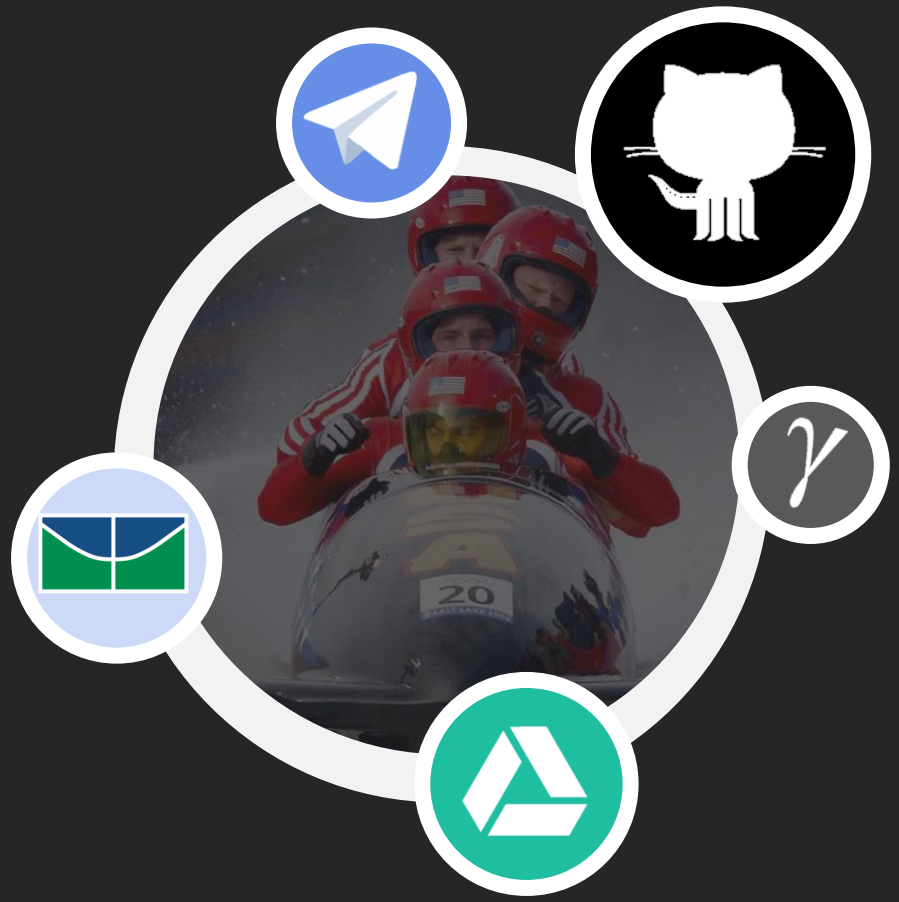
Inspeção de software

Técnica de verificação de Validação



Equipe

- Gabriel Araújo
- Jonathan Moraes
- Jonathan Rufino
- Laércio
- Luis Filipe
- Pedro Sales
- Phelipe Wener
- Rafael Rabetti
- Tâmara Barbosa

“**INSPEÇÕES** representam um tipo de revisões formais por **PARES**, ou peer **REVIEWS**, as quais são técnicas de análise para avaliação de **FORMA**, **ESTRUTURA** e **CONTEÚDO** de um documento, código fonte ou outro produto de trabalho.”

(Wieger, 2002)

OBJETIVO

- Identificação e Remoção de Defeitos

ALVO

- Artefatos de *Software*, Código Fonte, Projeto

SAÍDA

- Lista de Defeitos com Categorização Padronizada

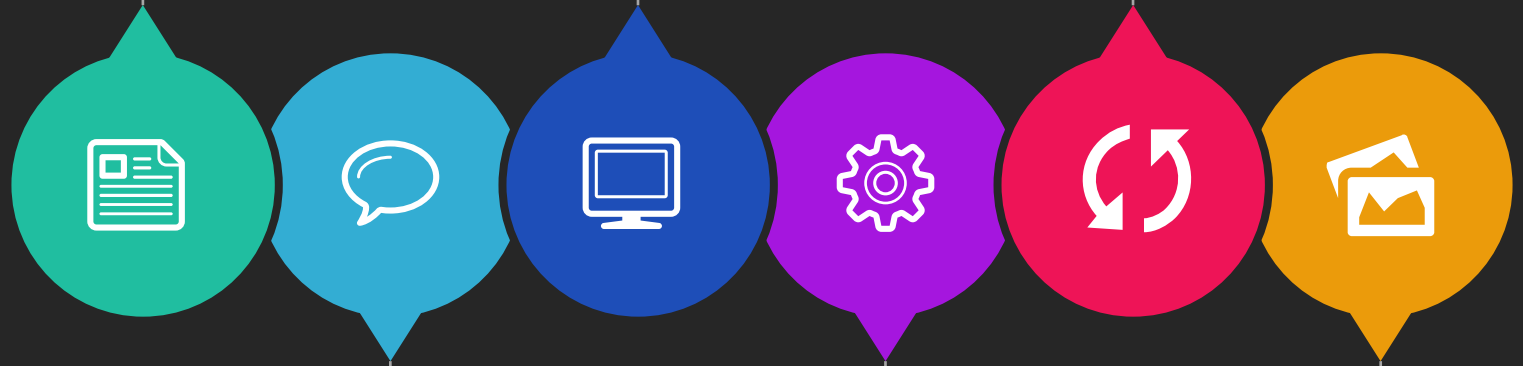
ATRIBUTOS	REVISÃO TÉCNICA	INSPEÇÃO	REVISÃO DE APRESENTAÇÃO
Objetivo	Avaliar conformidade, e verificar as alterações	Detectar e identificar efeitos, acompanhar a resolução	Detectar defeitos, avaliar alternativas, apresentar resultados
Método de Decisão	Revisores ecomendam, gerentes agem ou se justificam	Revisores apontam efeitos, gerentes providenciam remoção	Autores e gerentes decidem sobre alterações
Verificação da Decisão	Acompanhamento pelo líder ou auditorias da qualidade	Acompanhamento pelo íder, auditorias da qualidade	Análise pela Gerência Executiva, aceitação pelo cliente
Tamanho da Equipe	5 - 8	5 - 8	Limites físicos
Equipe	Autores e revisores	Autores e revisores	Autores
Apresentador	Autores, líder	Autores, líder	Autores
Dados Coletados	Defeitos, esforço de preparação, revisão e correção	Defeitos, esforço de preparação, revisão e correção	Defeitos, preparação e correção

Características

Estática
Analisa o artefato e não a execução

Qualquer artefato
Pode ser realizada no processo, no produto ou na documentação

Cíclica
Pode ser realizada diversas vezes durante o processo



Informal
Em relação a outras técnicas de V&V

Semi-automatizada
Existe a possibilidade de automatizar alguns passos para baratear o processo

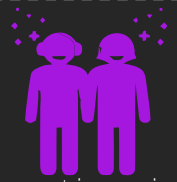
Diversos momentos
Pode encontrar defeitos mais cedo

TÉCNICAS DE LEITURA

AD HOC

Depende da sorte e/ou da experiência/esforço do inspetor

PERSPECTIVAS



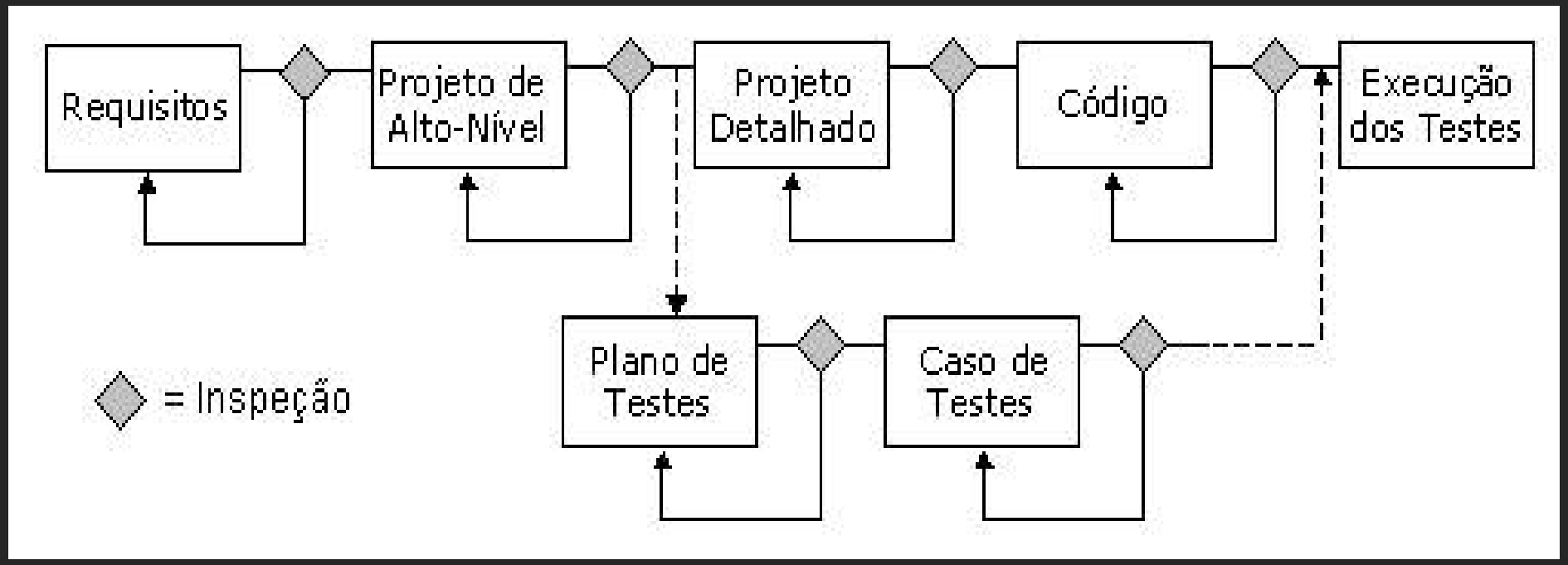
Uma perspectiva orienta o inspetor com respeito a como e onde procurar defeitos, concentra-se em certos aspectos e é diferente das demais.

CHECKLISTS

Os inspetores recebem uma lista de perguntas sobre assuntos do artefato a ser inspecionado

CENÁRIOS

Cada inspetor lê para um propósito específico, usando uma técnica especializada

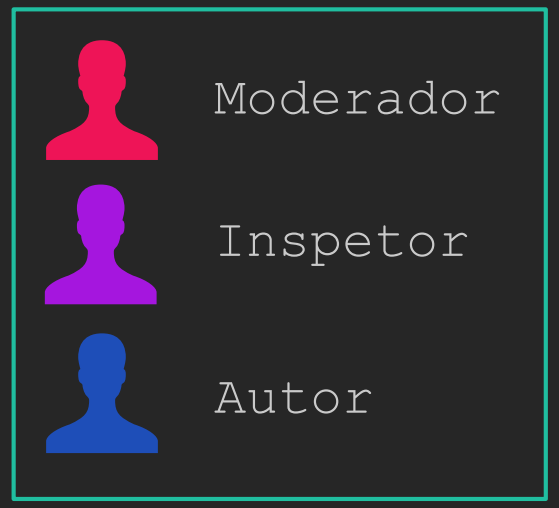


Inspeções de Software nos Diferentes Artefatos.
Adaptado de (ACKERMAN et al., 1989)

CLASSES DE DEFEITOS
(IEEE 830, 1998)

Omissão
Ambiguidade
Inconsistência
Fato Incorreto
Informação Estranha
Outros

Processo de Inspeção (Fagan, 1976)

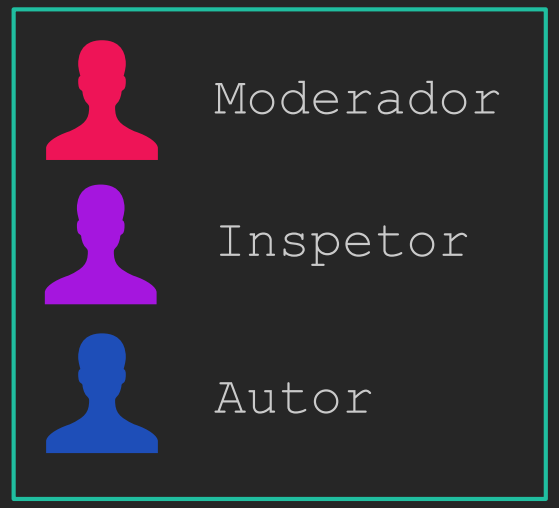


(Adaptado de FAGAN, 1976)

Reorganização do Processo

Baseados em diversos estudos experimentais sobre inspeções de software SAUER et al., (2000) propuseram uma reorganização do processo tradicional de inspeção. Essa reorganização visa a adequação do processo tradicional à inspeções com reuniões assíncronas e equipes geograficamente distribuídas. Assim, mudanças para reduzir o custo e o tempo total para a realização deste tipo de inspeção foram introduzidas.

Reorganização do Processo de Inspeção (SAUER et. al)



(Adaptado de FAGAN, 1976)

Reorganização do Processo

- **Autor** é o próprio desenvolvedor do artefato que será inspecionado;
- **Moderador** é quem lidera a inspeção e as reuniões;
- **Redator** é quem relata os defeitos encontrados e as soluções sugeridas durante a inspeção;
- **Inspetor** é o membro da equipe que tenta encontrar erros no produto.

ASPECTOS ABORDADOS

Inspeção de documentos de requisitos

Inspeção de código-fonte

Inspeção de documentos de processos

Tipos de defeitos encontrados em cada aspecto

DOCUMENTOS DE REQUISITOS E/OU PROCESSOS

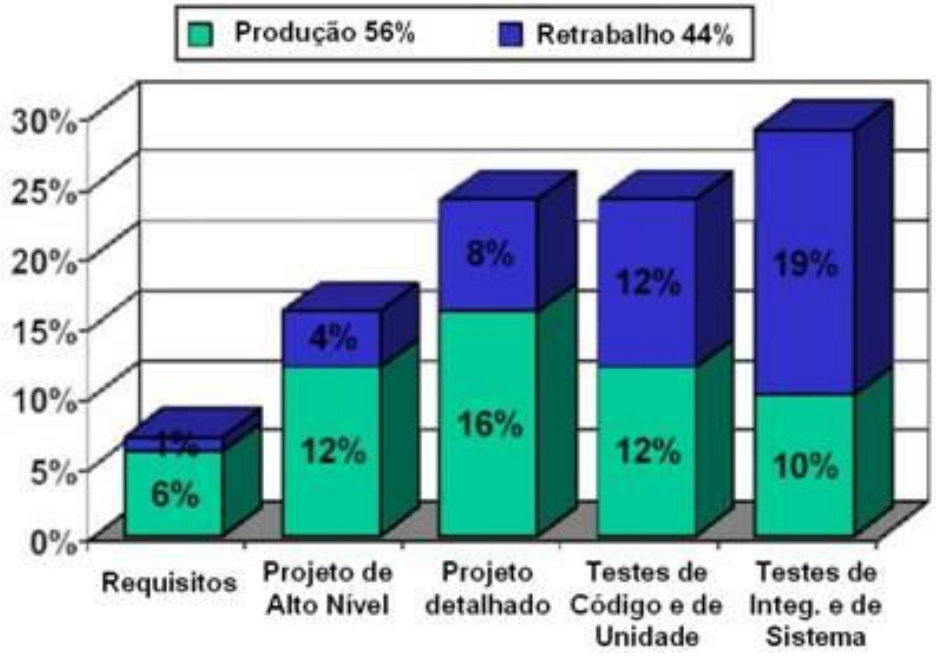
- ✓ Defeito de omissão
- ✓ Defeito de fato incorreto
- ✓ Defeito de inconsistência
- ✓ Defeito de ambigüidade
- ✓ Defeito de informação estranha

CÓDIGO FONTE

- ✓ Defeitos de Omissão
- ✓ Defeitos de Comissão
- ✓ Defeito de inicialização
- ✓ Defeitos de computação
- ✓ Defeito de controle
- ✓ Defeito de interface
- ✓ Defeitos de dados
- ✓ Defeitos cosmético

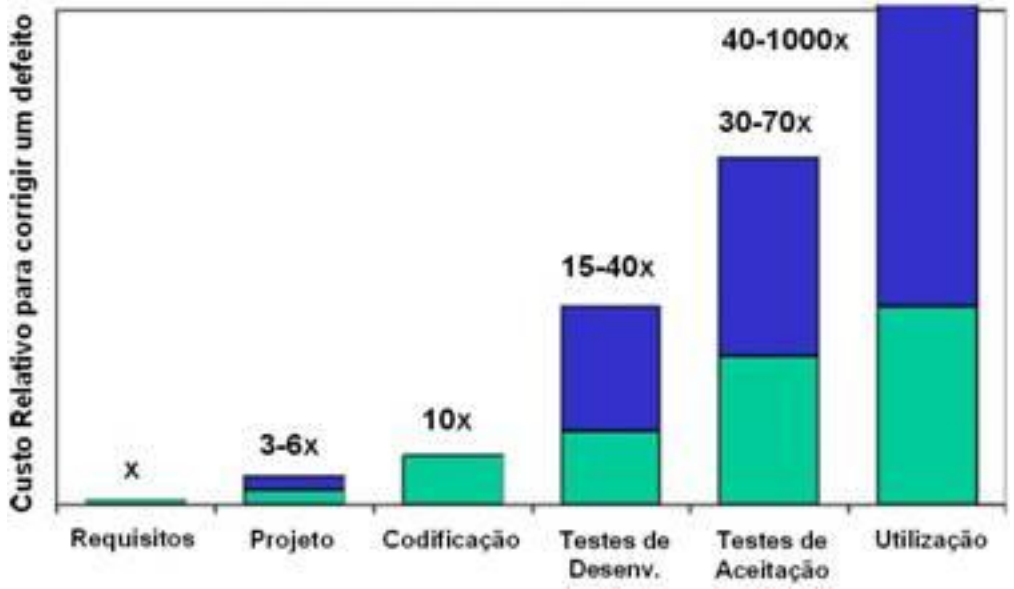
Benefícios da Aplicação de Inspeções de Software

O esforço gasto por organizações de software com **RETRABALHO** pode variar em média entre **40%** e **50%** do esforço total do desenvolvimento de um projeto. Uma estimativa da distribuição do retrabalho pelas atividades de desenvolvimento de software está ilustrada na figura abaixo:



Distribuição do retrabalho pelas atividades de desenvolvimento de software.
Adaptado de (WHEELER et al., 1996)

ATRAVÉS DA ANÁLISE DE 63 PROJETOS, BOEHM (1981) APRESENTA O CUSTO RELATIVO DA CORREÇÃO DE DEFEITOS ENCONTRADOS EM CADA UMA DAS ATIVIDADES DE DESENVOLVIMENTO:



Custo relativo para corrigir um defeito. Adaptado de (BOEHM, 1981).

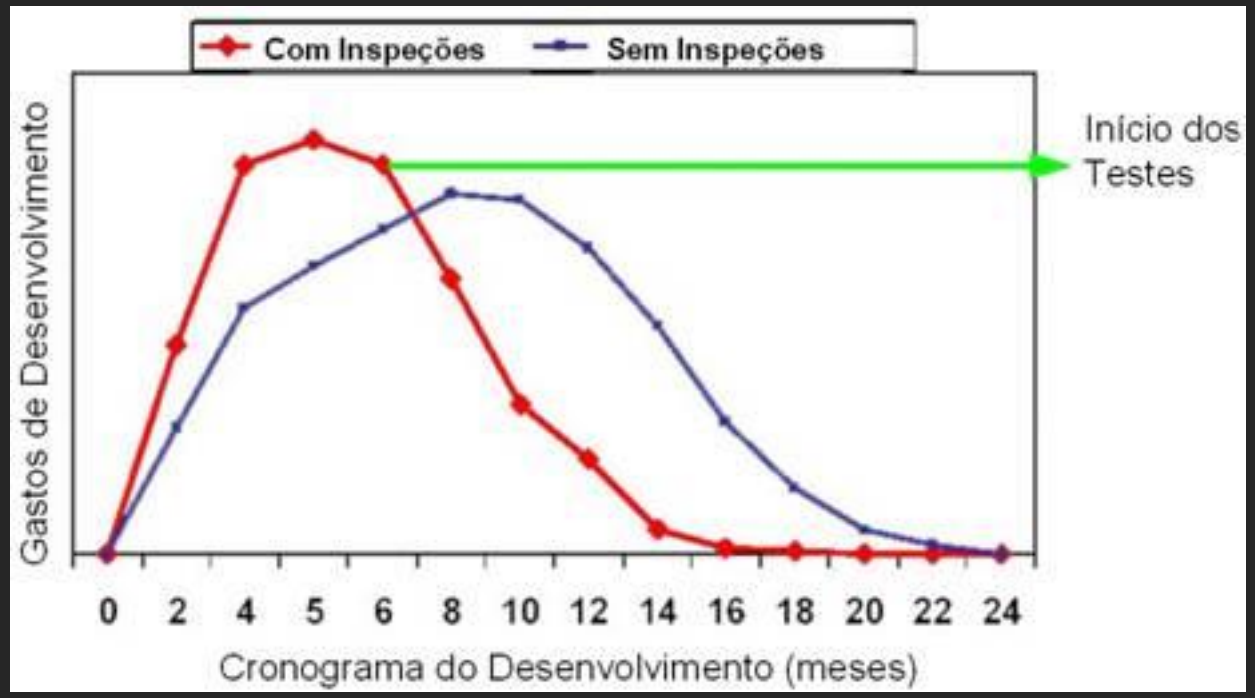
Outros resultados significativos

- **Esforço** o departamento de desenvolvimento da Ericsson em Oslo, Noruega, calculou uma redução bruta do esforço total de desenvolvimento em 20% aplicando inspeções. Além disso, resultados de estudos mostram que a introdução de inspeções de projeto pode reduzir o esforço com retrabalho em 44%.
- **Produtividade** de acordo com (GILB e GRAHAM, 1993), inspeções aumentam a produtividade de 30% a 50%;
- **Tempo** de acordo com (GILB e GRAHAM, 1993), inspeções reduzem o tempo de desenvolvimento de 10% a 30%;
- **Custo** resultados de estudos mostram que a introdução de inspeções de código pode reduzir os custos de implementação de projetos em 39%

Outros resultados significativos

- **Fagan (Faga, 1986)** Mais de 60% dos erros em um programa podem ser detectados usando-se inspeções informais de programa.
- **Mill et al. (Mills et al., 1987)** Uma abordagem mais formal para inspeção baseada em argumentos de correção pode detectar mais de 90% dos erros de um programa.
- **Selby e Basili (Selby, et al. 1987)** Compararam, empiricamente a eficiência das inspeções e dos testes. Eles constataram que a revisão estática de código era mais eficiente e custava menos do que teste de defeitos no descobrimento de defeitos de programas.
- **Gilb e Graham (Gilb e Graham, 1993)** constataram que isso era verdadeiro.

Estimativa dos gastos de desenvolvimento utilizando e não utilizando inspeções



Outros Benefícios

- Aprendizado
- Integração entre processos de detecção e de prevenção de defeitos
- Produtos mais inteligíveis
- Dados defeituosos ajudam a melhorar o processo de software do projeto

Parâmetros

- PROCESSO
- PESSOAS:
  - AUTOR
  - MODERADOR
  - REDATOR
  - INSPETOR

FERRAMENTAS

GRIP (GRoupware supported Inspection Process)

IBIS (Internet Based Inspection System)

ISPIS (Infra-estrutura de Suporte ao Processo de Inspeção de Software)

GRIP (GRoupware supported Inspection Process)

GRIP (Grünbacher et al., 2003). Nasceu da iniciativa de se adaptar um sistema COTS (Commercial Off-The-Shelf) de apoio a trabalho colaborativo (GSS) (Groupware Support System) para realizar inspeções em requisitos de software.

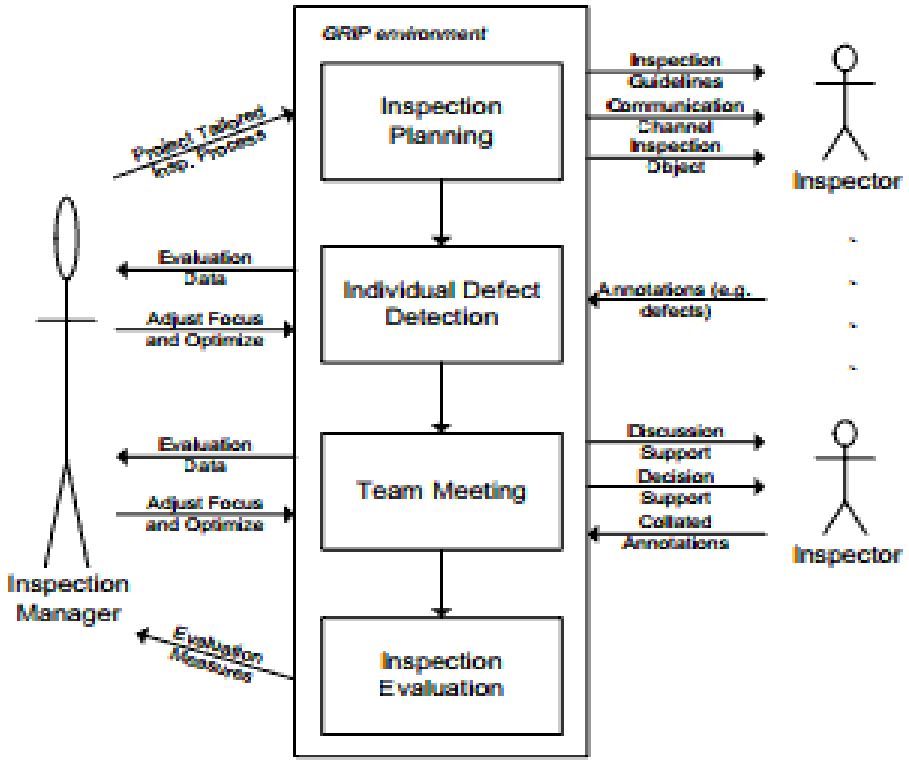


Figure 1: The GRIP framework [15].

Referências

- Kalinowski, Marcos. Introdução à Inspeção de Software. (2010) Engenharia de Software Magazine. 1ª ed.
- Grunbacher, et al. An Empirical Study on Groupware Support for Software Inspection Meetings. (2003)
- Melo, Silvana M. Inspeção de Software. (2010) Universidade de São Paulo (USP)
- Sommerville. Engenharia de Software. (2007) Addison Wesley, 8ª ed.
- Ferreira, Bruno. Uma Técnica para Validação de Processos de Desenvolvimento de Software. (2008)
- Kalinowski, Marcos. ISPIS: A Framework Supporting Software Inspection Proccesses. (2004)
- Vasconcelos, et al. Introdução à Engenharia de Software e à Qualidade de Software. (2006)
- Kalinowski, Marcos. Inspeções de Requisitos de Software em Desenvolvimento Incremental: Uma Experiência Prática. (2007) VI Simpósio Brasileiro de Qualidade de Software