

Inspeção

Profa. Sandra Fabbri

Departamento de Computação - UFSCar

Inspeção de Software

- **Definição**

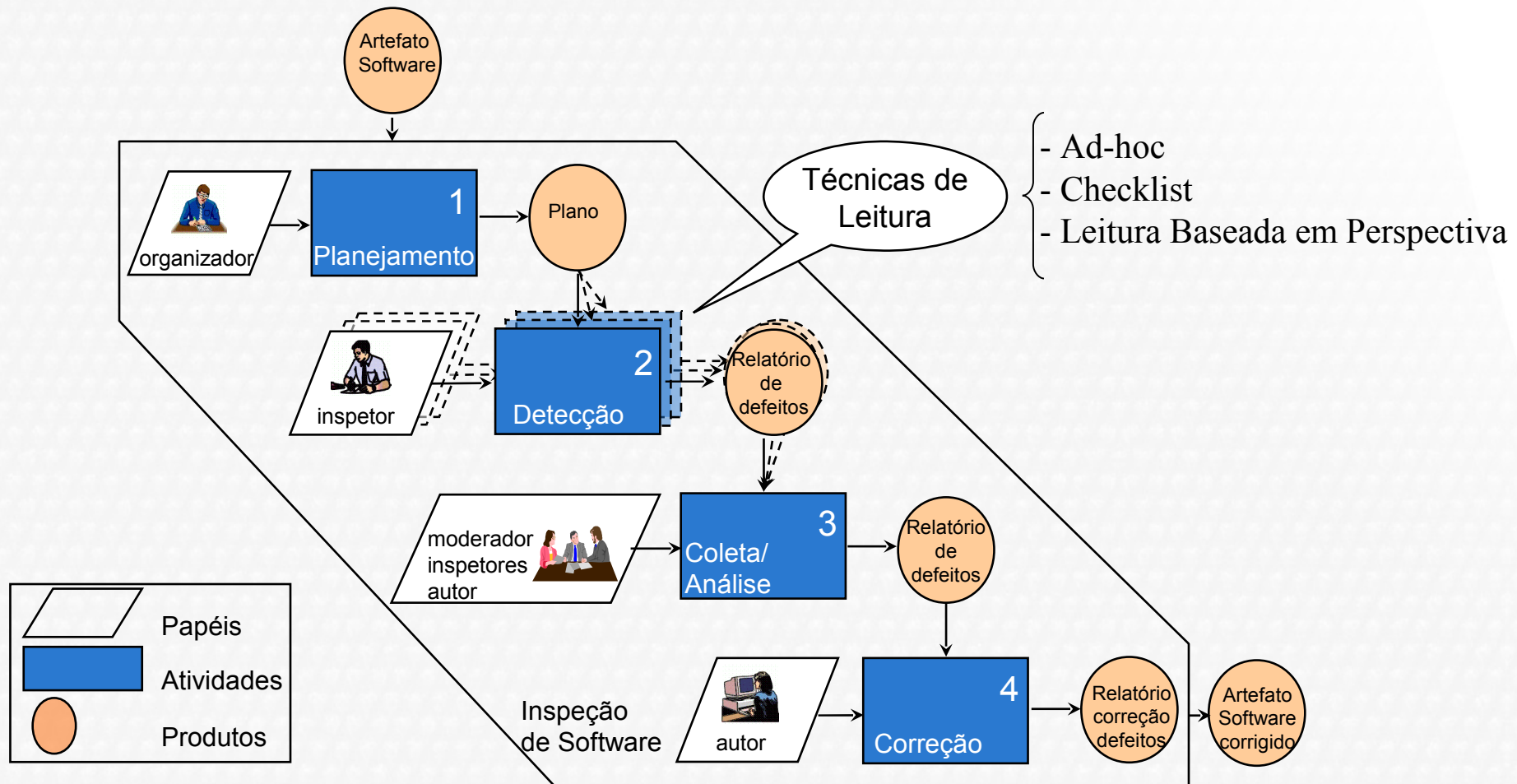
- é um método de análise estática para verificar propriedades de qualidade de produtos de software.

- **Características:**

- processo estruturado e bem definido.
- a equipe de inspeção consiste, normalmente, de pessoal técnico.
- os participantes possuem papéis bem definidos.
- os resultados da inspeção são registrados.

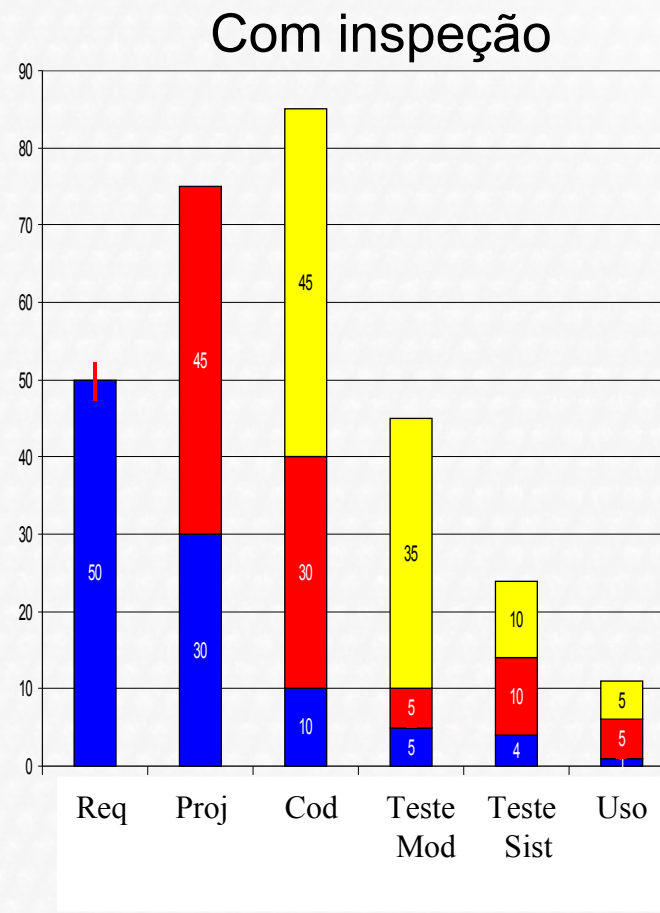
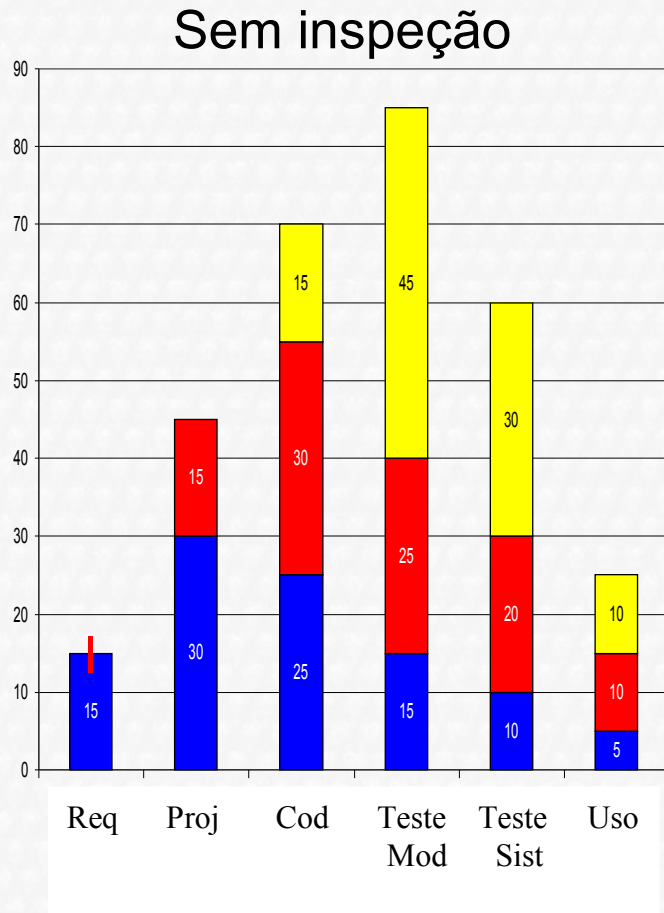
Inspeção de Software

■ Como conduzir uma inspeção?



Benefícios da Inspeção: detecção de defeitos antecipada

- As inspeções melhoram a qualidade desde o início do projeto detectando mais defeitos desde a fase de requisitos.

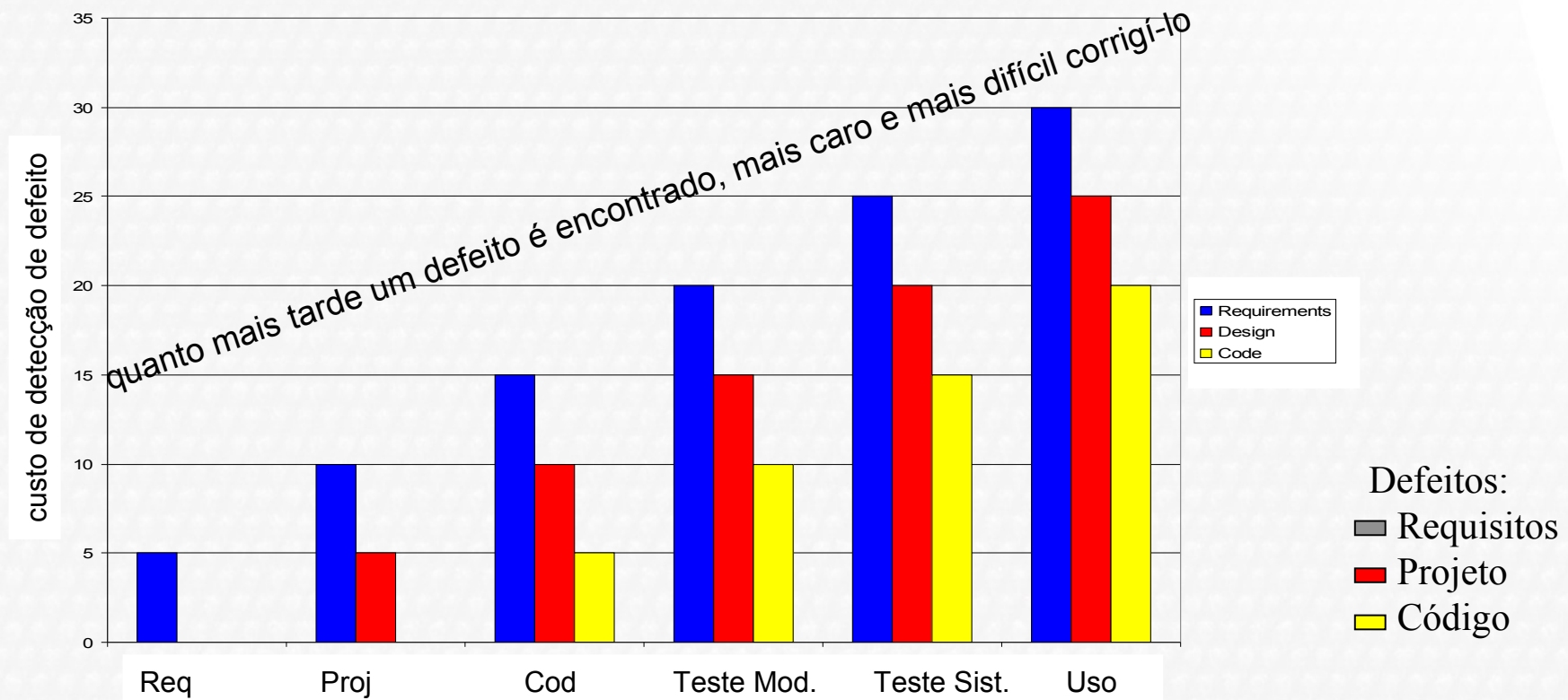


Defeitos:

- Requisitos
- Projeto
- Código

Benefícios da Inspeção: produtividade e custo

- As inspeções melhoram a produtividade uma vez que os defeitos são encontrados quando são mais fáceis e mais baratos para corrigir.



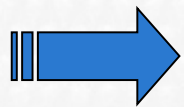
Benefícios Qualitativos da Inspeção

- **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**

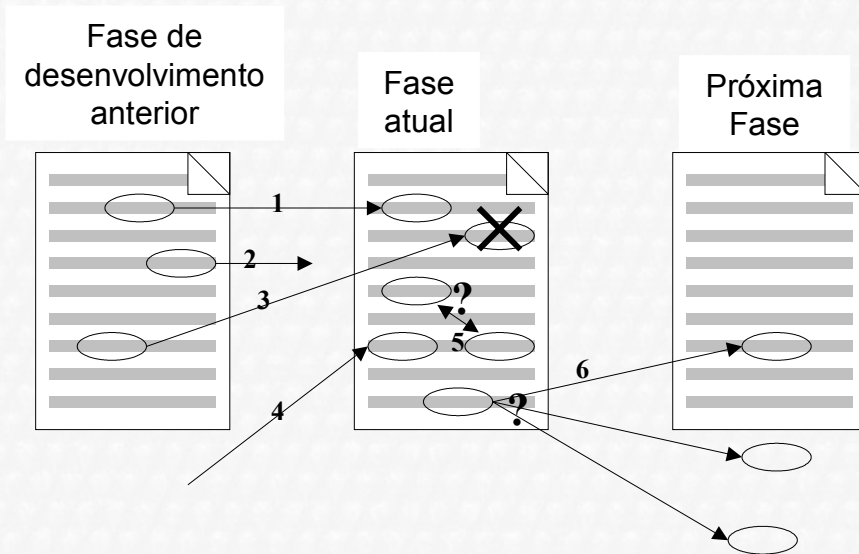
- a inspeção convence os participantes a desenvolver produtos mais compreensíveis e mais fáceis de manter.



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

Defeitos do Software

- Os defeitos surgem quando o desenvolvimento não está de acordo com a especificação já desenvolvida ou quando podem causar problemas daquele ponto em diante.



Situação ideal:

1. A informação é transformada corretamente.

Tipos de Erros:

2. A informação é perdida durante a transformação.
3. A informação é transformada incorretamente.
4. Informação estranha é introduzida.
5. A mesma informação é transformada em diversas ocorrências inconsistentes.
6. A mesma informação possibilita diversas transformações inconsistentes.

- **Interpretação de “defeito”**

- “defeito”: qualquer propriedade de qualidade que não seja satisfeita.
- deve-se evitar focar apenas na corretitude, como se fosse a única propriedade de qualidade.

Taxonomia de Defeitos

- **Definição:** são as classes de defeitos que serão usadas para classificar os defeitos encontrados.
- **Classes:**
 - Omissão (O): qualquer informação necessária que tenha sido omitida.
 - Fato Incorreto (FI): informação que consta do artefato mas que seja contraditória com o conhecimento que se tem do domínio de aplicação.
 - Inconsistência (I): informação que consta do artefato mais de uma vez e em cada ocorrência ela é descrita de forma diferente.
 - Ambiguidade (A): informação que pode levar a múltiplas interpretações.
 - Informação Estranha (IE): informação que, embora relacionada ao domínio, não é necessária para o sistema em questão.
 - Diversos (D): qualquer outro tipo de defeito que não se encaixe nas outras categorias. Ex: declarações em seções erradas.

Exemplo: Omissão

■ Omissão de Funcionalidade:

- Informação que descreva algum comportamento desejado do sistema foi omitida do Documento de Requisitos (DR).
 - Ex: considere um sistema de biblioteca e os seguintes requisitos funcionais (RF):
 -
 - RF2: o sistema deve solicitar a informação necessária para inserir um item bibliográfico: título, autor, data, lugar, assunto, resumo, número, editor, periódico, congresso.
 - RF3: o sistema deve dar uma mensagem de alerta quando o usuário tentar inserir um item incompleto. Essa mensagem deve questionar o usuário se ele deseja cancelar a operação, completar a informação ou concluir a inserção como está.
 -

Qual informação é necessária para possibilitar uma inserção incompleta?

Exemplo: Omissão

- **Omissão de Desempenho:**

- Informação que descreva um desempenho desejado para o sistema foi omitida ou descrita de uma forma não apropriada para que possa ser verificada posteriormente no teste de aceitação.
- Ex: considere o seguinte Requisito Não Funcional (RNF):
 - RNF1: o sistema deve fornecer os resultados tão rápido quanto possível

?

Exemplo: Omissão

Outros tipos de omissão:

- **Omissão de Interface:**

- Quando informação que descreva como sistema proposto vai fazer interface e se comunicar com outros objetos fora de seu escopo for omitida do DR.

- **Omissão de Recursos do Ambiente:**

- Quando informação que descreva o hardware, software, base de dados ou detalhes do ambiente operacional no qual o sistema vai rodar for omitida do DR.

Exemplo: Fato Incorreto

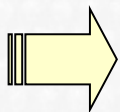
- Informação que consta do artefato mas que seja contraditória com o conhecimento que se tem do domínio da aplicação.

- Ex: considere um Sistema de Empréstimo numa Biblioteca e o seguinte RF:

-


- RF30: o sistema não deve aceitar devolução de livros se o usuário não tiver a carteirinha da biblioteca no momento.

- ...



para devolução de livros não é necessário apresentar a carteirinha
pois todas as informações estão registradas no sistema !

Exemplo: Inconsistência

- Informação que consta do artefato mais de uma vez e, em cada ocorrência, ela é descrita de forma diferente.
 - Ex: considere um Sistema de Empréstimo numa Biblioteca e o seguinte RF:
 -
 - FR5: o sistema não deve permitir períodos de empréstimo maiores que 15 dias.
 -
 - FR9: professores podem emprestar livros por um período de 3 semanas.
 -
- 

Exemplo: Ambigüidade

- quando a informação pode levar a múltiplas interpretações.
- Ex: considere um Sistema de Empréstimo numa Biblioteca e o seguinte RF:
 -
 - FR20: se o número de dias que o usuário está em atraso é menor que uma semana, ele deve pagar uma taxa de R\$1,00; se o número é maior que uma semana, a taxa é de R\$0,50 por dia.



qual a taxa a ser paga se o período for de uma semana?

no primeiro caso, a taxa deve ser calculada por dia?

Exemplo: Informação Estranha

- qualquer informação que, embora relacionada ao domínio, não é necessária para o sistema em questão.
- Ex: considere um Sistema de Empréstimo numa Biblioteca e o seguinte RF:
 -
 - RF15: quando um novo livro é adicionado ao acervo, ele permanece em uma prateleira especial por um período de um mês.
 -

essa informação não é necessária para o sistema



Técnicas de Leitura para Inspeção

- **Questão: Como detectar defeitos?**
- **Resposta:**
 - lendo o documento
 - entendendo o que o documento descreve
 - verificando as propriedades de qualidade requeridas

Técnicas de Leitura para Inspeção

- **Problema:**



em geral não se sabe como fazer a leitura de um documento !

- **Razão:**

- em geral, os desenvolvedores aprender a escrever documento de requisitos, código, projeto, mas não aprendem fazer uma leitura adequada dos mesmos.

- **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 influência humana nos resultados da inspeção.

Técnicas de Leitura para Inspeção

- **O que é uma técnica de leitura?**
 - é um conjunto de instruções fornecido ao revisor dizendo como ler e o que procurar no produto de software.
- **Técnicas de leitura para detecção de defeitos em Documentos de Requisitos:**
 - Ad-hoc
 - Checklist
 - Leitura Baseada em Perspectiva

- Os revisores não utilizam nenhuma técnica sistemática de leitura.
- Cada revisor adota sua maneira de “ler” o Documento de Requisitos
- Desvantagens:
 - depende da experiência do revisor
 - não é repetível
 - não é passível de melhoria pois não existe um procedimento a ser seguido.

Checklist

Checklist

- **Definição:** é uma técnica que fornece diretrizes para ajudar o revisor alcançar os objetivos de uma atividade de revisão formal que são:
 - verificar se o software está de acordo com os seus requisitos.
 - assegurar que o software está representado de acordo com padrões pré definidos.
 - cobrir erros de função, de lógica, de implementação em qualquer representação (artefato) de software.

Checklist

- É 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.

Checklist

- Pode ser desenvolvido para documentos de requisitos, análise, projeto, código e mesmo documentos de teste.



O projeto foi traduzido de forma apropriada para o código?

A linguagem foi utilizada de forma correta?

Existem comentários incorretos ou ambíguos?

As declarações e os tipos de dados estão corretos?

.....

Exemplo de Checklist
para a fase de código

Checklist

▪ Questões Gerais:

- Os objetivos do sistema foram definidos?
- Os requisitos estão claros e não ambíguos?
- Foi fornecida uma visão geral da funcionalidade do sistema?
- Foi fornecida uma visão geral das formas de operação do sistema?
- O software e o hardware necessários foram especificados?
- Se existe alguma suposição que afete a implementação ela foi declarada?
- Para cada função, os requisitos foram especificados em termos de entrada, processamento e saída?
- Todas as funções, dispositivos e restrições estão relacionadas aos objetivos do sistema e vice-versa?

Checklist

■ Omissão

de Funcionalidade

- As funções descritas são suficientes para alcançar os objetivos do sistema?
- As entradas declaradas para as funções são suficientes para que elas sejam executadas?
- Foram considerados os eventos indesejáveis e as respostas a eles foram especificadas?
- Foram considerados o estado inicial e os estados especiais (por ex. inicialização do sistema, término anormal)?

de Desempenho

- O sistema pode ser testado, analisado ou inspecionado para mostrar que ele satisfaz seus requisitos?
- Os tipos de dados, unidades, limites e resolução foram especificados?
- A freqüência e volume de entrada e saída foram especificados para cada função?

Checklist

■ **Omissão**

de Interface

- As entradas e saídas para todas as interfaces são suficientes?
- Foram especificados os requisitos de interface entre hardware, software, pessoas e procedimentos?

de Recursos do Ambiente

- Foram especificadas de forma apropriada as funcionalidades de interação entre hardware, software com o sistema?

■ **Informação Estranha**

- Todas as funções especificadas são necessárias para alcançar os objetivos do sistema?
- As entradas das funções são necessárias para executá-las?
- As entradas e saídas das interfaces são necessárias?
- As saídas produzidas por uma função são usadas por outra função ou transferidas para a interface externa?

Checklist

- **Ambigüidade**

- Cada requisito foi especificado de forma discreta, não ambígua e testável?
- Todas as transições do sistema foram especificadas de forma determinística?

- **Inconsistência**

- Os requisitos estão consistentes entre si?

- **Fato Incorreto**

- As funções especificadas são coerentes com o sistema e com os objetivos a serem alcançados?

Leitura Baseada em Perspectiva

Leitura Baseada em Perspectiva

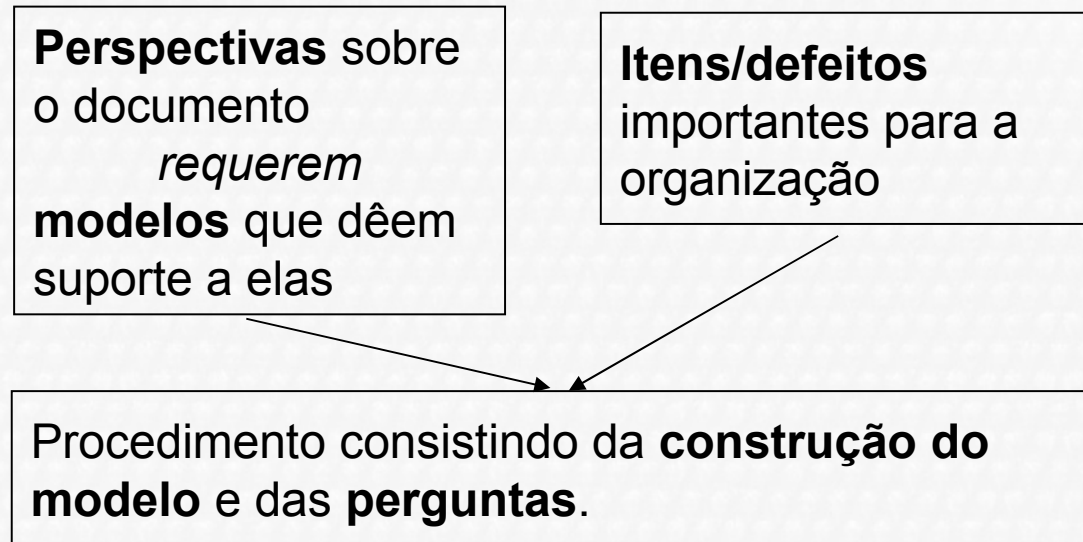
- Definição: é um conjunto de técnicas de leitura que focam em determinados pontos de vista.
- Fazem com que cada revisor se torne responsável por uma perspectiva em particular.
- Possibilita que o revisor melhore sua experiência em diferentes aspectos do documento de requisitos.
- Assegura que perspectivas importantes sejam contempladas.

Leitura Baseada em Perspectiva

- Cada revisor possui um “cenário” para guiar seu trabalho de revisão.
- Todo “cenário” consiste de duas partes:
 - Construir um *modelo* do documento que está sob revisão a fim de aumentar o entendimento sobre o mesmo.
 - Responder *questões* sobre o modelo, tendo como foco itens e problemas de interesse da organização.

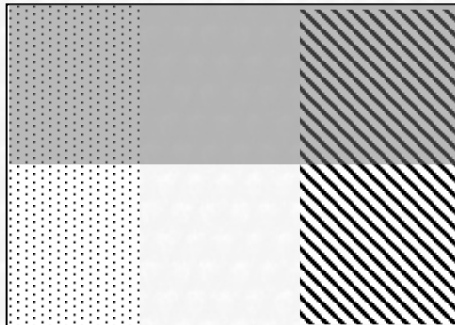
Leitura Baseada em Perspectiva

- Como é composto o “cenário”

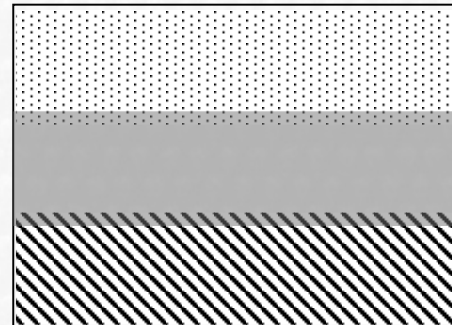


Leitura Baseada em Perspectiva

- Cada revisor vai ler o Documento de Requisitos com olhos diferentes
- Benefícios:
 - determina uma responsabilidade específica para cada revisor.
 - melhora a cobertura de defeitos.



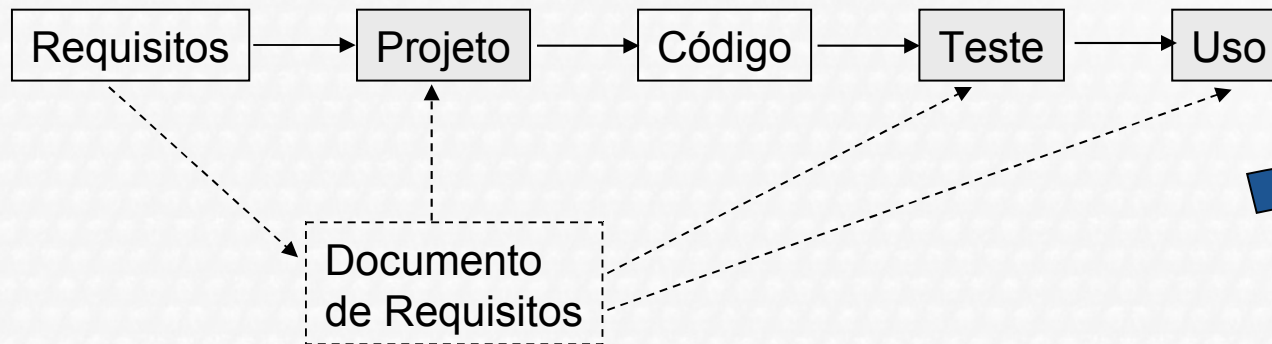
cobertura ad-hoc



cobertura baseada em perspectiva

Leitura Baseada em Perspectiva

- Considerando o Documento de Requisitos (DR), quais as leituras interessantes?



- o projetista que usa o DR para gerar o projeto do sistema.
- o testador que, com base no DR deve gerar casos de teste para testar o sistema quando este estiver implementado.
- o usuário para verificar se o DR está capturando toda funcionalidade que ele deseja para o sistema.

Leitura Baseada em Perspectiva - Visão do Usuário

- definir um conjunto de funções que o usuário esteja apto a executar.
- definir o conjunto de entradas necessárias para executar cada função e o conjunto de saídas que são geradas por cada função.
- isso pode ser feito escrevendo todos os cenários operacionais que o sistema deve executar.
- iniciar com os cenários mais óbvios até chegar nos menos comuns ou condições especiais.
- ao fazer isso, faça a você mesmo as seguintes perguntas:
- sugestão: usar como modelo subjacente Casos de Uso

Leitura Baseada em Perspectiva - Visão do Usuário

■ Questões:

- todas as funções necessárias para escrever os cenários estão especificadas no documento de requisitos ou na especificação funcional?
- as condições iniciais para inicializar os cenários estão claras e corretas?
- as interfaces entre as funções estão bem definidas e compatíveis (por ex., as entradas de uma função têm ligação com as saídas da função anterior?)
- você consegue chegar num estado do sistema que deve ser evitado (por ex., por razões de segurança)?
- os cenários podem fornecer diferentes respostas dependendo de como a especificação é interpretada?
- a especificação funcional faz sentido de acordo com o que você conhece sobre essa aplicação ou sobre o que foi especificado em uma descrição geral?

Leitura Baseada em Perspectiva - Visão do Testador

- para cada especificação funcional ou requisito gere um ou um conjunto de casos de teste que faça com que você se assegure de que a implementação do sistema satisfaz a especificação funcional ou o requisito .
- use a sua abordagem de teste normal e adicione critérios de teste.
- ao fazer isso, faça a você mesmo as seguintes perguntas para cada teste:
- sugestão: usar como critérios de teste Particionamento de Equivalência, Análise do Valor Limite

Leitura Baseada em Perspectiva - Visão do Testador

- **Questões:**

- você tem toda informação necessária para identificar o item a ser testado e o critério de teste? Você pode gerar um bom caso de teste para cada item, baseando-se no critério?
- você tem certeza de que os teste gerados fornecerão os valores corretos nas unidades corretas?
- existe uma outra interpretação dos requisitos de forma que o implementador possa estar se baseando nela?
- existe um outro requisito para o qual você poderia gerar um caso de teste similar, mas que poderia levar a um resultado contraditório?
- a especificação funcional ou de requisitos faz sentido de acordo com aquilo que você conhece sobre a aplicação ou a partir daquilo que está descrito na especificação geral?

Lista de Defeitos

Nro Sequencial	Local no Doc. Requisitos	Tipo do Defeito	Descrição
1	RF5	O	Não discriminadas as informações necessárias para que seja feito o cadastro da pessoa.
2	RF12	A	Não fica claro qual a taxa que deve ser paga, no caso de atraso de livro

de acordo com a taxonomia de erros

nro da seção ou do requisito no doc. de requisitos

uma explicação que dê para entender porque o inspetor considera que aquilo seja um defeito

Bibliografia

- (Basili et. al., 86) **Basili, V.; Selby, Richard W.; Hutchens, David H.** Experimentation in Software Engineering. *IEEE Transactions on Software Engineering*. n. 7, vol. SE-12 (1986), 733-743.
- (Basili & Selby, 87) **Basili, V.; Selby, Richard W.** Comparing the Effectiveness of Software Testing Strategies. *IEEE Transactions on Software Engineering*. n. 12, vol. SE-13 (1987), 1278-1296.
- (Basili et. al., 96) **Basili, V.; Green, S.; Laitenberger, O.; Lanubile, F.; Shull, F.; Sorumgard, S.; Zelkowitz, M.** The Empirical Investigation of Perspective-Based Reading. *Empirical Software Engineering: An International Journal*. n. 2, vol. 1 (1996), 133-164.
- (Kamsties & Lott, 95) **Kamsties, E.; Lott, C. M.** An empirical evaluation of three defect-detection techniques. Technical Report ISERN-95-02, Universität Kaiserslautern, Postfach 3049, D-67653 Kaiserslautern, 1995.
- www.cs.umd.edu/~mvz/mswe609/shull.pdf