Na sequência, um resumo do conteúdo apresentado no Artigo Engenharia de Software - Introdução à Inspeção de Software

Para obter uma classificação para os defeitos encontrados nas revisões (as faltas), partimos do fato de que todos os artefatos gerados durante o desenvolvimento de software utilizam como base o documento de requisitos ou artefatos gerados a partir deste. Desta forma, as classes de defeito seriam os tipos de defeito presentes em documentos de requisitos acrescidos dos tipos de defeitos introduzidos pela transformação de artefatos ao longo do desenvolvimento de software. Um padrão IEEE (IEEE 830, 1998), que recomenda práticas para especificação de requisitos de software, define atributos de qualidade que um documento de requisitos deve possuir. Foi considerado que a falta de qualquer um destes atributos constituiria um tipo de defeito. Assim, a seguinte taxonomia foi definida:

- **Omissão:** podendo ser em (1) algum requisito importante relacionado à funcionalidade, ao desempenho, às restrições de projeto, ao atributo, ou à interface externa não foi incluído; (2) não está definida a resposta do software para todas as possíveis situações de entrada de dados; (3) faltam seções na especificação de requisitos; (4) faltam referências de figuras, tabelas, e diagramas; (5) falta definição de termos e unidades de medidas.
- **Ambigüidade:** Um requisito tem várias interpretações devido a diferentes termos utilizados para uma mesma característica ou vários significados de um termo para um contexto em particular.
- **Inconsistência:** Dois ou mais requisitos são conflitantes.
- **Fato Incorreto:** Um requisito descreve um fato que não é verdadeiro, considerando as condições solicitadas para o sistema.
- **Informação Estranha:** As informações fornecidas no requisito não são necessárias ou mesmo usadas.
- **Outros**: Outros defeitos como a inclusão de um requisito numa seção errada do documento.

De acordo com Pressman (2001), qualidade de software é a conformidade a:

1. requisitos funcionais e não funcionais que têm sido explicitamente declarados;
2. padrões de desenvolvimento que tenham sido claramente documentados; e
3. características implicitamente esperadas de todo software a ser desenvolvido.

Conforme FAGAN (1976), o processo de inspeção de software é apresentado na Figura 1, e as fases são explicadas na sequência deste texto.

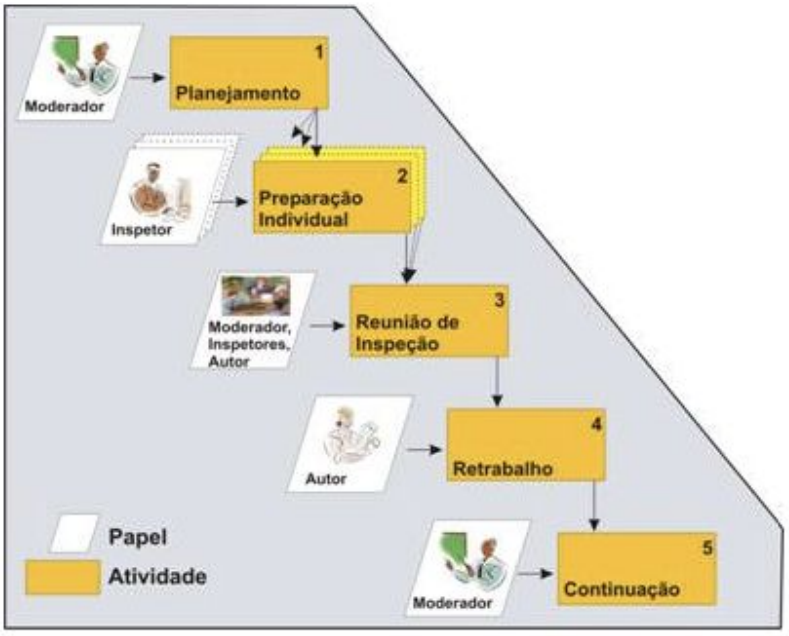

Figura 1 - Processo de inspeção de software

Fonte: (FAGAN, 1976, versão adaptada)

1. **Planejamento**: Seleciona-se a metodologia que será utilizada na detecção de defeitos, como documentos serão inspecionados, como a inspeção será descrita, quem serão os inspetores, quais partes do material serão inspecionados, entre outros.
2. **Apresentação**: É feita uma apresentação sobre o software e os artefatos a serem inspecionados, similar a um "briefing", pode ser omitida caso os inspetores já conheçam bem os artefatos.

3. **Preparação**: Os artefatos são estudados cautelosamente e são feitas anotações sobre as discrepâncias ou detalhes peculiares de cada um. São utilizadas técnicas de leitura para facilitar essa etapa.
4. **Reunião**: É feita uma reunião para discussão e classificação das discrepâncias encontradas, elas são então classificadas como defeitos ou falso positivos. É realizada pelo moderador, inspetores e autores dos documentos. Defeitos encontrados não tem sua solução discutida para que a reunião não seja demorada, essa costuma levar no máximo 2 horas, caso leve, deve ser dividida em vários dias.
5. **Retrabalho**: O autor do documento efetua as correções nos defeitos encontrados e confirmados pelos inspetores e o moderador, respectivamente.
6. **Continuação**: O moderador faz uma análise do material corrigido anteriormente e avalia a qualidade do artefato, ele então decide se uma nova inspeção é necessária ou não.

Entre as características deste processo, este pode ser aplicado à todos os artefatos produzidos ao longo do processo de desenvolvimento, permitindo a utilização de técnicas de leitura de artefatos específicos na atividade de preparação individual. Além disso, este processo possui uma estrutura rígida, com aspectos colaborativos, onde papéis, atividades e os relacionamentos entre atividades estão bem definidos.

SAUER et al. (2000) re-organizam o processo tradicional de inspeção proposto por (FAGAN, 1976). Esta re-organização visava a adequação do processo tradicional às inspeções com reuniões assíncronas e equipes geograficamente distribuídas. Assim, foram introduzidas mudanças para reduzir tanto o custo quanto o tempo total para a realização deste tipo de inspeção. Essa proposta manteve a estrutura rígida e os aspectos colaborativos do processo tradicional mas substituiu as atividades de preparação e de reunião do processo tradicional por três novas atividades denominadas por detecção de defeitos, coleção de defeitos e discriminação de defeitos:

1. **Detecção de Defeitos**: Cada um dos inspetores selecionados pelo moderador no planejamento realizará uma atividade de detecção de defeitos. A principal tarefa desta atividade consiste em buscar defeitos no documento a ser inspecionado e assim produzir uma lista de discrepâncias.
2. **Coleção de Defeitos**: O moderador agrupa as listas de discrepâncias dos inspetores. Esta atividade envolve eliminar discrepâncias repetidas (encontradas por mais de um inspetor), mantendo só um registro para cada discrepância.
3. **Discriminação de Defeitos**: O moderador, o autor do documento e os inspetores discutem as discrepâncias de forma assíncrona. Durante esta discussão, algumas discrepâncias serão classificadas como falso positivo e outras como defeito. Os falso positivos serão descartados e os defeitos serão registrados em uma lista de defeitos, que então será passada para o autor para que a correção possa ocorrer.

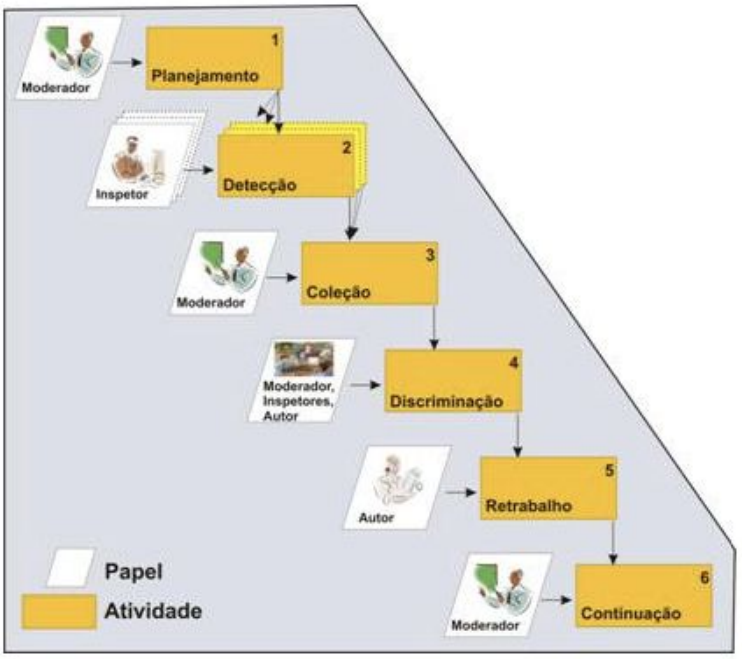

Figura 2 - Reorganização do processo de inspeção

Fonte: (SAUER et al., 2000, versão adaptada)

Este novo processo permitiu a utilização de um número grande de inspetores em paralelo para a detecção de defeitos e, assim, aumentar a probabilidade de se

encontrar defeitos difíceis de serem encontrados. Um grande número de inspetores pode provocar aumento no custo mas não aumenta o tempo de detecção de defeitos. Além disso, não provoca problemas de coordenação pois as discrepâncias encontradas por mais de um inspetor são encaminhadas diretamente para a atividade de retrabalho e as discrepâncias encontradas por apenas um inspetor não são discutidas por todos os inspetores.

Ao longo dos anos, muito conhecimento tem sido produzido na área de inspeções de software. Incluindo variantes do processo tradicional de inspeção, técnicas de estimativa do número de defeitos de documentos e da cobertura de inspeções, técnicas de leitura de documentos visando aumentar o número de defeitos encontrados por inspetores, diretrizes para pontos de tomada de decisão do processo de inspeção, dentre outras. Parte deste conhecimento tem sido avaliado em estudos experimentais e se mostrado adequado. No entanto, muito deste conhecimento não tem sido utilizado na prática por organizações de software ao realizarem suas inspeções, e é negligenciado na maioria das propostas de apoio ferramental ao processo de inspeção de software.

(CIOLKOWSKI et al., 2003) apresenta uma avaliação do estado da prática de revisões de software . De acordo com seus resultados, embora muitas organizações de software realizem revisões, a forma como as revisões são realizadas ainda é pouco sistematizada, pouco conhecimento da área de inspeções de software é utilizado e assim o verdadeiro potencial das revisões raramente é explorado. Este argumento é baseado em três observações sobre as organizações participantes do survey:

- Revisões raramente cobrem sistematicamente as fases de desenvolvimento de software;
- Muitas vezes a atividade de detecção de defeitos não é realizada sistematicamente;
- Organizações raramente utilizam revisões de software no contexto de um programa sistemático de avaliação e melhoria do processo.

O processo de inspeção é realizado por uma equipe composta por desenvolvedores e também por mais participantes, que realizam os seguintes papéis [Fagan 1986]:

- 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: membros da equipe que tentam encontrar erros no produto.

A inspeção pode ser feita tanto em produtos de software com projetos de software, depende do aspecto que será analisado durante a revisão. Podemos classificar dois tipos básicos de revisão de acordo com os aspectos analisados:

- Inspeção de documentos de requisitos: analisa documentos de requisitos encontrando erros enquanto eles são mais fáceis e baratos de serem corrigidos.
- Inspeção de código-fonte: Visa a encontrar defeitos no código-fonte, realizando uma análise estática do código. Tornam os programas menos complexos, pois os subprogramas são escritos em um estilo consistente e obedecem padrões estabelecidos, além disso o desenvolvimento de sistemas torna-se transparente, a estimativa e o planejamento tornam-se mais confiáveis e facilita a manutenção,com o desenvolvimento de sistemas mais compreensíveis e bem documentados.

Tipos de defeitos encontrados em cada aspecto A inspeção em documentos de requisitos pode revelar inúmeros defeitos, podemos classificá-los como segue:

- Defeito de omissão: informações necessárias ao sistema são omitidas, como exemplo a omissão de uma funcionalidade ou da capacidade de desempenho do sistema.

- Defeito de fato incorreto: há informações nos artefatos do sistema que são contraditórios com o domínio da aplicação.
- Defeito de inconsistência: a informação aparece mais de uma vez no artefato e de forma diferente em cada aparição causando incoerência.
- Defeito de ambigüidade: a informação leva a múltiplas interpretações.
- Defeito de informação estranha: uma informação que aparece no artefato e embora esteja relacionada ao domínio, não é necessária para o sistema em questão.

Os defeitos encontrados no código fonte podem ser classificados em:

- Defeitos de Omissão: causado pelo esquecimento de algum elemento no programa, como um comando que atribui valor a uma variável por exemplo.
- Defeitos de Comissão: um segmento de código incorreto, quando, por exemplo, um operador aritmético errado é usado em uma expressão.
- Defeito de inicialização: inicialização incorreta de uma estrutura de dados.
- Defeitos de computação: qualquer computação incorreta para a geração do valor de uma variável.
- Defeito de controle: causa a execução de um caminho de controle errado para um valor de entrada.
- Defeito de interface: quando um módulo usa ou faz suposições sobre dados que não fazem parte de seu escopo.
- Defeitos de dados: uso incorreto de uma estrutura de dados.
- Defeitos de cosmético: erros de ortografia e gramática.

Técnicas de leitura de artefato de software A inspeção é uma técnica de revisão baseada na leitura e entendimento de um documento a fim de encontrar defeitos. Porém um dos problemas enfrentados pelos desenvolvedores é que eles aprendem a escrever documentos de requisitos, código, projeto, mas não aprendem a fazer a leitura adequada dos mesmos. A solução é empregar técnicas de leitura, que são um conjunto concreto de instruções dadas ao leitor de como ler e o que

olhar em um produto de software. Leitura de software é uma análise individual de um produto textual de software que pode ser uma especificação de requisitos, código fonte planos de teste, com objetivo de obter entendimento necessário para realizar uma tarefa como detectar defeitos por exemplo. Existem diversas técnicas de leitura:

- Técnica de leitura Ad-Hoc - técnica que não utiliza nenhuma técnica formal de leitura, cada leitor lê o documento do seu modo, por este motivo ela torna-se dependente da experiência do leitor, e apresenta um grande defeito que é o fato de não ser repetível nem passível de melhoria, pois não existe um procedimento a ser seguido.
- Técnica de leitura Check-list - similar à técnica Ad-Hoc, porém cada revisor recebe um checklist, onde os itens deste checklist capturam importantes lições aprendidas em revisões anteriores. Itens individuais de um checklist podem enumerar defeitos característicos, priorizar diferentes defeitos ou propor questões para ajudar o revisor a descobrir defeitos.
- Técnica de leitura baseada em perspectiva (PBR) - técnica de leitura baseada em perspectiva é aplicada em inspeção de documentos de requisitos em linguagem natural. Ela prevê um conjunto de instruções específicas para os três papéis envolvidos diretamente com o documento de requisitos: o testador, o projetista e o usuário. A PBR define responsabilidades específicas para cada revisor, desse modo cada indivíduo é responsável por criar uma abstração do produto e então responder questões baseadas na análise da abstração a partir de uma perspectiva (ponto de vista) diferente. Em cada perspectiva é definido um cenário o e um conjunto de questões e atividades que dizem ao indivíduo o que ele deve fazer e como ele deve ler o documento, essas questões auxiliam o indivíduo a descobrir defeitos. A seguir, apresentam-se os cenários e questões, sob as perspectivas de leitura do usuário e do testador [Dória 2009]:
    - Perspectiva do usuário Espera-se que o revisor execute as seguintes tarefas: defina o conjunto de funções que um usuário esteja apto a executar; define um conjunto de entrada necessário para executar

cada função e o conjunto de saída gerado pela função. Isto pode ser visto como escrever todos os cenários operacionais ou subconjuntos de cenários operacionais que o sistema deveria executar. Iniciando com os cenários mais convencionais e prosseguindo para os cenários menos comuns ou condições especiais. Enquanto os cenários estão sendo gerados, o revisor faz a si mesmo as seguintes perguntas:

Todas as funções necessárias para escrever os cenários estão especificadas no documento de requisitos ou na especificação funcional? As condições 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?

- Perspectiva do testador - Tendo como perspectiva a visão de um testador, o revisor deve assegurar que para cada especificação funcional ou requisito gere um conjunto de casos de teste que assegure de que a implementação do sistema satisfaz a especificação funcional ou o requisito. É recomendável que ele use a sua abordagem de teste formal e critérios de teste. Após a construção do cenário ele deve fazer a si mesmo as perguntas a seguir para cada teste:

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 testes gerados fornecerão os valores corretos nas unidades corretas? Existe outra interpretação dos requisitos de forma que o programador possa estar se baseando nela? Existe 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?