Revisão de Software

Profa. Natasha Valentim
natasha@inf.ufpr.br

Baseado em materiais gentilmente cedidos por:
Tayana Conte (UFAM)
Guilherme Horta Travassos (COPPE-UFRJ)
Anna Beatriz (USES)
Adriana Lopes Damian (USES)

Motivação

- Identificar a presença de defeitos o mais cedo possível no processo de desenvolvimento:
  - Quanto **mais cedo** se descobre um defeito, **menor o custo** de correção e **maior a probabilidade de corrigi-lo** corretamente.
- Minimizar a ocorrência de erros e riscos associados;
- Prover ganhos significativos em relação a prazos e custos
  - Tendem a encontrar **mais defeitos a um custo menor.**
- Possuir um **baixo custo.**

Defeitos no Desenvolvimento de Software

- Defeito -> Erro - > Falha
  - **Defeito**: **deficiência mecânica ou algorítmica** que se ativada pode levar a uma falha;
  - **Erro**: item de **informação** ou **estado** de execução **inconsistente;**
  - **Falha**: **evento notável** onde o sistema **viola** suas **especificações**.
- Considerações:
  - Nem sempre **encontrar a falha conduz diretamente ao defeito**!
  - Testes revelam falhas, porém o que devem ser corrigidos são os defeitos.

Defeitos no Desenvolvimento de Software

- Principal causa:
  - Tradução incorreta de informações.
- Características:
  - Maior parte é de **origem humana;**
  - São gerados na **comunicação e na transformação de informações;**
  - Permanecem **presentes nos diversos produtos de software** produzidos e liberados;
  - A maioria encontra-se em **partes** do produto de software **raramente utilizadas** e/ou **executadas.**

Defeitos no Desenvolvimento de Software

- Consequências de defeitos
  - **Estimativas** de esforço e prazo **deixam de fazer sentido;**
  - **Desperdício de recursos** (retrabalho pode chegar entre 40% a 50% do esforço de um projeto);
  - Produto final **não atende as necessidades** do usuário;
  - Corrigir defeitos após a entrega do produto pode ser **até 100x mais caro que corrigi-los nas primeiras fases** do desenvolvimento (em projetos menores);
  - **Tradução incorreta de informações.**

Técnicas para Identificação e Correção de defeitos

- Técnicas que podem ser aplicadas durante o processo de desenvolvimento:
  - Revisões de Projeto;
  - Revisões de Código;
  - Inspeções de Projeto;
  - Inspeções de Código;
  - Testes
    - Unidade, Integração e Sistema
    - Beta teste

Técnicas para Identificação e Correção de defeitos

- Técnicas que podem ser aplicadas durante o processo de desenvolvimento:

Qual a técnica mais eficiente?

- Unidade, Integração e Sistema
- Beta teste

Eficiência das técnicas para correção de defeitos

ATIVIDADE	EFICIÊNCIA
Revisões informais de projeto	25% a 40%
Inspeções formais de projeto	45% a 65%
Revisões informais de código	20% a 30%
Inspeções formais de código	45% a 70%
Teste de unidade	15% a 50%
Teste de integração	25% a 40%
Teste do sistema	25% a 55%
Beta-teste (< 10 clientes)	24% a 40%
Beta-teste (> 1000 clientes)	60% a 85%

Fonte: Capers Jones, Software defect-removal efficiency, IEEE Computer, 1996

Revisão de Software

- São aplicadas em várias etapas durante o processos de software e servem para revelar erros e defeitos que podem ser eliminados.
- As revisões “purificam” os artefatos da Engenharia de Software, até mesmo o modelo de requisitos e projeto, dados de teste e código.
- Tipos:
  - Revisões informais
  - Revisões formais

Revisão de Software

- **Revisões informais:**
  - Um teste de mesa simples de um artefato (com um colega),
  - Uma reunião informal (envolvendo mais de duas pessoas) com a finalidade de revisar um artefato ou,
  - Os aspectos orientados a revisões da programação em pares.

Revisão de Software

- **Revisões formais:**
  - Descobrir erros na função, lógica ou implementação para qualquer representação do software;
  - Verificar se o software que está sendo revisado atende aos requisitos;
  - Garantir que o software foi representado de acordo com os padrões predefinidos;
  - Obter software que seja desenvolvido de maneira uniforme;
  - Tornar os projetos mais gerenciáveis.

  Revisões formais é uma classe de revisões que incluem inspeções.

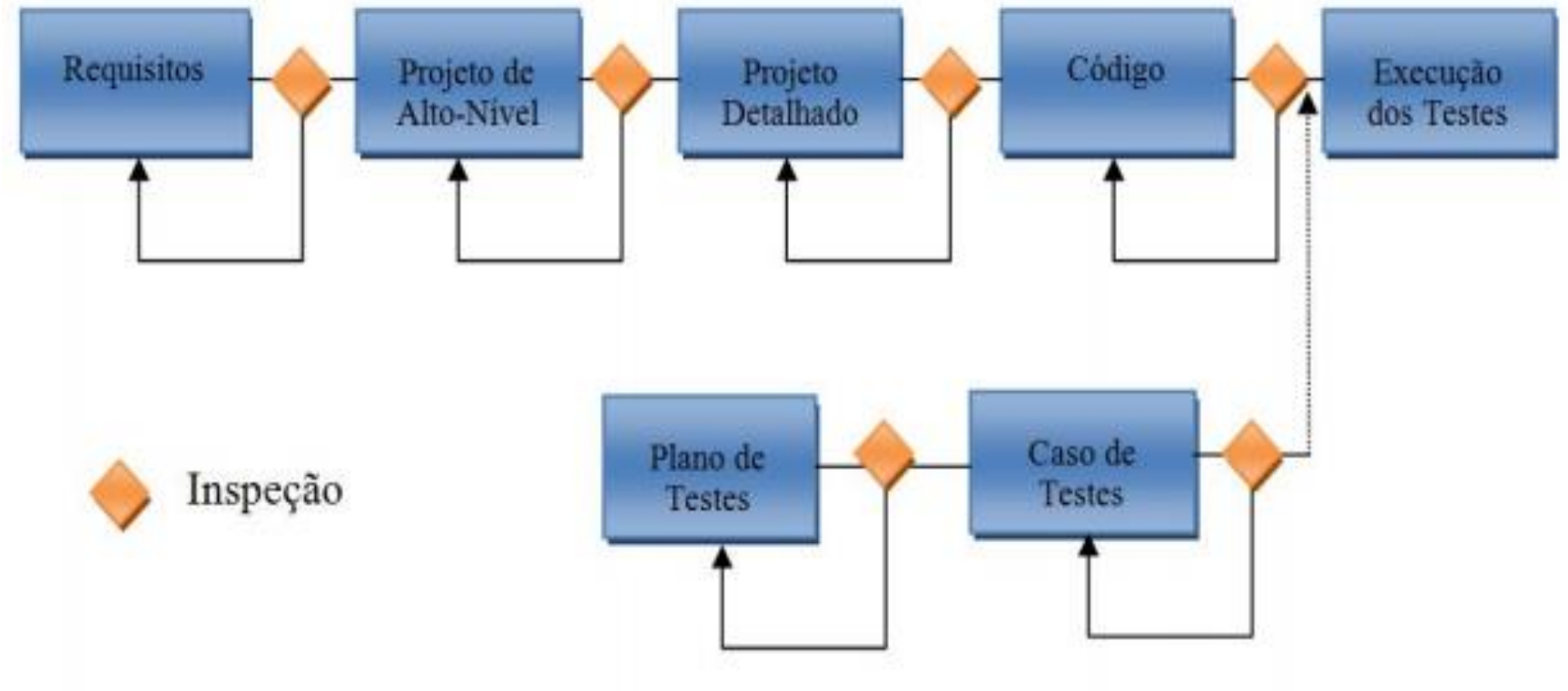
Inspeção de Software

A inspeção é um processo de revisão formal de software que tem por objetivo **a descoberta antecipada de defeitos e corrigi-los**, com um custo menor e eficiente no processo de desenvolvimento [Kalinowski *et al.*, 2004].

Inspeção de Software

- Em uma inspeção, a equipe de revisão **verifica se o artefato** está de acordo com uma lista de aspectos predeterminados;
- O processo de revisão é **dependente da técnica** que está sendo utilizada;
- Porém, se as atividades de inspeção **não forem feitas de forma adequada**, o verdadeiro potencial da **inspeção é prejudicado** [Ciolkowski et al., 2003].

Inspeção de Software

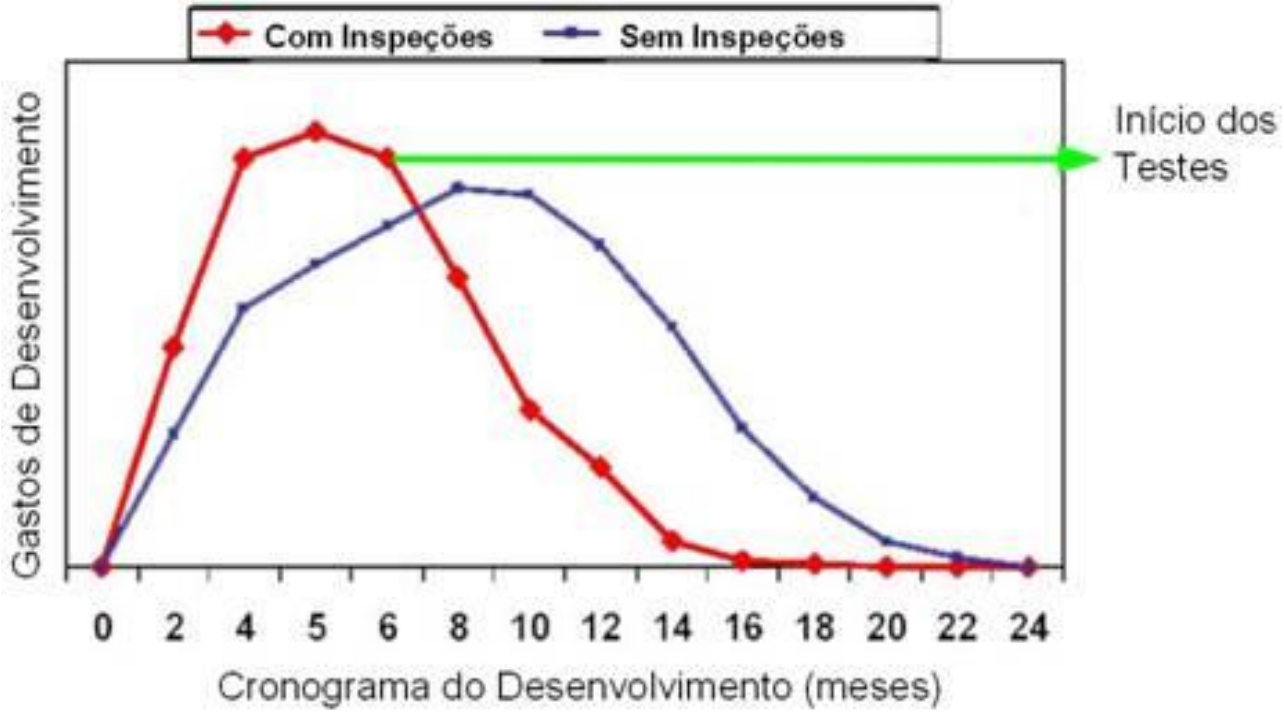


Inspeção em diferentes artefatos (adaptado de Kalinowski *et al.*, 2004).

Inspeção de Software

- Objetivos:
  - **Identificar erros específicos** num documento ou sistema
  - **Identificar erros sistemáticos** no processo de desenvolvimento;
  - **Identificar desvios** em relação à especificações e padrões
- Benefícios:
  - **Provê ganhos significativos** em relação a prazos e custos
  - Tende a **encontrar mais defeitos** que qualquer outro processo

Benefícios de Inspeções de Software



Gastos no desenvolvimento com e sem a utilização de inspeções [adaptado de Wheeler *et al.*, 1996].

Papéis da equipe de inspeção

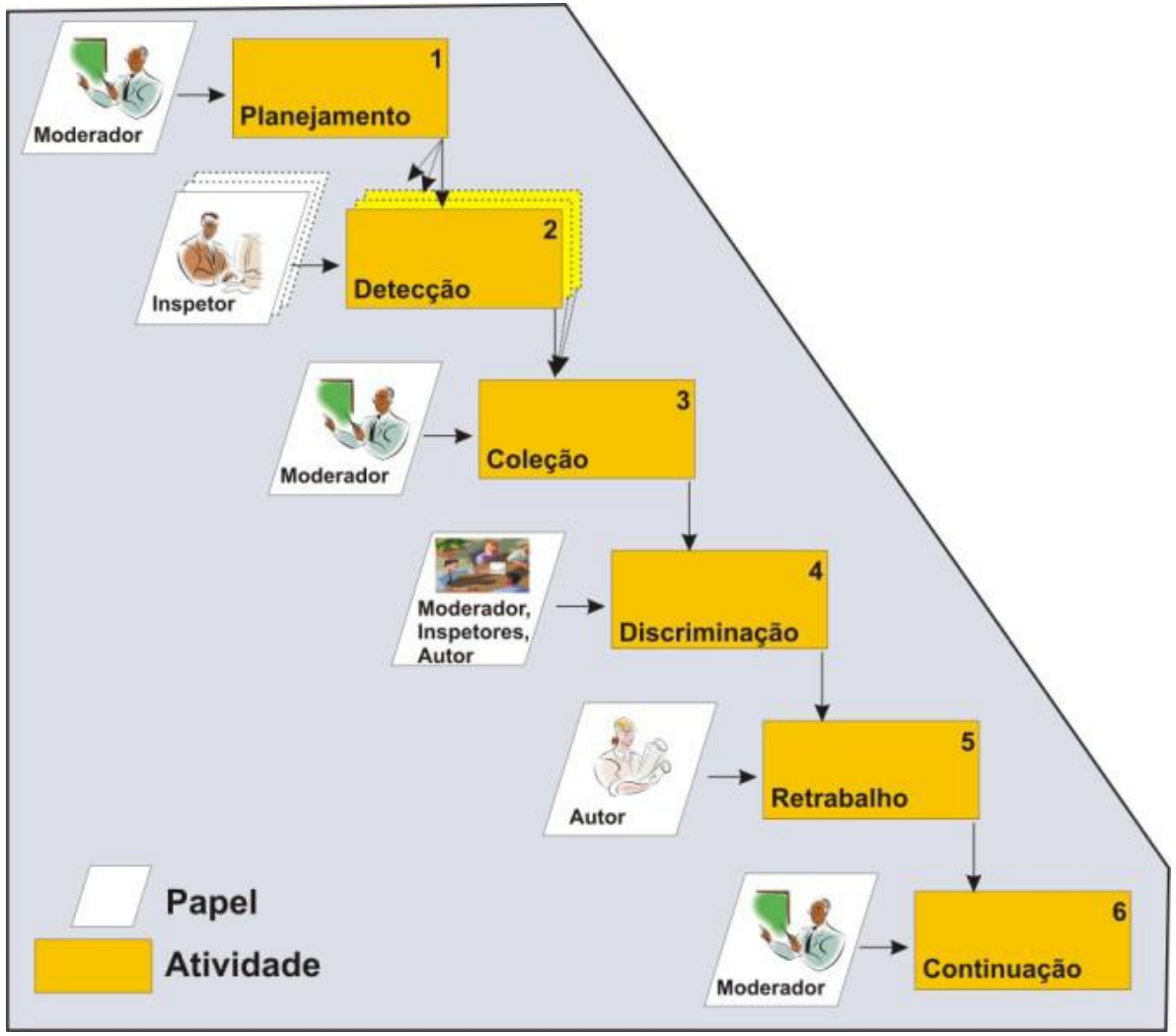
- Em uma inspeção, há a necessidade de três papéis:
  - **Autor**: **elabora o artefato** a ser inspecionado e **corrige os defeitos** identificados.
  - **Moderador**: **conduz** as atividades e **controla** as reuniões.
  - **Inspetor**: tem a função de **encontrar erros no documento** (artefato). Todos os participantes podem atuar como inspetores, além de suas atribuições /papéis.

Inspetor **Moderador** **Inspetor**

Autor

Inspetor

Reorganização do Processo de Inspeção por SAUER *et al.* (2000)



Processo de Inspeção

Processo de Inspeção

- Seleção do Escopo
- Seleção dos Inspetores
- Atribuição do que cada inspetor vai inspecionar
- Preparação da Infra-estrutura
- Treinamento na técnica (opcional)

Processo de Inspeção

Planejamento

Detecção

Coleção

Discriminação

Priorização

- Inspeção sendo realizada por cada inspetor (*ad-hoc*, checklist ou técnica de leitura)
- Envio da planilha com as discrepâncias para o responsável pela inspeção

Processo de Inspeção

Planejamento

Detecção

Coleção

Discriminação

Priorização

- O responsável pela inspeção avalia quais discrepâncias são repetidas (encontradas por mais de um inspetor)
- É gerada uma lista única de discrepâncias

Processo de Inspeção

Planejamento

Detecção

Coleção

Discriminação

Priorização

- Reunião com os inspetores, o gerente responsável pelo sistema e o responsável pela inspeção para classificar as discrepâncias encontradas em defeitos ou falso-positivos

Processo de Inspeção

Planejamento

Detecção

Coleção

Discriminação

Priorização

- Atividade opcional - Priorização para correção dos defeitos encontrados

Inspeção *Ad-Hoc*

- **Inspetor lê o documento** de acordo com sua perspectiva
- Experiência individual afeta o resultado final:
  - **Foco** reside **na especialidade** do inspetor;
  - **Produtividade individual**;
  - Difícil garantir que o inspetor **leu o documento de forma adequada**, pois cada um aplica sua "própria técnica".
- Custo/Eficiência (#defeitos/tempo)
  - Pode ser adequado quando **inspetores** possuem **alta experiência**.

Inspeção com *Checklists*

- Inspetor segue **uma lista de itens com características** a serem revisadas
  - Porém ainda aplica leitura *Ad-hoc* para identificar defeitos.
- Resultado final mais direcionado
  - **Características de qualidade** definidas à priori;
  - **Produtividade individual**;
  - **Cobertura do documento** relacionada aos itens do *Checklist.*
- Custo/Eficiência **depende do *Checklist* e dos inspetores**.

Técnicas de Leitura

- Fornecer um **conjunto de instruções a serem seguidas** pelos revisores para que os defeitos sejam detectados
  - Introduzir um formalismo que **garanta que o procedimento possa ser seguido e repetido**, Possibilitando a melhoria contínua do processo de revisão.
- As técnicas de leitura permitem:
  - **A análise individual** de um artefato de software
    - Ex.: Requisitos, projeto, código, planos de teste
  - **Alcançar a compreensão** necessária para uma tarefa em particular
    - Ex. Identificação de defeitos, reutilização, manutenção

Técnicas de Leitura

- Técnicas de leitura são:
  - Direcionadas a um objeto
    - Ex.: código, requisito, projeto.
  - Específicas ao documento e notação;
  - Ajustáveis para o projeto e o ambiente;
  - Definidas proceduralmente;
  - Focadas para fornecer uma cobertura particular do documento;
  - Experimentalmente verificadas em relação à sua eficácia.

Técnica DBR – *Defect Based Reading*

- **Defeitos** em artefatos de software **podem ser classificados** através de diversas taxonomias.
- Essas taxonomias auxiliam os inspetores na **identificação e categorização dos defeitos encontrados** durante a atividade de detecção.
- A DBR **baseia-se nos tipos de defeitos** que podem ser encontrados nos artefatos que são criados;

Técnica DBR – *Defect Based Reading*

Omissão

Informação necessária sobre o sistema foi omitida do artefato de software.

Fato incorreto

Alguma informação no artefato de software contradiz informação do documento de requisitos ou o conhecimento geral do domínio.

Inconsistência

Informação contida em uma parte do artefato de software está inconsistente com outra informação no artefato de software.

Informação estranha

Alguma informação que não é necessária ou nunca utilizada.

Ambiguidade

Informação contida no artefato de software é ambígua, ou seja, várias interpretações podem ser derivadas da definição, o que leva o desenvolvedor à implementação incorreta.

Técnica DBR – *Defect Based Reading*

Omissão

Informação necessária sobre o sistema foi omitida do artefato de software.

Fato incorreto

Alguma informação no artefato de software contradiz informação do documento de requisitos ou o conhecimento geral do domínio.

Inconsistência

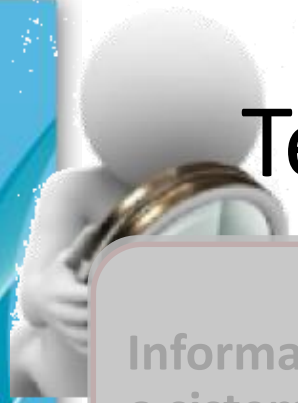
Informação contida em uma par
artefato de software está incons
outra informação no artefato de

Faltou descrever no caso de uso alguma informação ou funcionalidade que deveria ser atendida conforme a lista de requisitos

Ambiguidade

Informação contida no artefato de software é ambígua, ou seja, várias interpretações podem ser derivadas da definição, o que leva o desenvolvedor à implementação incorreta.

Técnica DBR – *Defect Based Reading*



Omissão
Informação necessária sobre o sistema foi omitida do artefato de software.

Fato incorreto
Alguma informação no artefato de software contradiz informação do documento de requisitos ou o conhecimento geral do domínio.

Algum texto descrito no caso de uso que contém informações erradas de um conceito descrito na lista de requisitos.

Informação estranha
Alguma informação que não é necessária ou nunca utilizada.

Ambiguidade
Informação contida no artefato de software é ambígua, ou seja, várias interpretações podem ser derivadas da definição, o que leva o desenvolvedor à implementação incorreta.

Técnica DBR – *Defect Based Reading*

Omissão
Informação necessária sobre o sistema foi omitida do artefato de software.

Fato incorreto
Alguma informação no artefato de software contradiz informação do documento de requisitos ou o conhecimento geral do domínio.

Inconsistência
Informação contida em uma parte do artefato de software está inconsistente com outra informação no artefato de software.

Informação estranha
Alguma informação que não é necessária ou nunca utilizada.

Informação contida no artefato
interpretações podem ser deri
desenvolvedor à implementaç

Algum texto descrito no caso de uso que está em desacordo com o requisito do caso de uso.

Técnica DBR – *Defect Based Reading*

Omissão
Informação necessária sobre o sistema foi omitida do artefato de software.

Fato incorreto
Alguma informação no artefato de software contradiz informação do documento de requisitos ou o conhecimento geral do domínio.

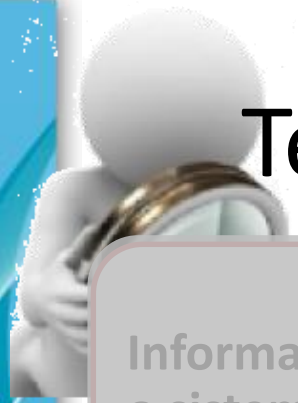
Inconsistência
Informação contida em uma parte do artefato de software está inconsistente com outra informação no artefato de software.

Informação estranha
Alguma informação que não é necessária ou nunca utilizada.

Foi descrito no caso de uso informações que não se aplicam ao mesmo e não deveriam ser incluídas.

é ambígua, ou seja, várias
ição, o que leva o

Técnica DBR – *Defect Based Reading*



Omissão

Informação necessária sobre o sistema foi omitida do artefato de s

Fato incorreto

Alguma informação no artefato de software contradiz informação do documento de to geral do

Algum texto descrito no caso de uso que não ficou claro e poderia causar má interpretação ou entendimento errado do que realmente significa por parte do usuário do documento.

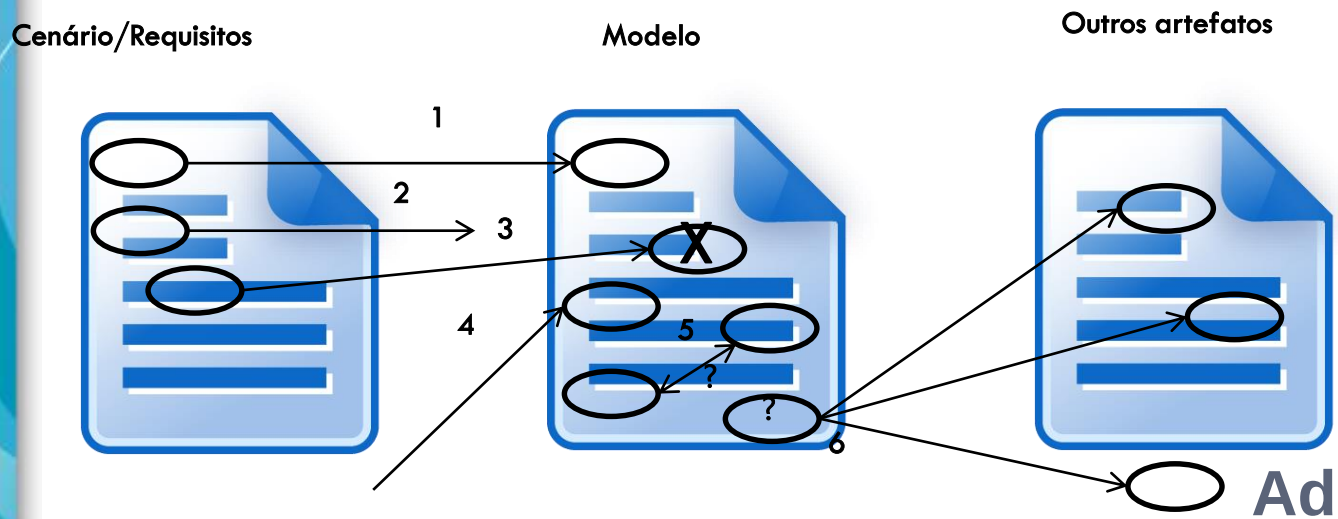
Informação d
artefato de software está inconsistente
outra informação no artefato de softwar

ção estranha
ormação que
ssária ou nunca
utilizada.

Ambiguidade

Informação contida no artefato de software é ambígua, ou seja, várias interpretações podem ser derivadas da definição, o que leva o desenvolvedor à implementação incorreta.

Técnica DBR – *Defect Based Reading*



Adaptado de Travassos et al. (1999)

1. **Informação transformada corretamente.**
2. **Omissão de informações necessárias. (Omissão).**
3. **Uso indevido dos elementos (Fato Incorreto).**
4. **Informação desnecessária incluída (Informação Estranha).**
5. **Conflito de informações nos elementos (Inconsistência).**
6. **Informação que não é bem definida, permitindo assim múltiplas interpretações. (Ambiguidade).**

Tipos de Defeitos

OMISSÃO:

- Deve-se a omissão de alguma informação necessária na especificação do caso de uso.

4. Fluxo de Eventos

4.1 Fluxo Principal- Manter

(FP1) O administrador realiza

(FP2) O sistema exibe a tela p[illegible] funcionalidades específicas do administrador [RN002].

(FP3) O administrador clica no botão Manter Cliente

(FP4) O sistema exibe a tela de Manter Cliente

(FP5) O administrador seleciona a opção desejada

(FP6) O caso de uso é finalizado.

Não foi especificado a regra de negócio RN002

???

Tipos de Defeitos

FATO INCORRETO:

- Deve-se à utilização de maneira incorreta das informações da especificação do caso de uso.

Não é um fluxo de exceção e sim alternativo.

4. Fluxo de Eventos

4.1 Fluxo Principal- Manter Cliente
(FP1) O cliente realiza login no sistema usando o CPF e a senha (FE01).
(FP2) O sistema exibe a tela principal com as funcionalidades específicas do administrador [RN002].

5. Fluxo de exceção

5.1 Acesso alternativo ao sistema
O cliente pode acessar o sistema usando sua digital

Tipos de Defeitos

INCONSISTÊNCIA:

- Informações contraditórias em especificação do caso de uso.

A regra de negócio mostra que a forma de acesso deve ser de uma forma e o fluxo apresenta outra .

4. Fluxo de Eventos

4.1 Fluxo Principal- Manter Cliente

(FP1) O administrador realiza login no sistema usando o CPF e senha

(FP2) O sistema exibe a tela principal com as funcionalidades específicas do administrador [RN002].

5. Regra de Negócio

5.1 Para acessar o sistema, o administrador deve realizar login usando o número de matrícula

Tipos de Defeitos

O sistema é sobre cadastro de aluno e está pedindo informações sobre alimento

INFORMAÇÃO ESTRANHA:

- Informação desnecessária incluída na e
  caso de uso.

3. Pré-Condições
3.1 O administrador já possui acesso ao sistema
4. Fluxo de Eventos
4.1 Fluxo Principal- Manter Cliente (FP1) O administrador realiza login no sistema usando o CPF e a senha (FP2) O sistema exibe a tela de cadastro de alunos [RN002]. (FP3) O administrador preenche as informações do alimento [RN003]

Tipos de Defe

Não há uma diferenças entre os botões que foram desabilitados e os que foram habilitados .

AMBIGUIDADE:

- Determinada informação não é bem definida na [illegible]cificação de caso de uso, permitindo múltiplas interpretações.

4. Fluxo de Eventos
4.2 Fluxo Alternativo - Cadastrar Cliente FA01.2 O sistema desabilita os botões Alterar, Inativar e Confirmar FA01.3 O sistema habilita os botões Incluir, Pesquisar, Cancelar e Voltar FA01.4 Todos os botões da tela são habilitados[RN002]
5. Regras de Negócio
RN001 Para acessar o sistema, o administrador deve realizar login usando o número de matrícula RN002. Exibir os campos de dados do Cliente (nome, endereço, telefone, CPF, e-mail, CNH, RG)

Comparação de diferentes técnicas de inspeção

- Abordagens para identificação de defeitos em documentos de requisitos
  - Ad-hoc, Checklists, Técnicas de leitura ajustadas (DBR – *Defect Based Reading*)

Técnica	**Notação**	**Sistemá -tica?**	**Focada?**	**Melhoria Controlada?**	**Ajustável?**	**Treinam. Neces.?**
Ad-hoc	qualquer	Não	Não	Não	Não	Não
Checklist	qualquer	Parcial	Não	Parcial	Sim	Parcial
DBR	Formal	Sim	Sim	Sim	Sim	Sim

Dicas para realizar uma boa inspeção

- Independente da abordagem utilizada, lembre-se:
  - **Revise o produto**, nunca o desenvolvedor;
  - **Defina uma agenda e cumpra-a** rigorosamente;
  - **Limite** o debate e a discussão;
  - **Mostre onde estão os problemas**, mas não tente resolvê-los durante a inspeção;
  - Não tente se lembrar, **faça anotações** sempre.

Dicas para realizar uma boa inspeção

- Prepare-se corretamente:
  - **Limite** o número de **participantes;**
  - **Motive a preparação** antecipada da inspeção;
  - **Desenvolva uma lista de tópicos** para cada produto que deverá ser revisado;
  - **Aloque recursos** e **tempo** para a inspeção;
  - **Forneça treinamento adequado** a todos os revisores;
  - **Aprimore-se**: revise suas revisões anteriores.

Bibliografia

- Boehm, Barry W.: Software Engineering Economics. Prentice Hall, 1981.
- Ciolkowski, M., Laitenberger, O., Biffl, S., 2003, “Software Reviews: The State of the Practice”, IEEE Software 20 (6): 46-51.
- Fagan, M.E.Design and Code Inspection to Reduce Errors in Program Development”, IBM Systems Journal, vol. 15, no. 3, pp. 182-211, 1979.
- Kalinowski, M., Spinola, R.O., Travassos, G.H., (2004). “Infra-Estrutura Computacional para Apoio ao Processo de Inspeção de Software”, III Simpósio Brasileiro de Qualidade de Software (SBQS 2004), Brasília.
- Lopes, Adriana Costa; MARQUES, Anna Beatriz; CONTE, Tayana. Avaliação do Jogo InspSoft: Um Jogo para o Ensino de Inspeção de Software. XII Simpósio Brasileiro de Qualidade de Software, 2013.
- Sauer, C., Jeffery, D.R., Land, L., Yetton, P., (2000). “The Effectiveness of Software Development Technical Review: A Behaviorally Motivated Program of Research”, IEEE Transactions on Software Engineering, 26 (1): 1-14, January.
- Travassos, G., Shull, F., Fredericks, M., Basili, V. Detecting Defects in Object Oriented Designs: Using Reading Techniques to Increase Software Quality. Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA), Denver, Colorado, 1999.
- Travassos, G. H., 2003, “Técnicas para Revisão e Inspeção de Software”, Mini-curso SBQS 2003;