Inspeção de software

Silvana M. Melo[1]

[1]Instituto de Computação e Matemática Computacional – Universidade de São Paulo (USP)
Caixa Postal 668 – 13560-970 – São Carlos – SP – Brazil

morita@icmc.usp.br

Abstract. *This article describes the methods and techniques used in the search for quality of software artifacts during all stages of its development, methods of verification and validation, verification and other static and dynamic, emphasizing the process of software inspection.*

Resumo. *Este artigo descreve os métodos e técnicas utilizadas na busca de qualidade dos artefatos de software, durante todas as fases de seu desenvolvimento, métodos de verificação e validação, verificação estática e dinâmica entre outros, dando ênfase ao processo de inspeção de software.*

1. Introdução

Dada a popularização de sistemas de software, e o fato deles se tornarem cada vez maiores e mais complexos, a garantia de qualidade nesses sistemas é um grande desafio. Uma forma de garantir a qualidade do produto é tratar de problemas o mais cedo possível, ou seja, assim que eles aparecem e não adiando até o final do desenvolvimento, pois quanto mais tarde o problema é descoberto maior é custo de sua correção.

Na tentativa de diminuir o retrabalho e melhorar a qualidade dos produtos, uma abordagem que tem se mostrado eficiente e de baixo custo para encontrar defeitos, reduzindo o retrabalho e melhorando a qualidade dos produtos é a revisão dos artefatos produzidos ao longo do processo de desenvolvimento de software. Inspeção de software é um tipo particular de revisão que pode ser aplicado a todos os artefatos de software e possui um processo de detecção de defeitos rigoroso e bem definido. A figura a seguir ilustra a possibilidade de realizar inspeções nos diferentes artefatos de software [Kalinowski 2004].

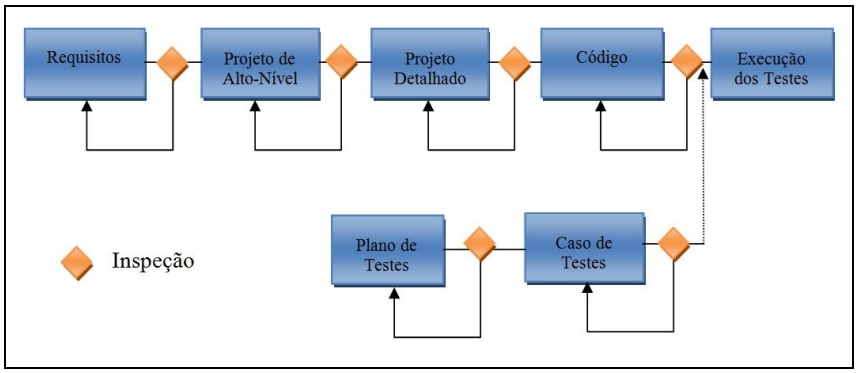

Figura 1. Inspeção em diferentes artefatos (adaptado de [Kalinowski 2004])

2. Qualidade de software

Qualidade de Software é um processo sistemático que focaliza todas as etapas e artefatos produzidos com o objetivo de garantir a conformidade de processos e produtos, prevenindo e eliminando defeitos. Defeitos são encontrados em todas as fases do desenvolvimento, porém é comprovado que a maioria encontra-se nas fases iniciais do processo de desenvolvimento. Isso gera prejuízos enormes para as empresas desenvolvedoras de software, pois o preço para corrigir um erro cresce muito com o passar do tempo. Com isso, fica clara a importância de um processo de Garantia da Qualidade que atue em todas as fases do processo de desenvolvimento [Bartié 2002].

Para isso, o processo de Garantia da Qualidade utiliza-se de atividades de verificação, validação com o intuito de encontrar erros em todas as etapas do desenvolvimento. Essas atividades são muito importantes, pois cuidam de analisar diversos pontos do processo de desenvolvimento, impedindo que um erro se propague para as fases posteriores do projeto.

2.1 Verificação e Validação

Técnicas de verificação e validação são aplicadas aos softwares durante e depois de seu desenvolvimento para garantir que ele atente sua especificação e fornece as funcionalidades esperadas pelos clientes.

Essas atividades ocorrem durante todo o ciclo de vida do software começando com revisões de requisitos e continua ao longo das revisões de projeto e das inspeções de código até o teste do produto.

2.1.1 Análise dinâmica

Análise dinâmica de software é uma técnica de verificação e validação muito usada, que consiste em exercitar o programa usando dados reais processados pelo programa e verificar se as saídas obtidas estão de acordo com as saídas esperadas.

Uma técnica estática muito utilizada são os testes de software que são essenciais para descoberta de defeitos e garantia de qualidade e confiabilidade que só podem ser obtidas com a execução do programa.

2.1.2 Análise estática

São métodos usados para garantir a qualidade do software que não necessita de uma versão executável do programa. Por este motivo podem ser utilizadas em todas as fases do desenvolvimento do software, pode verificar tanto o produto quanto o processo de software.

Porém essa técnica oferece garantia somente a correspondência entre um programa e sua especificação (verificação), não demonstra que o software é útil operacionalmente, não pode testar propriedades emergentes do software como desempenho e confiabilidade [Sommerville 2007].

Dentre as técnicas de verificação, revisões são técnicas amplamente difundidas e que podem ser usadas tanto na análise estática quanto dinâmica.

3. Revisões

Revisões fazem parte do conjunto de atividades de garantia de qualidade de software. Essas atividades constituem um padrão sistemático e planejado de ações que são exigidas para garantia de qualidade do software e que devem ser aplicadas ao longo de todo processo de engenharia de software [Pressman 2000].

Nesse contexto a revisão de software é um filtro para o processo de engenharia de software. Pois revisões são aplicadas em vários pontos durante o ciclo de vida do desenvolvimento de software eliminando defeitos em cada fase antes de passar para a fase seguinte [Pressman 2000].

As revisões podem ser classificadas em:

- Discussão informal: realizada pelos grupos de desenvolvedores para resolver problemas técnicos.
- Apresentação: exposição do projeto de software pelo autor, para os clientes, administradores e pessoal técnico.
- Revisões Técnicas Formais (RTF): avaliações técnicas do software, realizadas em pequenos grupos, fornecendo informações confiáveis sobre as atividades que são realizadas.

As revisões técnicas formais têm como objetivos principais:

- Descobrir erros de função, lógica ou implementação, em qualquer produto de software;
- Verificar se o software que se encontra sob revisão atende a seus requisitos;
- Garantir que o software tenha sido representado de acordo com padrões predefinidos;
- Obter um software que seja desenvolvido uniformemente;
- Tornar os projetos mais administráveis.

Dentre os principais métodos de revisão técnica formal estão: Walkthrough, Peer-review e Inspeção.

3.1.1 Walkthrough

Nesta técnica a revisão é feita através de uma execução passo a passo de um procedimento ou programa (no papel), com o objetivo de encontrar erros. Dura em média uma a duas horas. Envolve equipes pequenas de três a cinco pessoas, onde é feita uma simulação da execução por cada revisor, controlada um testador que durante a reunião disponibiliza um conjunto de casos de teste e monitora os resultados obtidos de cada revisor.

3.1.2 Peer-Review

É uma técnica formal de inspeção de código realizada em pares de programadores com mesmo nível de conhecimento, o objetivo desta técnica é obter pontos de vista diferentes do desenvolvedor e revisar o material, a fim de encontrar problemas de qualidade, apenas um programa ou algumas funcionalidades são revisados de cada vez. Os resultados obtidos vão para um relatório para a revisão e se forem pertinentes passam para o relatório final oficial. O problema desta técnica são as disputas pessoais. Por esse motivo deve ser analisado o produto não o desenvolvedor.

3.1.3 Inspeção

A inspeção é um processo de revisão formal de software e corresponde a uma das mais importantes atividades de Garantia de Qualidade de Software, sendo que o principal objetivo é descoberta antecipada de falhas (produção de uma saída incorreta em relação à especificação), de modo que eles não se propaguem para o passo seguinte do processo de software. Assim, a Engenharia de software tem utilizado a inspeção como um dos métodos mais eficientes e efetivos na busca por um produto de melhor qualidade. [Felizardo 2004].

A inspeção visa encontrar erros lendo, entendendo o que o documento descreve e checando através de um checklist as propriedades de qualidade requeridas, é composto por seis fases, que são: Planejamento, Apresentação, Preparação, Reunião de Inspeção, Retrabalho e Acompanhamento [Fagan 1986].

No Planejamento os inspetores são selecionados e os materiais a serem revisados são preparados; na Apresentação, o grupo recebe instruções essenciais sobre o material a ser inspecionado, especialmente sobre o que deve ser inspecionado; na Preparação, integrantes do time de inspeção se preparam para desempenhar o papel designado a cada um; no momento da Reunião de Inspeção os defeitos são encontrados, discutidos e categorizados; no Retrabalho o autor do documento corrige os defeitos encontrados pelo time de inspeção e na etapa de Acompanhamento, o time de inspeção é responsável por assegurar que todos os defeitos encontrados foram corrigidos e nenhum outro tipo de defeito foi introduzido na fase de Retrabalho. O Acompanhamento também pode ser realizado somente pelo moderador [MacDonald 1995] [Fagan 1986].

A figura a seguir ilustra as etapas e papéis envolvidos no processo de inspeção de software.

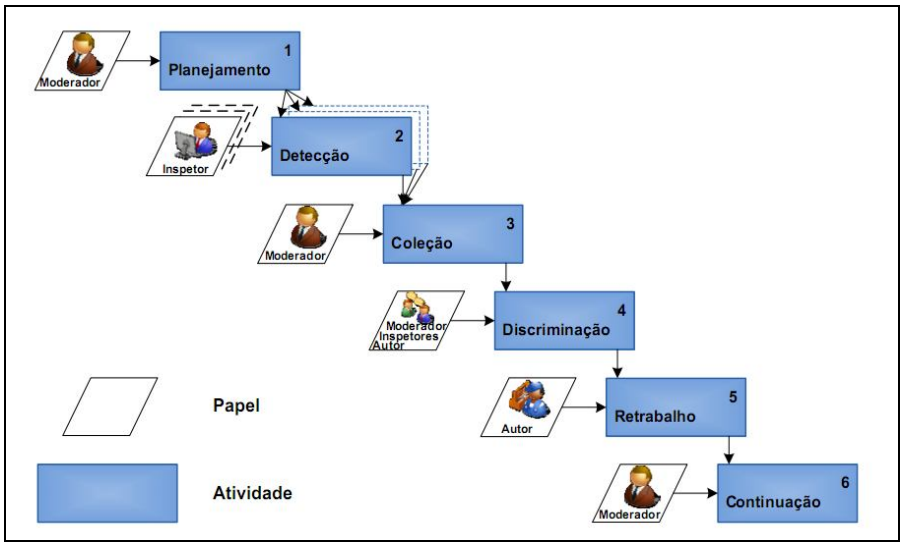

Figura 2. Etapas da inspeção de software [Porto 2009].

3.1.3.1 Etapas

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.

3.1.3.2 Aspectos abordados

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.

3.1.3.2.1 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.

3.2 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, aqui trataremos das essenciais, são elas: Ad-hoc, Check-list e Leitura Baseada em Perspectiva (PBR).

3.2.1 Técnica de leitura Ad-Hoc

Essa técnica 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.

3.2.2 Técnica de leitura Check-list

Similar à técnica Ad-Hoc, porém cada revisor recebe um checklist, onde os itens desse 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.

3.2.3 Técnica de leitura baseada em perspectiva

A 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 [Travassos 2004]. A figura a seguir ilustra a utilização de PBR no desenvolvimento do produto de software.

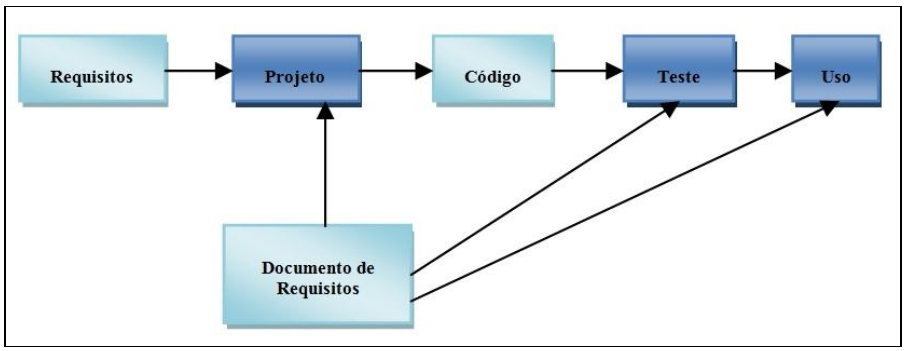

Figura 3. Técnica de leitura baseada em Perspectiva

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. O indivíduo pode revisar o documento de software da perspectiva do desenvolvedor, do testador e do usuário final. 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. Os cenários descrevem as atividades que devem ser executadas pelo indivíduo no momento da leitura a fim de descobrir defeitos em suma um cenário é uma coleção de procedimentos que permitem operacionalizar estratégias para detectar defeitos. A seguir, apresentam-se os cenários e questões, sob as perspectivas de leitura do usuário e do testador [Dória 2009]:

3.2.3.1 Perspectiva do usuário

Espera-se que o revisor execute as seguintes tarefas: defina o conjunto de funções que um usuário esteja apto a executar; defina 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?

3.2.3.2 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?

4. Ferramentas de apoio

Em geral revisões são trabalhosas, muitas vezes executadas manualmente e por isso lentas podendo levar a erros. A automatização de algumas tarefas pode auxiliar os revisores a compreenderem melhor o processo e diminuir o esforço, além de diminuir o custo e aumentar o desempenho desta atividade. Muitos esforços têm sido empregados no desenvolvimento de ferramentas para apoiar a inspeção de software, existem hoje ferramentas que apóiam a inspeção de código fonte, como também ferramentas de

inspeção de artefatos de software, até mesmo ferramentas que oferecem suporte a todas as etapas do desenvolvimento. Dentre as ferramentas desenvolvidas e usadas pelas indústrias de software, aqui apresentamos duas, dentre tantas outras de relevância no mercado:

- CRISTA [Porto 2009]. Uma ferramenta que apóia o processo de inspeção baseado na técnica Stepwise Abstraction, facilita a navegação pelo código e possui vários recursos que ajudam na compreensão do código e em sua documentação. Ela provê a geração de diferentes relatórios que auxiliam o processo de inspeção, por meio de uma interação constante do usuário com a ferramenta, solicitando relatórios que possam transferir abstrações realizadas para o código em forma de comentário, gerando uma redocumentação para o mesmo. A ferramenta oferece um sistema de wizards, que guia o usuário na realização de várias tarefas. Além disso, a CRISTA disponibiliza para o inspetor uma seção de ajuda, na qual existem informações a respeito do processo de inspeção com a técnica Stepwise Abstraction. É uma ferramenta muito útil no contexto de inspeções de código.
- IBIS [Lanubile 2003]. Utiliza a web em conjunto com notificações por e-mail para apoiar inspeções assíncronas com equipes geograficamente distribuídas. IBIS visa explicitamente apoiar a reorganização do processo de inspeção e permitir a sua realização de forma sistematizada. IBIS não limita o tipo de artefato a ser inspecionado e na detecção de defeitos atualmente permite apenas a utilização das técnicas ad-hoc e de checklists. IBIS não fornece apoio aos pontos de tomada de decisão do processo de inspeção, sendo as atividades de planejamento e de continuação do processo tratadas como simples cadastros, sem apoio para a realização de suas tarefas. IBIS tem sido utilizada em estudos experimentais recentes para obter conhecimento na área de inspeções de software e para avaliar aspectos da reorganização do processo de inspeção.

5. Conclusão

Inspeções não requerem necessariamente a execução do sistema e assim podem ser usadas antes da implementação estar concluída. Podem ser aplicadas em qualquer representação (artefato) do sistema. Por detectar defeitos assim que eles aparecem, quando um erro é encontrado, é conhecida também sua natureza e localização, facilitando sua correção. Isso não ocorre com o Teste de Software.

Inspeção e Teste são técnicas de V&V complementares. Elas devem ser usadas em conjunto para garantir maior cobertura e confiabilidade.

Inspeções podem checar a conformidade com especificação, mas não a conformidade com os requisitos reais do cliente, ou seja, é uma técnica de verificação e não de validação por isso não podem ser utilizadas para checar características não funcionais como desempenho, usabilidade, estas atividades ficam a cargo dos Testes de software.

Referências

FAGAN, M. (1986) "Advances in Software Inspection", IEEE Transactions on Software Engineering, Vol. SE-12, NO. 7.

MACDONALD, F; MILLER, J; Brooks, A.; Roper, M; Wood, M. (1995) "Automating the Software Inspection Process", Empirical Foundations of Computer Science (EfoCS), Departament of Computer Science, University of Stranthclyde.

PRESSMAN, R. S. (2000) Software Engineering – "A Practitioner's Approach". 5th edition McGraw Hill.

DÓRIA, E. S. (2001) "Replicação de estudos empíricos em engenharia de software". Dissertação (Mestrado em Computação) – Instituto de Ciências Matemáticas e de Computação (ICMC) Universidade de São Paulo, São Carlos.

BARTIÉ, A. (2002) "Garantia de Qualidade de Software". Edição: 2002, Campus.

LANUBILE, F., MALLARDO, T., CALEFATO, F., (2003). "Tool support for Geographically Dispersed Inspection Teams", Software Process Improvement and Practice, 2003, 8: 217-231 (DOI: 10.1002/spip.184).

FELIZARDO, K. R. (2004). "Apoio computacional para inspeção de software". INFOCOMP: Revista de Ciência da Computação, v.3, n.2. Lavras, MG.

KALINOWSKI, M., SPÍNOLA, R.O., TRAVASSOS, G. H., (2004). "Infra-Estrutura Computacional para Apoio ao Processo de Inspeção de Software". No: Simpósio Brasileiro de Qualidade de Software, Brasília.

SOMMERVILLE, (2007). "Engenharia de Software". 8ª edição, Addison Wesley.

PORTO, D. P. (2009) "CRISTA: um apoio computacional para atividades de inspeção e compreensão de código". Dissertação (Mestrado em Computação) – Universidade Federal de São Carlos, São Carlos.