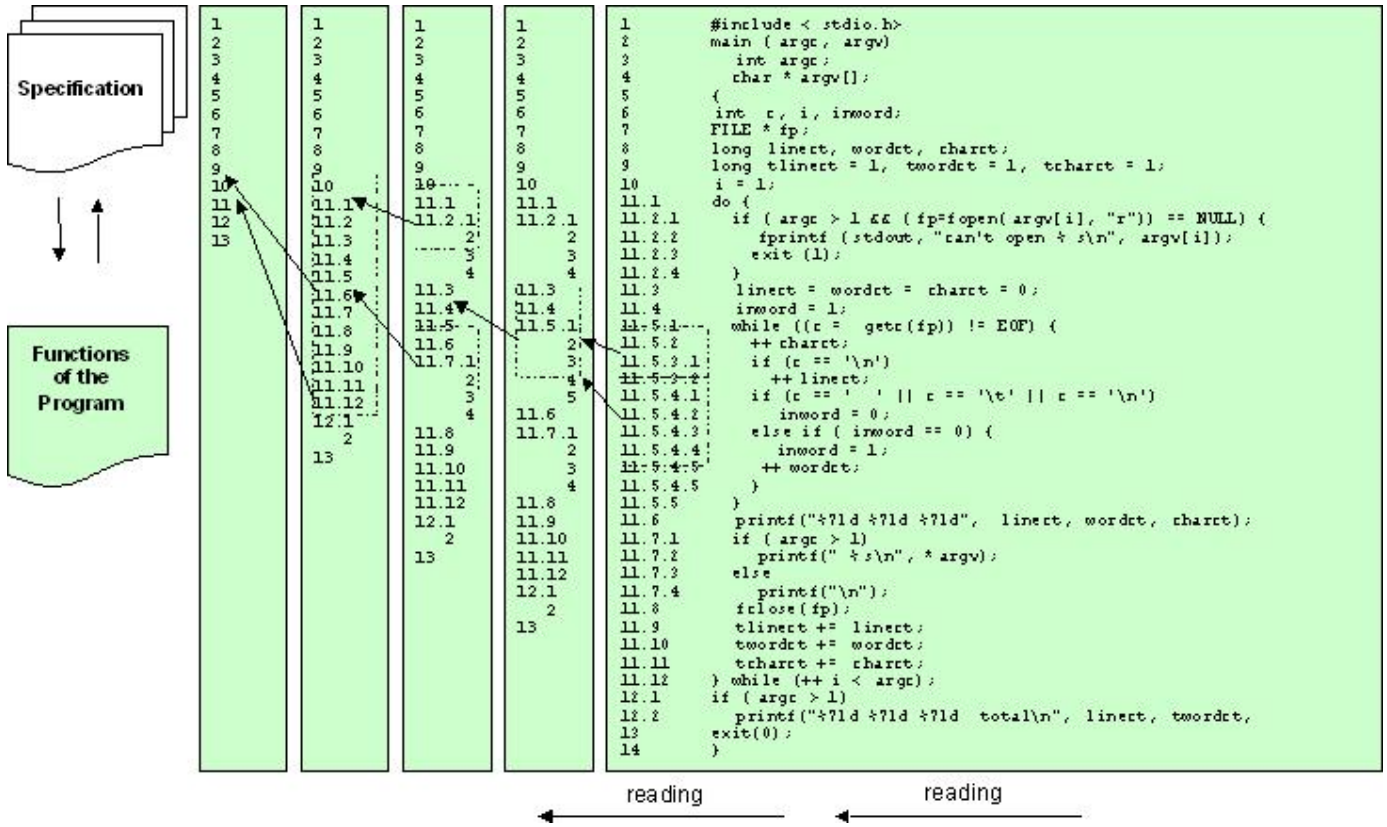
- Defeitos de Dados: Uso incorreto de uma estrutura de dados.
- Exemplo: Determinar incorretamente o último índice de um array.

- Defeitos Cosmético: Erro de escrita no programa.
- Exemplo: Uma mensagem de erro com erro de escrita (ortografia, gramática).

Code Reading I

- Técnica de leitura utilizada para detectar defeitos em código.
- Consiste em desenvolver abstrações funcionais a partir do código, para determinar a funcionalidade do programa.
  - Stepwise abstraction

Code Reading II



Code Reading III

- Os revisores identificam subprogramas no código e escrevem suas próprias especificações para os subprogramas (abstrações de funcionalidade).
  - As abstrações construídas devem ser combinadas em uma abstração mais geral, até que se tenha capturado a função completa do programa.
- Inconsistências são detectadas comparando a especificação original com a especificação construída por meio das abstrações.

Code Reading IV

- Exemplo de abstrações de código:

```
20   while ((c = getc(fp)) != EOF) {
21     ++charct;
22     if (c == '\n')
23             ++linect;
24     if (c == ' ' || c == '\t' || c == '\n')
25             inword = 0;
26     else if (inword == 0) {
27             inword = 1;
28             ++wordct;
29     }
30   }
```

Code Reading V

```
20   while ((c = getc(fp)) != EOF) {
21     ++charct;

22     if (c == '\n')
23             ++linect;

24     if (c == ' ' || c == '\t' || c == '\n')
25             inword = 0;
26     else if (inword == 0) {
27             inword = 1;
28             ++wordct;
29     }

30   }
```

Code Reading VI

```
20   while ((c = getc(fp)) != EOF) {
21     ++charct;
22     if (c == '\n')
23             ++linect;
24     if (c == ' ' || c == '\t' || c == '\n')
25             inword = 0;
26     else if (inword == 0) {
27             inword = 1;
28             ++wordct;
29     }
30   }
```

Abstrações linhas 22-23

```
If (c = '\n') then
   linect ← linect + 1;
```

Abstrações linhas 24-29

```
If (c = ' ', or c = '\t' or c = '\n' ) then
   inword ← 0;
else if (inword = 0) then
   inword ← 1;
   wordct ← wordct +1;
```

Abstrações linhas 20-30

```
c ← getc(fp)
while (c ≠ EOF) do
   charct ← charct + 1;
   22-23;
   24-29;
   c ← getc(fp);
```

Princípios para Aplicação do Code Reading I

1. Detectar como o programa irá falhar durante a execução.
   - Construir as abstrações.
     - Determinar o comportamento de um componente através da leitura.
     - Determinar as dependências entre as funções individuais no código. Inicie aplicando a técnica de leitura de código nas funções *mais internas* caminhando em direção às *mais externas*.
     - Desenvolver um entendimento da estrutura de cada função individual, identificando as estruturas elementares (seqüência, condições, atribuições, repetições).
     - Marcar as estruturas identificadas, desenhando quadrados para delimitá-las.

Princípios para Aplicação do Code Reading II

- Combinar essas estruturas em outras mais externas até construir um quadrado que representa uma única função.
- Agregar as funções elementares de acordo com o fluxo de controle do programa.

- Isolar as falhas.
  - Comparar as abstrações com a especificação para detectar possíveis falhas.

2. Isolar os defeitos que levariam a falhas.
   - Buscar as causas do comportamento inadequado (defeitos) do programa.
   - O entendimento do programa adquirido durante a leitura facilita a localização do defeito.

Vantagens

- Detecção antecipada de defeitos.
- Aprende-se pela experiência.
  - Participantes aprendem os padrões e o raciocínio utilizado na detecção de defeitos.
  - Participantes aprendem bons padrões de desenvolvimento.
- A longo prazo...
  - A inspeção convence os participantes a desenvolverem produtos mais compreensíveis e mais fáceis de manter.

As inspeções ajudam a integrar o processo de **detecção** de defeitos com o processo de **prevenção** de defeitos.

Exercício de Fixação