PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 47/03

Qualidade em Requisitos

Miriam Sayão
Arndt von Staa
Julio Cesar Sampaio do Prado Leite

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22453-900
RIO DE JANEIRO - BRASIL

PUC RIO - DEPARTAMENTO DE INFORMÁTICA ISSN 0103-9741

Monografias em Ciência da Computação, Nº 47/03
Editor: Carlos J. P. Lucena Outubro, 2003

Qualidade em Requisitos*

Miriam Sayão
Arndt Von Staa
Julio Cesar Sampaio do Prado Leite

* Trabalho patrocinado pela PUC-RS, CAPES e MCT

Responsável por publicações:

Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC-Rio Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22453-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3114-1516 Fax: +55 21 3114-1530
E-mail: bib-di@inf.puc-rio.br

Qualidade em Requisitos

Miriam Sayão, Arndt von Staa, Julio Cesar Sampaio do Prado Leite

{miriam, arndt, julio}@inf.puc-rio.br

Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)
Rua Marquês de São Vicente 225, Gávea
22453-900 Rio de Janeiro, RJ, Brasil

PUC-Rio.Inf.MCC47/03 Outubro/2003

Abstract: In this article we tackle different quality measurement and improvement techniques that can applied over requirements documentation generated using a classical software approach. Recent literature points to the fact that over 50% of faults detected during tests can be traced down to problems in the requirements documentation. In particular, defective requiremens are pointed as the main cause of faults in software projects. Requirements artefacts quality can be measured using approapriate metrics, indicators and tests. A critical analysis of these artefacts can be obtained by the use of inspections. Software metrics provide substancial information to project managent and can be applied during different development stages. Inspections help the early detection of faults. At the core of the inspection process is identifying the information to be inspected and the tecniques that will be used to uncover defects. IN this paper we present metrics and techniques that can be used in the inspection of requirement documentation generated using a classical software approach.

Keywords: Requirements quality, requirements inspections, requirements metrics

Resumo: Neste artigo procuraremos abordar técnicas para aferição e melhoria da qualidade de um documento de requisitos, documento este gerado segundo uma das metodologias clássicas de desenvolvimento de software. Publicações recentes apontam que aproximadamente 50% das falhas detectadas na fase de testes são oriundas de problemas no documento de requisitos; defeitos em requisitos, isoladamente, é a maior falha em projetos de software. A qualidade de um artefato de requisitos pode ser avaliada através da utilização de métricas, indicadores ou testes; a análise crítica desses artefatos pode ser realizada através de inspeções. Métricas fornecem subsídios importantes para o gerenciamento do projeto, e podem ser aplicadas em diferentes etapas do processo de desenvolvimento. As inspeções ajudam a encontrar defeitos em artefatos, antes que se passe à fase seguinte do processo; no cerne do processo de inspeção estão a identificação das informações a serem checadas e a técnica a ser utilizada para identificar defeitos nessas informações. Apresentaremos métricas e técnicas de leitura aplicáveis a documentos de requisitos gerados pelo uso de metodologias clássicas de desenvolvimento de software.

Palavras chave: Qualidade em requisitos, inspeções em requisitos, métricas para requisitos

Sumário

1. Introdução

Qualidade de software é definida pelo IEEE (The Institute of Electrical and Electronics Engineers) como "o grau com que um sistema, componente ou processo atende (1) aos requisitos especificados e (2) às expectativas ou necessidades de clientes ou usuários". Já a ISO (The International Standards Organization) define qualidade como "a totalidade de características de um produto ou serviço que comprovam sua capacidade de satisfazer necessidades especificadas ou implícitas" [Rosenberg02]. Estas duas definições, oriundas de organizações respeitadas pela comunidade acadêmica, mostram que a qualidade de um produto está estritamente ligada ao atendimento de seus requisitos.

A preocupação com a qualidade na produção de software inicialmente centrou foco na qualidade do produto; esta visão evoluiu e atualmente a preocupação com qualidade envolve tanto o processo de produção (o ciclo de desenvolvimento do software) quanto o resultado final - o produto ou software gerado.

O ciclo de desenvolvimento de um produto de software, de acordo com as metodologias clássicas de desenvolvimento, tem início pelo Processo de Requisitos; nesta fase são determinados os requisitos que o software em construção deverá atender. Este processo, composto pelas fases de Elicitação, Modelagem e Análise [Leite97], gera um conjunto de artefatos que constituem uma *baseline* para o registro e acompanhamento da evolução dos requisitos ao longo do ciclo de desenvolvimento, possibilitando um efetivo gerenciamento de requisitos.

Um desses artefatos, denominado de Documento de Requisitos, é a base para o desenvolvimento do software; os requisitos aqui registrados delimitam a abrangência do software, estabelecem funcionalidades requisitadas pelo conjunto de clientes e usuários, impõem restrições de qualidade, fornecem subsídios para o processo de verificação e validação do software construído. Nas metodologias clássicas, o processo de requisitos é fundamental para o sucesso de um projeto de software; um bom documento de requisitos possibilita estimativas de custos razoavelmente precisas, o cronograma de execução não deverá sofrer variações significativas, usuários podem participar ativamente do processo de validação do software. E, principalmente, pode-se colocar o software em operação com a convicção que os principais requisitos foram atendidos.

Ao longo do processo de desenvolvimento do software, e mesmo durante o Processo de Requisitos, não é incomum que os requisitos já definidos sofram alterações devido a diferentes motivos: mudanças no contexto onde o software está inserido, novas expectativas por parte dos clientes e usuários, negociação entre clientes e desenvolvedores, etc. Posteriormente, durante a fase de testes, erros podem ser detectados e sua correção implicar em alterações em artefatos de requisitos, de desenho, de código ou de casos de testes. Os custos de alterações motivadas por mudanças em requisitos ou mesmo por correção de erros serão tanto maiores quanto menor for a preocupação da equipe em projetar um software de fácil evolução.

Estima-se que o custo de descobrir e corrigir um defeito na fase de testes seja de 5 a 100 vezes maior que o custo de descobrir e corrigir o problema ainda no Processo de Requisitos [Blackburn01] [Rosenberg98]. A correção de um defeito identificado na fase de testes pode implicar em retrabalho nos artefatos gerados nas fases anteriores: requisitos, arquitetura, projeto e implementação. O uso de inspeções ou outras técnicas para melhoria da qualidade possibilita a descoberta de defeitos em todas as fases do ciclo de desenvolvimento, e a maior parte dos defeitos pode ser identificada antes da implementação ter sido completada. Desta forma o retrabalho é menor, já que defeitos detectados nas fases iniciais podem ser corrigidos antes que a implementação aconteça.

Estudos recentes indicam que aproximadamente 50% das falhas detectadas na fase de testes são causadas por defeitos em requisitos [Blackburn01], e que a correção dessas falhas implica em retrabalho muitas vezes evitável; defeitos em requisitos, isoladamente, é a falha com maior freqüência em projetos de software. Após avaliar um bom conjunto de organizações, Capers Jones [Jones96] descobriu que o processo de requisitos é deficiente em mais de 75% das empresas; isto mostra que obter os requisitos além de ser uma fase importante é também uma das mais difíceis de um projeto de software.

No processo de requisitos, alguns aspectos podem ser considerados como críticos [Hammer96]; esses aspectos são: definição, análise e gerência de requisitos. Neste trabalho abordaremos aspectos do processo de requisitos utilizado em metodologias clássicas de desenvolvimento. A seção 2 aborda características desejáveis no Documento de Requisitos; a seção 3 apresenta métricas para apoio à aferição da qualidade do processo e dos artefatos sendo produzidos e o processo de inspeção e técnicas de leitura aplicáveis a Documentos de Requisitos. Finalmente na seção 4 temos as conclusões deste trabalho. Não abordaremos a Gerencia de Requisitos, que justificaria um trabalho à parte.

2. Documento de requisitos: características e riscos

O sucesso de um projeto é diretamente afetado pela qualidade do Documento de Requisitos (DR). É desejável que cada requisito relacionado no Documento de Requisitos apresente as características a seguir [Costello95] [Kar96] [Wilson97]:

univocamente identificável: requisitos que referenciam tabelas ou figuras ou que são na verdade compostos por outros devem ser individualizados;

ser necessário: deve refletir uma funcionalidade ou característica essencial, que não possa ser preenchida por outra existente no produto ou processo; sua ausência implicará numa deficiência no sistema;

ser conciso: a definição do requisito deve ser clara e objetiva, sendo facilmente compreensível;

independente de implementação: esta característica indica que a definição do requisito deve apontar para o que deve ser feito, e não para como fazê-lo. Adequada na maior parte das vezes, esta característica no entanto é função de requisitos não funcionais, os quais podem pré-determinar uma série de aspectos de implementação;

viável: claramente, a implementação do requisito deve ser possível de ser feita, nas condições e no estado-da-arte atuais, e a um custo razoável;

bem definido: a definição deve procurar ser objetiva e o mais completa possível, não necessitando de informações adicionais para ser entendida;

não ambíguo: muitos documentos de requisitos são escritos em linguagem natural, que é inerentemente ambígua; nesses casos deve-se utilizar também um glossário ou incluir o Léxico Ampliado da Linguagem [Leite90] na *baseline* de requisitos, possibilitando a todos os envolvidos o mesmo entendimento;

consistente: ele não deve estar em conflito ou contradição com outros requisitos. Cada requisito deve ser consistente com todos os demais requisitos na especificação; a consistência também deve existir entre os vários níveis de abstração, se é utilizada a decomposição de requisitos em níveis;

unicidade: não deve haver duplicidade de requisitos, ou seja, não devem existir vários requisitos correspondendo a uma mesma característica ou funcionalidade.

verificável/mensurável: deve ser possível, após o sistema estar codificado, verificar se o requisito foi atendido (está presente no sistema) e se a implementação está correta. Isto é particularmente importante para requisitos não funcionais, como por exemplo desempenho, dado que é relativamente comum encontrar definições do tipo "o tempo de resposta deve ser adequado" ou "o usuário não deve ficar aguardando muito tempo pela informação solicitada";

categorizado: deve estar explicitamente indicado a categoria à qual ele pertence, ou seja, se é requisito funcional, funcional, não funcional, inverso ou de interface.

Os requisitos de um projeto devem estar claramente definidos, possibilitando assim, após a fase de testes, a validação do software pelos clientes e a conclusão que os mesmos foram corretamente atendidos. Para criar um Documento de Requisitos mensurável, alguns procedimentos devem ser seguidos [Costello95]:

a) padrões, se definidos, devem ser consistentes para todo o documento de especificação, em especial para especificações em um mesmo nível;

b) deve haver uma definição cuidadosa dos atributos a utilizar, e esta definição deve ser rigidamente seguida. Os atributos devem ser registrados para todo e qualquer requisito (por exemplo: prioridade, status, componente que o implementa, ...);

c) se utilizada a decomposição de requisitos em níveis, o grafo dirigido acíclico que mostra essa decomposição deve indicar, para cada requisito, aqueles aos quais ele é subordinado e também aqueles que o compõem;

d) especificações de requisitos devem ser armazenadas em meio eletrônico, possibilitando assim sua identificação e rastreabilidade para requisitos em níveis superiores e inferiores, para suas fontes (pré-rastreabilidade) e componentes que os implementam (pós-rastreabilidade), conforme item 4.1.

Os maiores riscos a serem evitados no processo de construção e registro do documento de requisitos estão relacionados a: [Lawrence01]

não incluir um requisito crucial: deixar de incluir um requisito vital pode implicar em grande volume de retrabalho; se o problema é descoberto apenas na fase de testes, podem ser necessárias mudanças na estrutura do sistema e nos componentes previstos. Usuários às vezes não conseguem explicitar suas reais necessidades ou acreditam estar claro aquilo que está implícito na sua colocação e que pode não ser percebido pelo engenheiro de requisitos. Requisitos não funcionais não explicitados ou não mensuráveis também são fortes causadores de problemas para os responsáveis pelo sistema. Provavelmente este é o maior risco de um documento de requisitos;

representação inadequada de clientes: se uma classe de clientes ou usuários não foi representada no conjunto de interessados, é bastante provável que suas necessidades em relação ao software não estejam presentes no documento de requisitos. Quando isto for detectado, provavelmente vai implicar em revisão em todos os artefatos já elaborados ou em elaboração (desenho, codificação, casos de teste, ...). A interação com os clientes na negociação e validação do conjunto de requisitos é fundamental, principalmente para o que for decidido em relação a conflitos entre requisitos;

modelar apenas aspectos funcionais: a ênfase na elicitação de requisitos costuma ser maior em relação a requisitos funcionais; atributos não funcionais, no entanto, são relacionados à qualidade que o software deve apresentar e não são facilmente modelados em casos de uso. O software, além de apresentar as funcionalidades necessárias, deve apresentar características relacionadas à facilidade de uso, desempenho, tolerância a falhas, segurança e integridade. Outra característica importante para a evolução é a estruturação adequada dos componentes, voltada à facilidade de manutenção.

falta de inspeções nos requisitos: o custo da remoção de defeitos em requisitos cresce geometricamente ao longo do ciclo de desenvolvimento. Inspeções podem detectar defeitos precocemente, ainda na fase de requisitos, diminuindo o retrabalho e custos associados (vide item 3.2);

busca da perfeição nos requisitos antes de iniciar a construção do software: deve-se assumir que alterações em requisitos são inevitáveis, e portanto é necessário um efetivo controle de mudanças (vide item 4). Deve-se identificar áreas de incerteza, mantendo atenção nas solicitações de alteração e buscando o preenchimento de lacunas antes da implementação estar concluída;

estabelecer requisitos não implementáveis ou não evolutíveis: deve-se considerar restrições de hardware e software para o projeto em questão, principalmente para projetos de software embarcado (ou embutido). Em projetos deste tipo, restrições relacionadas à capacidade de armazenamento e velocidade devem estar sempre presentes, pois o custo final do produto deve estar afinado com o mercado ao qual ele se destina e também precisa ser competitivo.

definir requisitos incorretamente: se a definição está incorreta, a implementação também estará. Provavelmente o problema só será detectado na fase de validação do software ou após liberação da versão. Os requisitos devem ser validados pelos clientes e usuários antes da fase de desenho iniciar.

3. Qualidade em requisitos: métricas e inspeções

Avaliar a qualidade de um Documento de Requisitos envolve aspectos relativos ao controle da qualidade dos artefatos de software produzidos no processo de requisitos. Diferentes abordagens podem ser aplicadas: validação por prototipação, uso de requisitos executáveis, geração e execução de baterias de testes. A qualidade de um artefato de requisitos pode ser avaliada através da utilização de métricas, indicadores ou testes; a análise crítica desses artefatos pode ser realizada através de inspeções.

Métricas fornecem subsídios importantes para o gerenciamento do projeto, e podem ser aplicadas em diferentes etapas do processo de desenvolvimento. É possível desenvolver casos de teste ainda no processo de requisitos; dificuldades na geração indicam problemas na definição dos requisitos. As inspeções ajudam a encontrar defeitos em artefatos, antes que se passe à fase seguinte do processo; no cerne do processo de inspeção estão a identificação das informações a serem checadas e a técnica a ser utilizada para identificar defeitos nessas informações. Testes são efetuados após a implementação dos componentes estabelecidos no processo de desenho/projeto e visam identificar erros que possam comprometer a funcionalidade do software; geralmente são efetuados testes do tipo *caixa branca* (executados pelo programador visando verificar a correção da implementação) e *caixa preta* (executados por equipe externa visando verificar completude e funcionalidades) [Oliveira01].

Abordaremos a utilização de métricas e de inspeções para os artefatos gerados no processo de requisitos. Inspeções localizam e apontam defeitos/problemas no artefato em questão, sendo então possível ações corretivas que melhorem a qualidade do artefato antes que se passe à fase seguinte no processo de desenvolvimento ou evolução. Métricas são utilizadas como indicadores da evolução do processo e podem apoiar o gerenciamento do projeto; também são aplicáveis a artefatos visando

compará-los à média na empresa. Em caso de identificação de desvios em relação ao esperado, devem ser tomadas medidas corretivas para sanar o problema detectado; quanto mais cedo se faz essa correção, menor será o custo a ela associado [Blackburn01] [Laitenberger01].

3.1. Métricas

Na busca por qualidade, temos alguns questionamentos a fazer: porque medir? o que deve ser medido? A literatura pesquisada aponta para um conjunto de métricas e indicadores, que nem sempre abarcam o ciclo de vida completo de desenvolvimento de software. Buscamos aqui apresentar aqueles que tratam do processo de requisitos, que é a fase inicial e fundamental do ciclo de vida do software, independente da abordagem de desenvolvimento utilizada (espiral, prototipação, XP, ...).

Porque medir? As métricas e indicadores trazem informações objetivas úteis para o acompanhamento, gerenciamento e controle do processo de desenvolvimento [Costello95]. As métricas identificadas na literatura para o processo de requisitos [Cleland-Huang01], [Costello95], [Davis93], [Farbey90], [Fenton97], [Halligan93], [Hammer98], [Lawrence01], [Leishman02] [O´Neill99], [Robertson97], [Rosenberg98a], [Hammer96], [Rosenberg98] podem ser agrupadas nas seguintes classes:

- métricas para aferição da qualidade
- métricas para gerenciamento e evolução de requisitos
- métricas para verificação/validação

Deve-se observar que as métricas não devem ser analisadas isoladamente, dado que podem existir correlações entre elas; por exemplo, se a volatilidade do sistema é alta, o número de requisitos já implementados e verificados (aprovados na fase de testes) pode refletir um valor que rapidamente irá ser modificado.

Para melhor avaliar os índices e valores obtidos, a organização deve manter uma *baseline* com as medições obtidas em seus projetos, comparando então o valor obtido numa determinada avaliação com outros anteriormente registrados na base. Se a organização em questão dispõe de uma boa *baseline*, pode aplicar técnicas estatísticas e identificar se o valor encontrado está fora do esperado, considerando uma distribuição de freqüências que seja adequada à métrica ou indicador em questão [Fenton97].

A existência de uma *baseline* também é desejável para estimativas em relação a projetos que irão iniciar; a partir do histórico de métricas pode-se verificar o modelo de distribuição de freqüência aplicável [Fenton97] e tornar possível a realização de estimativas que apoiarão as atividades de gerência de projeto de software, especialmente aquelas de forte importância junto aos clientes como por exemplo cronogramas e custos.

3.1.1. Métricas para aferição da qualidade do documento de requisitos

É comum trabalhar com documentos de requisitos escritos em linguagem natural. Apesar de possibilitar ambigüidades, a linguagem natural possui a característica de ser compreendida por todos os participantes de um projeto de software. Linguagens formais também têm sido utilizadas para documentos de requisitos, mas seu formalismo não é facilmente compreendido por parte dos clientes e usuários. Problemas com ambigüidades, por outro lado, podem facilmente ser equacionados com glossários ou léxicos da linguagem utilizada no domínio da aplicação. As métricas apresentadas a seguir são aplicáveis a documentos escritos em linguagem natural.

A) Métricas da SATC

A literatura pesquisada apontou para trabalhos desenvolvidos pelo Software Assurance Technology Center (SATC) do Goddard Space Flight Center (GSFC-NASA). O trabalho desenvolvido pelo grupo iniciou-se após a constatação que as diversas ferramentas disponíveis para apoio do processo de requisitos não trabalhavam a questão da qualidade dos artefatos gerados. Após avaliação de numerosos documentos de requisitos desenvolvidos naquela organização, o grupo criou uma ferramenta para apoio às tarefas de análise de documentos de requisitos escritos em linguagem natural. Esta ferramenta foi denominada de ARM (Automated Requirement Measurement), e está disponível para *download* no endereço http://satc.gsfc.nasa.gov/tools/index.html.

Diversas publicações da equipe do SATC [Hammer98] [Rosenberg98a] [Hammer96] [Rosenberg98] [Rosenberg02] [Wilson97] apresentam as métricas obtidas com o uso do ARM em documentos de requisitos e analisam os resultados obtidos. As métricas sugeridas são:

linhas de texto: indicador do tamanho do documento

imperativos: corresponde à quantidade de palavras e frases indicando algo a ser executado ou providenciado; usado como base para contagem de requisitos

continuação: frases que seguem um imperativo e introduzem a especificação de requisitos de nível mais baixo; utilizada para contagem de requisitos suplementares

diretivas: referências para tabelas, figuras, notas

frases dúbias: podem causar incerteza e originar ambigüidades ou múltiplas interpretações

incompletos: dependem ainda de definição por parte dos clientes/usuários; são os denominados TBD´s - *to be defined* - registrados no Documento de Requisitos como "a ser definido", "a ser complementado", "a ser fornecido" ...

escolhas/opções: palavras que aparentam dar flexibilidade ao desenvolvedor para atender às especificações, mas que ao mesmo tempo mostram ausência de definição ou ambigüidade, tais como "opcionalmente .....", "pode-se verificar que ...", ...

Os valores obtidos na aplicação destas métricas devem ser utilizados para comparação com a média da empresa, obtida de uma *baseline* onde foram registrados os resultados da análise de projetos já desenvolvidos na organização. Esta comparação pode apontar para desvios em relação à média, mostrando pontos que devem receber atenção, análise e talvez medidas corretivas. Também devem ser analisados cruzamentos de resultados, como por exemplo imperativos x tamanho texto, incompletos x imperativos. A tabela 1 a seguir apresenta informações da *baseline* resultante da análise de 56 documentos de requisitos da SATC, e sua comparação com os dados obtidos na análise do Projeto X [Hammer98]:

		linhas de texto	imperativos	continuações	diretivas	frases dúbias	incompletos	escolhas/opções
56 docs	mínimo	143	25	15	0	0	0	0
	mediana	2.265	382	183	21	37	7	27
	média	**4.772**	**682**	**423**	**49**	**70**	**25**	**63**
	máximo	28.459	3.896	118	224	4	32	130
	desvio padrão	759	156	99	12	21	20	39
	Projeto X	**34.664**	**1.176**	**714**	**873**	**13**	**480**	**187**

Tab. 1 - Comparação das métricas do Projeto X com informações da *baseline*

A comparação do Projeto X com os valores da *baseline* de métricas mostrou os seguintes desvios/problemas:

a) apesar do número de linhas de texto ser 21% maior que o valor máximo da *baseline*, o número de imperativos (indicador do número de requisitos) é de apenas um terço do máximo registrado. Esta constatação indica que o documento de requisitos do Projeto X é um documento prolixo, não atendendo portanto a requisitos de qualidade como concisão. Por ser um documento em linguagem natural, deve-se evitar palavras, expressões ou mesmo frases desnecessárias, pois elas podem gerar ambigüidades;

b) o documento apresenta alto número de definições incompletas; existe ainda para ser definido um volume de requisitos que corresponde a aproximadamente metade dos já definidos;

c) o número de escolhas/opções é bastante alto, indicando que nesses casos não se tem clareza a respeito de requisitos que o software deverá atender.

No mesmo endereço onde pode ser obtida a ferramenta ARM (http://satc.gsfc.nasa.gov/tools/index.html) também pode ser recuperada uma tabela que reúne os resultados da análise de 96 documentos da NASA e de projetos comerciais e militares; esta tabela pode ser uma referência inicial para comparação com métricas resultantes da análise de outros documentos de requisitos.

Um aspecto importante nesse trabalho refere-se ao fato de que os resultados obtidos pela utilização do ARM permitem uma análise que pode levar à produção de documentos de requisitos com uma linguagem mais apropriada. "Dada a natureza técnica e o uso pretendido dos documentos de requisitos, recomenda-se que quem escreve documentos de requisitos seja incentivado a utilizar um estilo de linguagem simples e direto e escrever sentenças com estruturas simples. O documento não precisa ser interessante, mas é necessário que comunique-se eficazmente com o leitor" [Wilson97].

B) Densidade de defeitos em requisitos

$$\text{DensDefReq} = \frac{\text{número de defeitos em requisitos}}{\text{total de requisitos}}$$

Defeitos em requisitos são identificados através de inspeções; a densidade de defeitos é calculada considerando o total de requisitos contidos no Documento de Requisitos. Defeitos não comprometem a funcionalidade do software, dado que são detectados antes que a implementação estar finalizada; valores ideiais para esta métrica são próximos a zero.

As informações a serem registradas na *baseline* incluem identificação do requisito, tipo, fonte e grau de risco associado ao defeito identificado, data da inspeção. Em projetos de maior porte diversas inspeções podem ser efetuadas ao longo do ciclo de desenvolvimento; neste caso, espera-se que a taxa de defeitos diminua a partir do momento em que a estabilidade e a volatilidade de requisitos fiquem próximas a zero.

Esta métrica é de grande importância para o gerenciamento do projeto, já que defeitos em requisitos implicam em retrabalho; quanto mais cedo identificados os problemas, menores os impactos em custos e prazos para a solução dos mesmos.

3.1.2. Métricas de evolução e apoio ao gerenciamento do processo

A) Estabilidade e Volatilidade de Requisitos

A estabilidade e a volatilidade de requisitos são métricas simples, e devem ser avaliadas em conjunto, ao longo do período de desenvolvimento do projeto, em intervalos de tempo regulares. São obtidas pela razão:

$$\text{EstabReq} = \frac{\text{número de novos requisitos}}{\text{intervalo de tempo}}$$

$$\text{VolatReq} = \frac{\text{número de requisitos modificados}}{\text{intervalo de tempo}}$$

O número de novos requisitos é obtido contabilizando-se apenas requisitos elicitados e que ainda não são parte do documento de requisitos em elaboração; já o número de requisitos modificados contabiliza tanto as exclusões quanto as alterações em requisitos.

Estas métricas são adequadas tanto para o gerenciamento do projeto quanto para apontar para a evolução dos requisitos; podem ser aplicadas tanto para o conjunto de requisitos do sistema quanto para um subsistema do mesmo. Levantamento realizado em [Hammer98] aponta para uma taxa média de alterações em requisitos da ordem de 1% ao mês. Ao longo do desenvolvimento do projeto espera-se que os valores de volatilidade, altos no início do processo, diminuam, dada a maturidade do que se deseja implementar e a estabilidade dos requisitos. A maturidade nos requisitos é claramente identificada quando estas métricas atingem valores próximos de zero.

Deve-se considerar que o processo de desenvolvimento do software interfere com a compreensão que os próprios clientes e/ou usuários possuem a respeito de suas necessidades e das possibilidades que o sistema sendo desenvolvido pode apresentar; é comum, portanto, haver mudanças em relação aos requisitos. Deve-se observar se a volatilidade encontrada está dentro dos padrões para aquela organização, para um mesmo domínio de aplicação. Também se deve acompanhar a volatilidade de requisitos quando a estabilidade dos novos requisitos é atingida: isto não implica necessariamente na maturidade do processo de requisitos, pois mesmo que não hajam novos requisitos sendo definidos, pode haver mudanças significativas naqueles já acordados.

Índices de estabilidade e volatilidade fora dos padrões esperados (espera-se valores altos no início do processo, diminuindo à medida que o desenvolvimento avança e tendendo a zero na fase de implementação) apontam para problemas na elicitação dos requisitos, demandando uma investigação sobre suas causas; a origem pode estar em problemas de comunicação entre clientes/usuários e equipe de desenvolvedores, ou na dificuldade dos usuários expressarem suas necessidades. Deve-se observar que quanto maior o porte do projeto, maior será o tempo de desenvolvimento e maior o número de requisitos alterados, implicando em maiores impactos em custos e cronogramas, caso uma ação efetiva não seja tomada quando detectado o problema.

Deve-se observar também que quando o processo de desenvolvimento utiliza ciclos iterativos a faixa de variação dos índices de estabilidade e volatilidade é de menor amplitude que no desenvolvimento clássico. Isto acontece pois enquanto no desenvolvimento clássico o processo de requisitos busca elicitar o conjunto completo de requisitos a implementar, no processo que utilizam ciclo iterativo parte do conjunto de requisitos é obtido a cada ciclo, completando-se ao final do conjunto de ciclos. Espera-se também que a correção de defeitos encontrados na especificação tenha um custo menor, pois o conjunto de componentes a revisar é menor.

Para efeito de inclusão na *baseline*, considerar os seguintes ítens básicos: contadores para requisitos adicionados, excluídos e modificados, classificados por motivo de alteração [Costello95].

B) Taxa de alterações em requisitos

Esta métrica é uma variação da Volatilidade de Requisitos; aqui não são computados os requisitos excluídos, apenas os alterados. Ela é obtida pela razão:

$$\text{AlterReq} = \frac{\text{número de alterações aceitas}}{\text{total de requisitos}}$$

O total de requisitos aqui considerado é o total inicial; a taxa de alterações aceitas mostra o quanto as solicitações iniciais estavam próximas ou distantes das reais necessidades dos clientes/usuários, ou o quanto as especificações iniciais estavam incorretamente compreendidas. Assim como a métrica de Volatilidade de Requisitos, os valores ideiais devem estar próximos a zero.

C) Cobertura de Rastreabilidade Requisito-Fonte e Requisito-Componente

O rastreamento de requisitos é utilizado para prover relacionamentos entre requisitos, projeto e implementação final do sistema e possibilita uma adequada compreensão dos relacionamentos inter e através dos artefatos de requisitos, desenho e implementação. A rastreabilidade, característica de um sistema que registre esses relacionamentos, é implementada por um conjunto de ligações entre requisitos inter-relacionados, entre requisitos e suas fontes, entre requisitos e os componentes que os implementam.

A Figura 1 a seguir mostra como as ligações possibilitam acompanhar a "vida" de um requisito, em ambas as direções; podemos visualizar quatro tipos de ligações (também chamadas rastros). A pré-rastreabilidade permite passar da origem dos requisitos (fonte ou contexto a partir do qual eles emergem) à sua especificação no documento de requisitos e vice-versa; pós-rastreabilidade vincula os requisitos ao desenho do sistema e sua implementação, e vice-versa [Ramesh01].

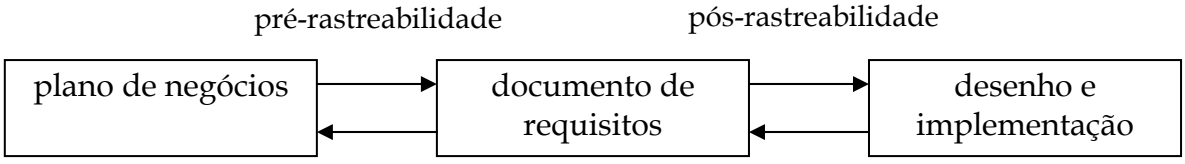

Figura 1 – Rastreabilidade de Requisitos

Estas métricas podem ser aplicadas tanto para o conjunto de requisitos do sistema quanto para um subsistema do mesmo. A implementação dos quatro tipos possíveis de rastreabilidade [Ramesh93] permite um efetivo gerenciamento de requisitos. Devido ao alto custo da rastreabilidade, é mais comum serem aplicadas as métricas Cobertura de Rastreabilidade Requisito-Fonte (CobRastrRF) e Cobertura de Rastreabilidade Requisito-Componente (CobRastrRC); elas são dadas pelas fórmulas:

$$\text{CobRastrRF} = \frac{\text{número de requisitos ligados às fontes}}{\text{total de requisitos}}$$

$$\text{CobRastrRC} = \frac{\text{número de requisitos ligados a componentes}}{\text{total de requisitos}}$$

A Cobertura de Rastreabilidade Requisito-Fonte mostra o percentual de requisitos que permitem a identificação da fonte ou contexto que os originou; já a Cobertura de Rastreabilidade Requisito-Componente mostra o percentual de requisitos que permitem associação aos componentes que os implementam. Valores ideiais, neste caso, são aqueles próximos a 1.

Em seu extenso e excelente artigo sobre rastreabilidade [Ramesh01], Ramesh e Jarke apresentam uma classificação dos desenvolvedores de software em dois grandes grupos: usuários *low-end* e *high-end*. Os chamados usuários *low-end* possuem pouca experiência em desenvolvimento, e tendem a ver a questão da rastreabilidade como uma obrigação imposta pelos gerentes de projeto ou apenas para atender a exigências de padrões. Já usuários do segundo grupo, os *high-end users*, possuem maior tempo de atuação e experiência nas questões relativas ao desenvolvimento de projetos de software e tratam adequadamente a questão da rastreabilidade de requisitos, provavelmente por já terem vivenciado situações de alterações de requisitos em projetos de médio ou grande porte. A rastreabilidade é particularmente importante para o registro da evolução dos requisitos ao longo do desenvolvimento do projeto e para estimativas de impacto das alterações no desenho, principalmente em prazos e custos de desenvolvimento.

3.1.3. Métricas de verificação e validação de requisitos

Verificação é o processo para determinar se um produto preenche os requisitos estabelecidos durante a fase anterior no processo de desenvolvimento; significa verificar se ele está internamente completo, consistente e correto, o que possibilita passar à fase seguinte. Validação é a avaliação para assegurar que ele atende às necessidades do usuário. Verificação responde à questão: "estamos construindo o produto corretamente?", e validação responde à questão: "estamos construindo o produto certo?".

A) Taxa de validação de requisitos

$$\text{ValidReq} = \frac{\text{número de requisitos validados}}{\text{total de requisitos}}$$

A validação de requisitos é a última fase do processo de requisitos; pode-se iniciar o desenho e implementação do software. Espera-se que o valor obtido esteja próximo de um ao iniciar a fase de desenho, indicando que a maioria dos requisitos já foram validados pelos usuários/clientes.

Este indicador deve ser analisado em conjunto com a estabilidade e volatilidade dos requisitos; chegada de novos requisitos ou alterações nos já validados implica em nova validação.

B) Ligações de requisitos a casos de teste

Na fase de análise de requisitos, podem ser gerados casos de testes para o processo de verificação do software. Dificuldades no processo de geração de casos de teste significam que ou o requisito não está claro ou não é mensurável, o que implica em revisão da definição.

A análise das ligações de requisitos a casos de teste explicita requisitos não testáveis e/ou requisitos que, por terem muitas ligações, dificultam o processo de verificação da correção. A figura 2 a seguir, adaptada de [Hammer98] mostra que uma simples análise visual dessas ligações aponta requisitos que se enquadram no caso de requisitos não testáveis.

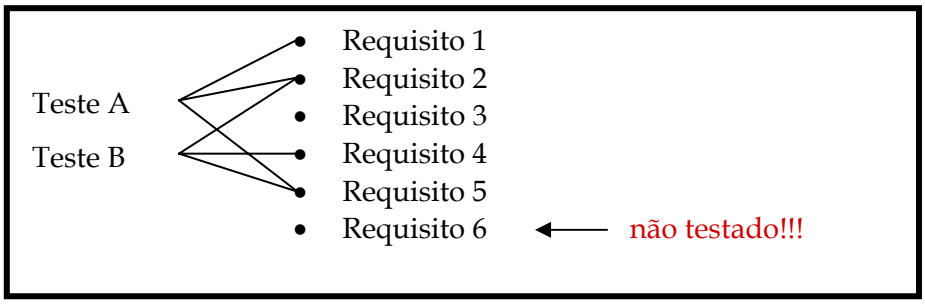

Figura 2 – Ligações de requisitos a casos de teste

A contagem de requisitos por caso de teste é uma métrica simples de obter e foi apresentada em [Hammer98]; os valores obtidos devem gerar um gráfico como o exibido na figura 3. A análise do gráfico resultante é um bom parâmetro para apontar requisitos cuja verificação é complexa, pois exigem a execução de um grande número de casos de teste para conclusões sobre sua correção.

Na figura 3 observamos que 446 requisitos necessitam de apenas um caso de teste para que seja verificada a correção da implementação; já cada um do conjunto de 105 requisitos necessita da execução de 5 casos de teste para que seja verificada a implementação. E um único requisito necessita de 13 diferentes casos de teste para ser verificado.

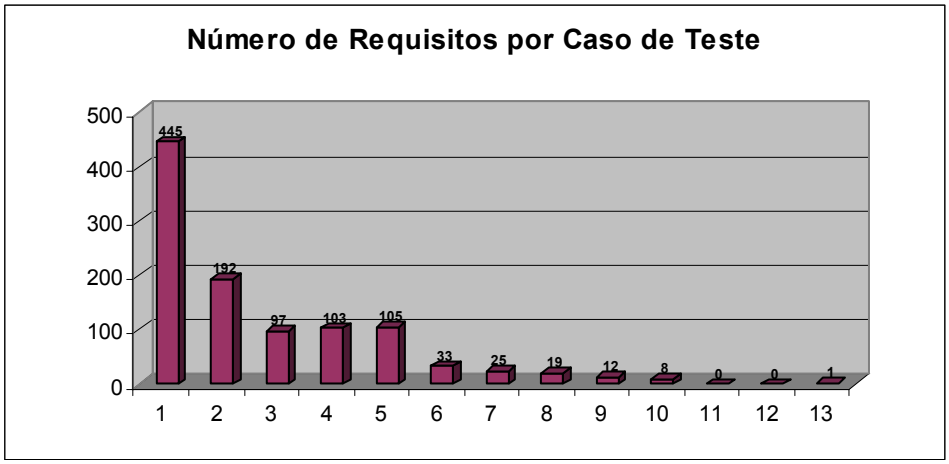

Figura 3 – Distribuição de requisitos por caso de teste

3.2. Inspeções

Inspeção é uma técnica desenvolvida por Fagan em 1972, enquanto desenvolvia seu trabalho na IBM, e foi criada visando aumentar a qualidade de software e aumentar a produtividade dos programadores. Esta técnica inicialmente centrou foco na localização de defeitos na estrutura e no código de programas. Posteriormente ampliada para aplicação em outros artefatos de software [Fagan86] (como requisitos, especificações, arquitetura, planos de teste), atualmente é bastante utilizada por desenvolvedores de software [Lanubile98] [Shull00] [Porter95] [Laitenberger01]. A literatura aponta que o processo de inspeção pode detectar de 30% a 90% dos erros existentes nos artefatos gerados num processo de desenvolvimento de software [Laitenberger01], [Blackburn01].

O processo de inspeção caracteriza-se pela utilização de uma técnica de leitura aplicável a um artefato, buscando a localização de erros ou defeitos no mesmo, segundo um critério pré-estabelecido. A inspeção é aplicável a praticamente todos os tipos de artefatos, sendo possível a utilização de diferentes técnicas de leitura. A técnica escolhida deve identificar as informações a serem verificadas e apoiar a localização de defeitos nestas informações. Das técnicas existentes até o presente, algumas ainda se encontram em estágio de validação e outras já estão integradas à prática empresarial [Laitenberger01]. Dentre estas últimas, apresentamos brevemente as técnicas de leitura baseadas na experiência do inspetor (*ad-hoc*) e as baseadas em checklists e em perspectivas.

Inspeções envolvem análises criteriosas do artefato, considerando aspectos de qualidade que devem ter sido atendidos na sua elaboração. O processo de inspeção compreende várias etapas, iniciando pela coleta dos artefatos a serem utilizados e/ou analisados, passando pela inspeção propriamente dita e chegando ao acompanhamento da correção dos defeitos encontrados. Diversas técnicas de leitura foram propostas e estão em uso corrente; geralmente algum treinamento é necessário para sua efetiva aplicação. Os participantes desempenham diferentes papéis, dependendo da etapa em desenvolvimento, e também do grau de conhecimento e envolvimento no projeto.

A Figura 4 adaptada de [Laitenberger01] apresenta uma visão geral do processo de inspeção e seus principais componentes: etapas, papéis, artefatos e técnicas de leitura.

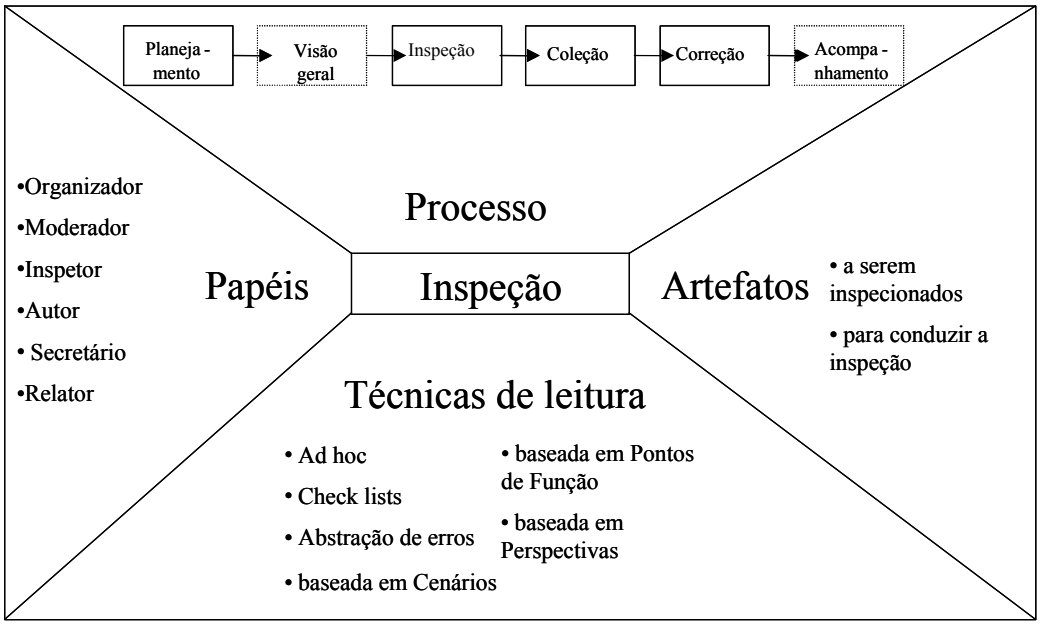

Figura 4 – Visão geral do processo de inspeção

As etapas do processo de inspeção estão descritas a seguir:

Planejamento: nesta etapa inicial do processo, uma pessoa da organização, envolvida diretamente ou não no projeto, é responsabilizada pela organização e condução geral da inspeção. Os participantes são selecionados e recebem atribuições correspondentes aos papéis que irão desempenhar; é definida uma data para a realização da reunião de inspeção e o material a ser utilizado é reproduzido e distribuído aos participantes;

Visão geral: no início da reunião de inspeção, cuja duração deve ser definida previamente, o autor apresenta o artefato a ser inspecionado aos participantes; também pode ser passada a visão geral do projeto. Esta fase é considerada particularmente interessante nas fases iniciais do projeto, que é o caso da inspeção em artefatos de requisitos;

Inspeção: durante a reunião, inspetores avaliam o artefato e registram os defeitos encontrados. A inspeção pode ser realizada de forma individual pelos inspetores, com maior ou menor grau de interação entre eles;

Coleção: defeitos são reunidos e registrados num documento único. Pode haver uma revisão nos defeitos relatados, verificando-se também se eles realmente correspondem a defeitos que necessitam de correção. A comunicação dos defeitos encontrados pode ser feita durante uma reunião com a presença dos inspetores e os autores do artefato, e neste caso é interessante a presença do moderador; ou então o autor do artefato sendo avaliado recebe o documento com os defeitos consolidados;

Correção: defeitos encontrados durante a inspeção deverão ser corrigidos pelos desenvolvedores;

Acompanhamento: são efetuados o acompanhamento e verificação da correção dos defeitos apontados no processo de inspeção.

No processo de inspeção existem diferentes papéis a serem desempenhados pelos participantes. O organizador é aquele que se responsabiliza pela organização e condução do processo como um todo, coletando documentos e informações necessárias, selecionando os participantes de acordo com seu perfil, distribuindo os papéis, agendando encontros e acompanhando a correção dos defeitos. O autor do artefato apresenta uma visão global do mesmo, antes da inspeção ser efetuada. O inspetor analisa os artefatos, seguindo uma técnica de leitura pré-definida, anotando os defeitos encontrados e repassando-os ao secretário; este último reúne os defeitos encontrados pelos vários inspetores, consolida-os num documento e os repassa ao moderador. O moderador também conduz a reunião de inspeção, e deve ter experiência na condução de trabalhos em equipe. O relator apresenta os defeitos coletados aos autores do artefato; esta reunião tem ainda a presença do moderador, para mediar e resolver eventuais conflitos, e do secretário, que registra o encontro.

Das técnicas de leitura indicadas para artefatos produzidos na fase de requisitos ressaltamos: Leitura Ad-hoc, Leitura Baseada em Checklists (ou ckecklists) e Leituras Baseadas em Perspectivas (ou PBR). Estas técnicas já foram validadas em processos experimentais e estão em uso na indústria de desenvolvimento de software, apesar da última, PBR, ser ainda muito recente [Laitenberger01].

3.2.1. Leitura Ad-hoc

A técnica de leitura *ad-hoc* é fortemente baseada no conhecimento e na experiência do inspetor; não há suporte técnico para indicar quais informações checar e como identificar defeitos nessas informações. Isto não significa que as inspeções com esta abordagem não sigam critérios adequados, mas apenas indica que os critérios e

métodos adotados na análise/leitura dos artefatos dependem do inspetor que as efetua.

As qualidades a serem satisfeitas num documento de requisitos e que podem ser verificadas por esta técnica de leitura são: clareza (os requisitos estão bem determinados?), completude (estão presentes todos os requisitos necessários à especificação do sistema?), consistência (os requisitos são consistentes com a visão geral do sistema?), corretude (os requisitos descrevem as funcionalidades de maneira correta?), funcionalidade (as funcionalidades descritas são necessárias e suficientes para atingir os objetivos do sistema?), testabilidade (as funcionalidades permitem a verificação ou teste de forma a mostrar que os requisitos são satisfeitos?) e detalhamento (o nível de detalhe nos requisitos é suficiente para fornecer uma base adequada ao desenho do sistema?).

3.2.2. Leitura baseada em *checklists*

Na técnica de leitura baseada em *checklists*, já consolidada na indústria, o conjunto de inspetores utiliza uma mesma lista para a leitura e análise do artefato. Esta lista relaciona os itens a serem verificados e o que deve ser entendido como defeito. Para cada tipo de artefato há uma lista específica (Documento de Requisitos, Cenários, Léxico Ampliado da Linguagem, Plano de Testes, Casos de Teste, Código, ...).

Essas listas são adaptáveis, e podem levar em consideração os defeitos de maior ocorrência nos sistemas desenvolvidos naquele ambiente. Um dos efeitos colaterais desta abordagem é que, com o uso freqüente de inspeções, os desenvolvedores tenderão a colocar maior atenção nos pontos que originam esses defeitos, e conseqüentemente deverá diminuir o número desses defeitos nos artefatos de software. Por outro lado, defeitos de tipos não caracterizados na *checklist* não serão detectados; para diminuir esses problemas, deve-se procurar seguir algumas regras básicas na construção da lista: a) uma página deve ser suficiente para relacionar as questões a serem respondidas; b) as questões devem ser objetivas e precisas e c) os atributos de qualidade que devem estar presentes no artefato devem ser claros, e as respostas às questões devem assegurar que tal atributo se encontra ou não presente.

Esta técnica de inspeção, quando aplicada a artefatos de requisitos (cenários, documento de requisitos, léxico ampliado da linguagem), pode detectar os seguintes tipos de defeitos:

- sintaxe incorreta nos artefatos (as sentenças/termos não seguem a sintaxe estabelecida)
- informação inconsistente entre artefatos (símbolos definidos num artefato e não referido em outros; símbolos utilizados mas não definidos; descrição não condizente com o símbolo; sinônimos incorretos)

- requisitos não funcionais não explicitados
- informação ambígua (símbolos, termos ou sentenças que possam provocar diferentes interpretações)
- informação desnecessária (atores e/ou recursos em excesso nos cenários)
- ausência de informação (pré-condições, atores e recursos necessários nos cenários)
- exceções não previstas

3.2.3. Leitura baseada em perspectivas

O processo de inspeção baseado em perspectivas, ou PBR (Perspective Based Reading), caracteriza-se por considerar as diferentes perspectivas (visões) dos atores do processo [Shull00] [Porter95] e é aplicável a artefatos de requisitos, desenho e código. Enquanto na leitura baseada em *checklists* todos os inspetores utilizam uma mesma lista de questões, nesta abordagem existem listas e procedimentos diferentes para cada perspectiva ou visão. Esta técnica considera que os diferentes agentes envolvidos no processo de desenvolvimento vêem os artefatos sob diferentes pontos de vista e portanto os revisores para o processo são selecionados de acordo com a utilização que farão do artefato, por exemplo: usuário final, *system designer*, programador, representante da equipe de manutenção, executor de testes e outros.

Um documento de requisitos pode ser visto através de diferentes visões: o usuário final deseja ver ali refletido o conjunto de funcionalidades a ser atendido pelo software; o system designer irá utilizá-lo como base para a criação da estrutura do software, considerando as funcionalidades e restrições descritas, e o testador verá o documento de requisitos como fonte primeira para a geração dos testes, que deverão assegurar que o mesmo atende às necessidades (funcionais e não funcionais) registradas.

As listas utilizadas pelos diferentes inspetores não são genéricas, sendo adaptadas a cada organização; para sua geração são considerados as metodologias adotadas no ciclo de desenvolvimento, o ambiente operacional em uso, o grau de experiência e conhecimento dos desenvolvedores e os problemas de maior ocorrência nos produtos já desenvolvidos pela equipe de desenvolvimento em questão.

Outra importante diferença desta técnica de leitura em relação às técnicas de leitura *ad-hoc* ou baseada em *checklists* está relacionada à atuação dos inspetores: enquanto que nas duas primeiras os inspetores limitam-se a avaliar e registrar os defeitos encontrados, na PBR os inspetores possuem a atribuição adicional de criar um modelo voltado à sua visão, e avaliar criticamente os requisitos e o modelo seguindo uma lista de questões. Exemplificando: um inspetor que assume o papel de testador (preparador dos testes) deve gerar casos de teste para cada entrada do artefato de requisitos em análise e, ao mesmo tempo, verificar se as informações presentes nos

requisitos são adequadas para a geração desses testes. Após isto, analisa criticamente o caso de testes gerado e avalia se ele atende aos requisitos de qualidade estabelecidos por um conjunto de questões associadas a casos de teste. O modelo de testes assim obtido será a base para o conjunto de casos de testes a ser gerado posteriormente.

Esta técnica de inspeção, quando aplicada a artefatos de requisitos (cenários, documento de requisitos, léxico ampliado da linguagem), pode detectar os seguintes tipos de defeitos:

- ausência de informação (definição de termos, unidades de medida,...)
- informação ambígua (vários significados possíveis para um único termo)
- informação inconsistente (quando existem requisitos em conflito)
- fatos incorretos (fato que não pode ser verdadeiro nas condições especificadas para o sistema)
- informação desnecessária (excesso de informação pode confundir os usuários)
- outros tipos de defeitos (defeitos não classificados em nenhum dos tipos anteriores, como por exemplo requisito em seção incorreta)

3.2.4. Automação, custos e benefícios da inspeção

A utilização das inspeções tem sido bem aceita pelos desenvolvedores, e o treinamento, quando necessário, exige apenas de um a dois dias de trabalho. A realização de uma inspeção envolve um total de 10 a 20 horas de trabalho: cada participante necessita de uma a duas horas para as atividades de preparação e mais uma ou duas horas para a inspeção propriamente dita. Nesta geralmente participam de 4 a 5 pessoas, incluindo o organizador, que necessita de um pouco mais de tempo para a preparação do processo e posterior acompanhamento da correção dos defeitos. Não há consenso sobre o modo da execução da inspeção: ela tanto pode ser efetuada individualmente pelos inspetores, ou ocorrer numa reunião com presença adicional de um moderador.

Além da melhoria de qualidade resultante nos artefatos inspecionados (de 30% a 90% de defeitos são localizados em inspeções) a prática tem registrado que, a partir do envolvimento das equipes no processo de inspeção, um dos resultados é a diminuição de erros nos projetos posteriores e conseqüentemente diminuição do tempo do ciclo de desenvolvimento e aumento da produtividade. Os resultados obtidos com utilização das técnicas de leitura *ad hoc* e *checklists*, largamente aplicadas nas empresas, são equivalentes [Porter95].

A especificação de requisitos pode utilizar tanto uma linguagem formal quanto a linguagem natural. A utilização da linguagem natural é mais freqüente, pois favorece o entendimento por parte de todos os envolvidos no processo de desenvolvimento, o que não acontece quando se utiliza uma linguagem formal. Artefatos de requisitos escritos em linguagem natural costumam seguir um conjunto básico de regras de sintaxe, visando a diminuição da ambigüidade e aumento da clareza e concisão do documento. É possível automatizar a verificação da sintaxe com um analisador léxico, e também a verificação cruzada de artefatos (caso de cenários, léxicos e documentos de requisitos).

Independente do processo de inspeção ser manual ou automatizado, a relação custo/benefício é bastante positiva: dados coletados em inspeções efetuadas em diferentes empresas mostram que o retorno do investimento (economia estimada/custo da inspeção) é da ordem de quatro a oito, independente do contexto de aplicação (requisitos, desenho, código ...) [O´Neill99].

4. Conclusões

Não se questiona a importância do processo de requisitos no ciclo de desenvolvimento, quando utilizadas metodologias clássicas no desenvolvimento do software. Requisitos são os alicerces do projeto e definem o nível de qualidade desejado; defeitos em requisitos são, isoladamente, responsáveis por aproximadamente 50% dos problemas detectados em testes. Neste artigo foram apresentados características desejáveis e riscos associados aos documentos de requisitos, e algumas métricas que podem auxiliar na avaliação da qualidade desses artefatos. Também foram apresentadas técnicas de leitura utilizadas em inspeções, para apoiar a detecção de defeitos nos artefatos de requisitos. A gerencia de requisitos, processo vital para o acompanhamento de mudanças e do gerenciamento do software em desenvolvimento, não foi apresentada por ser um tema que justifica um trabalho à parte.

Sem um documento de requisitos que atenda às características de qualidade apresentadas, não há como dizer que um projeto atingiu seus objetivos. Finalizando, reforçamos alguns aspectos considerados essenciais para um documento de requisitos com qualidade: documentos de requisitos devem ser redigidos em estilo simples e direto, evitando-se construções que possam dar margem a mais de uma interpretação. Inspeções devem ser utilizadas como forma de encontrar defeitos ainda no início do ciclo de desenvolvimento, evitando retrabalho para sua correção, como seria o caso se os defeitos fossem encontrados apenas nas fases de testes. A fase final do processo de requisitos, que abrange a verificação e validação, deve contar com a participação efetiva dos clientes e usuários. A qualidade do processo de requisitos e dos artefatos gerados deve ser aferida com utilização de métricas; estas devem ser consideradas não só como mecanismos de acompanhamento da evolução do projeto, mas também para a aferição de qualidade dos artefatos.

Para uma boa utilização de métricas, um ponto fundamental é a escolha do conjunto de medidas a serem coletadas; elas devem efetivamente trazer informações que apóiem a reflexão e decisão sobre o processo ou artefato sendo avaliado. Métricas que exijam muito esforço para sua obtenção devem ser evitadas, e uma vez obtidas, essas medidas devem ser analisadas criteriosamente, para efetivamente contribuir para a melhoria da qualidade. Finalmente, elas devem ser registradas numa *baseline* empresarial para que possam ser futuramente utilizadas como estimativas e parâmetros para outros projetos.

Referências bibliográficas

[Blackburn01] BLACKBURN, M. R.; BUSSER, R.; NAUMAN, A. **Removing Requirement Defects and Automating Test.** Software Productivity Consortium NFP, 2001. Disponível em <http://www.software.org/pub/taf/downloads/RemovingRequirementDefects.pdf>. Acesso em 26 nov 2002.

[Cleland-Huang01] CLELAND-HUANG, J. et al. Requirement-Based Dynamic Metrics in Object-Oriented Systems**.** In: INTERNATIONAL SYMPOSIUM ON REQUIREMENTS ENGINEERING, 5., Los Alamitos, California. **Proceedings**. IEEE Computer Society Press, 2001. p. 212-219.

[Costello95] COSTELLO, R.; D. LIU. Metrics for Requirements Engineering**. Journal of Systems and Software**, vol. 29, nº 1, p. 39-63, april 1995.

[Davis93] DAVIS, Alan et al. Identifying and Measuring Quality in Software Requirements Specifications**.** In: IEEE-CS INTERNATIONAL SOFTWARE METRICS SYMPOSIUM, 1., 1993, Baltimore. **Proceedings**. Los Alamitos, California: IEEE Computer Society Press, may 1993, p. 141-152.

[Fagan86] FAGAN, M. E. Advances in Software Inspections. **IEEE Transactions on Software Engineering**, vol. SE-12, nº 7, p. 744-751, july 1986.

[Farbey90] FARBEY, B. Software Quality Metrics: Considerations About Requirements and Requirements Specifications**. Information and Software Technology**, vol. 32, nº 1, p. 60-64, jan./feb. 1990.

[Fenton97] FENTON, Norman E.; PFLEEGER, Shari L. **Software Metrics: a rigorous and practical approach**. 2nd edition, PWS Publishing Company, 1997.

[Halligan93] HALLIGAN, R. Requirements Metrics: The Basis of Informed Requirements Engineering Management. In: COMPLEX SYSTEMS ENGINEERING AND ASSESSMENT TECHNOLOGY WORKSHOP, 1993, Naval Surface Warfare Center, Dahlgren, Virginia. **Proceedings**.

[Hammer96] HAMMER, T. et al. Requirements Metrics for Risk Identification. In: SOFTWARE ENGINEERING WORKSHOP, 21st, 1996, NASA-GSFC, MD. **Proceedings**. Disponível em <http://satc.gsfc.nasa.gov/support/SEW_DEC96/sel.PDF>. Acesso em 21 out 2002.

[Hammer98] HAMMER et al. Doing Requirements Right the First Time. In: SOFTWARE TECHNOLOGY CONFERENCE '98, April 1998, Salt Lake City, Utah. Disponível em <http://satc.gsfc.nasa.gov/support/STC_APR98/do_reqmnt/do_reqmnt.pdf>. Acesso em 23 set 2002.

[Jones96] JONES, T. Capers. **Applied Software Measurement: Assuring Productivity and Quality.** New York, McGraw-Hill, 1996.

[Kar96] KAR, P.; BAILEY, M. **Characteristcs of Good Requirements**. INCOSE, 1996. Disponível em <http://www.incose.org/rwg/goodreqs.html>. Acesso em 26 out 2002.

[Laitenberger01] LAITENBERGER, Oliver. **A Survey of Software Inspection Technologies.** *Handbook on Software Engineering and Knowledge Engineering, vol. II,* World Scientific Pub. Co, 2001. Disponível em <ftp://cs.pitt.edu/chang/handbook/61b.pdf>. Acesso em 13 set 2002.

[Lam99] LAM, W. et al. Managing Requirements Change Using Metrics and Action Planning. In: EUROPEAN CONFERENCE ON SOFTWARE MAINTENANCE AND REENGINEERING, 3rd, 1999, Amsterdam, Netherlands. **Proceedings**. p. 122-128.

[Lanubile98] LANUBILE, F.; SHULL, F.; BASILI, V.R. Experimenting with error abstraction in Requirements Documents. In: INTERNATIONAL SYMPOSIUM ON SOFTWARE METRICS, 5., 1998, Bethesda, Maryland. **Proceedings**. Los Alamitos: IEEE Computer Society Press, 1998, p. 114-121.

[Lawrence01] LAWRENCE, B. et al. The Top Ten Risks of Requirements Engineering**. IEEE Software**, vol. 18, nº 6, p. 62-63, nov./dec. 2001.

[Leite90] LEITE, J.C.S.P. and Franco, A.P.M O uso de Hypertexto na Elicitação de Linguagens da Aplicação. In: IV Simpósio Brasileiro de Engenharia de Software, SBC, Brasil, 1990. Anais. p. 134-149.

[Leishman02] LEISHMAN, T.; COOK, D. Requirements Metrics can Drow Software Projects. **CrossTalk**, apr. 2002. Disponível em <http://www.stsc.hill.af.mil/crosstalk/2002/04/leishman.html>. Acesso em 16 set 2002.

[Leite97] LEITE, J.C.S.P et al. Enhancing a requirements baseline with scenarios. In: IEEE INTERNATIONAL SYMPOSIUM ON REQUIREMENTS ENGINEERING – RE97, 3rd, 1997, Annapolis, MD. **Proceedings.** IEEE Computer Society Press, 1997. p. 44-53.

[Oliveira01] OLIVEIRA, Flávio & COPSTEIN, Bernardo. **Testes de Software**. Porto Alegre, PUC-RS, 2001. Notas de aula.

[O´Neill99] O'NEILL, Don. National Software Quality Experiment: Results 1992-1999. In: SOFTWARE TECHNOLOGY CONFERENCE, 12th, 2000, Salt Lake City. **Proceedings**. STC Disponível em <http://www.stc-online.org/cd-rom/cdrom2000/default.htm> Acesso em 18 set 2002.

[Porter95] PORTER, A. A. ; VOTTA JR, L. G.; BASILI, V. Comparing Detection Methods for Software Requirements Inspections: a replicated experiment. **IEEE Transactions on Software Engineering**, vol. 21, nº 6, p. 563-575, june 1995.

[Ramesh01] RAMESH, B.; JARKE, M., Towards reference Models for Requirements Traceability. **IEEE Transactions on Software Engineering**, vol. 27, nº 1, pp. 58-93, jan. 2001.

[Robertson97] ROBERTSON, J.; ROBERTSON, S. Requirements: made to measure. **American Programmer**, vol. X, nº 8; aug. 1997. Disponível em <http://www.systemsguild.com/GuildSite/Robs/apmeas.html>. Acesso em 22 out 2002.

[Rosenberg98a] ROSENBERG, L. et al. Software metrics and reliability. 1998. In: INTERNATIONAL SYMPOSIUM ON SOFTWARE RELIABILITY ENGINEERING, 9th. **Proceedings**. Disponível em <http://satc.gsfc.nasa.gov/support/ISSRE_NOV98/software_metrics_and_reliability.html> Acesso em 15 out 2002.

[Rosenberg98] ROSENBERG, Linda. et al. Requirements, Testing, and Metrics. In: PACIFIC NORTHWEST SOFTWARE QUALITY CONFERENCE, 15th, 1998, Utah. **Proceedings**. Disponível em <http://satc.gsfc.nasa.gov/ support/PNSCQ_OCT98/requirements_testing_and_metrics.html>. Acesso em 12 out 2002.

[Rosenberg02] ROSENBERG, L. What is Software Quality Assurance? **CrossTalk**, may 2002, p. 22-25. Recuperada em 26/11/2002 no endereço <http://www.stsc.hill.af.mil/crosstalk/2002/05/may02.pdf>. Acesso em 11 set 2002.

[Shull00] SHULL, F.; RUS, I.; BASILI, V. How Perspective-Based Reading can Improve Requirements Inspections. **IEEE Computer**, vol. 33, nº 7, p. 73-79, july 2000.

[Wilson97] WILSON, W.M.; ROSENBERG, L.H.; HYATT, L. E. Automated Analysis of Requirement Specifications. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING (IASTED), 1997, Boston, MA. **Proceedings**. Disponível em <http://satc.gsfc.nasa.gov/support/ICSE_MAY97/ arm/ICSE97-arm.htm>. Acesso em 13 set 2002.