

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

LIA DEGRAZIA TRODO

Uso de Métricas nos Testes de Software

Prof. Marcelo Pimenta
Orientador

Porto Alegre, novembro de 2009.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Reitor: Prof. Carlos Alexandre Netto

Vice-Reitor: Prof. Rui Vicente Oppermann

Pró-Reitora de Graduação: Profa. Valquiria Link Bassani

Diretor do Instituto de Informática: Prof. Flávio Rech Wagner

Coordenador do CIC: Prof. João César Netto

Bibliotecária-Chefe do Instituto de Informática: Beatriz Regina Bastos Haro

AGRADECIMENTOS

Este trabalho – e o que ele representa – não seria possível sem o apoio de um conjunto de pessoas que direta ou indiretamente contribuíram para que fosse realizado.

Agradeço à Marselha Altmann e à Ana Meira, que me apoiaram continuamente, dando informações e sugestões muito importantes durante a execução deste trabalho.

Ao professor orientador Marcelo Pimenta, pelo apoio recebido nessa jornada difícil, mas que acredito foi concluída com sucesso.

Aos colegas e professores da Ufrgs, com quem convivi e aprendi muito durante todo o curso.

Ao meu marido Volmir, pelo apoio, pela paciência nos momentos difíceis, e, principalmente por me fazer acreditar na minha capacidade de vencer todas as dificuldades.

Aos meus pais, por terem me ensinado valores fundamentais, como a importância do trabalho e do estudo, a disciplina e a persistência, que foram essenciais durante todo o curso.

À minha irmã Janice e meu cunhado Paulo, amigos de todas as horas, e meus sobrinhos Paula e Filipe, que me ensinam constantemente a viver a vida com mais tranquilidade.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS	6
LISTA DE FIGURAS.....	7
LISTA DE TABELAS	9
RESUMO.....	10
ABSTRACT	11
1 INTRODUÇÃO	12
2 MÉTRICAS DE TESTE	13
2.1 Conceitos Relacionados a Métricas de Teste de Software	15
2.2 Motivação para o Uso de Métricas de Teste de Software	18
2.3 Métricas de Teste de Software Existentes	23
2.3.1 Métricas de Produto.....	23
2.3.2 Métricas de Processo	38
2.3.3 Métricas de Projeto.....	49
3 ADOÇÃO DE MÉTRICAS: DIAGNÓSTICO EM EMPRESAS DE TI DO RIO GRANDE DO SUL	53
4 USO DE MÉTRICAS DE TESTE NA PRÁTICA	59
4.1 Implementação de um Programa de Métricas de Teste de Software	62
4.1.1 Identificação das Métricas de Teste.....	64
4.1.2 Obtenção das Métricas de Teste	68
4.1.3 Análise das Métricas de Teste	69
4.1.4 Aplicação das Métricas de Teste	70
4.2 Como Relatar as Métricas de Teste de Software.....	78
4.2.1 Público que Fará Uso do Relatório de Status dos Defeitos	79
4.2.2 Antecipação de Questões Úteis aos Destinatários.....	79
4.2.3 Uso de Objetividade nas Respostas às Questões.....	80
4.2.4 Distribuição das Informações no Relatório	80
4.2.5 Tratamento dos Itens Relacionados aos Riscos.....	80
4.3 Problemas Verificados na Utilização de Métricas de Teste de Software.....	81
4.4 Proposta de Uso de Métricas de Teste de Software.....	87
5 EXEMPLO COMENTADO DE USO DE MÉTRICAS DE TESTE DE SOFTWARE	94
5.1 Contextualização.....	94
5.2 Escolha das Métricas de Teste.....	94
5.3 Métricas Utilizadas	95
5.3.1 Primeira Etapa	95
5.3.2 Segunda Etapa	100
5.3.3 Terceira Etapa.....	102
5.4 Resultados das Métricas.....	107

6 CONCLUSÃO.....	109
REFERÊNCIAS	111
APÊNDICE A	114
APÊNDICE B.....	119
APÊNDICE C	127

LISTA DE ABREVIATURAS E SIGLAS

CMM	Capability Maturity Model
GQM	Goal Question Metric
GUTS	Grupo de Usuários de Teste de Software
GUT	Gravidade, Urgência e Tendência
HTML	Hyper Text Markup Language
RS	Rio Grande do Sul
RH	Recursos Humanos
RUP	Rational Unified Process
SUCESU	Sociedade dos Usuários de Informática e Telecomunicação do RS
TI	Tecnologia da Informação
TMM	Testing Maturity Model
UFRGS	Universidade Federal do Rio Grande do Sul
ULOC	Unit Line of Code
XML	Extensible Markup Language

LISTA DE FIGURAS

Figura 2.1: Distribuição dos defeitos por severidade.	25
Figura 2.2: Densidade dos defeitos por severidade em relação a todo o período de teste e ao teste de regressão	26
Figura 2.3: Índice de severidade dos defeitos.....	27
Figura 2.4: Defeitos reabertos por severidade.	27
Figura 2.5: Tempo para corrigir um defeito por ciclo de teste	28
Figura 2.6: Tempo para encontrar um defeito por ciclo de teste.....	28
Figura 2.7: Distribuição dos defeitos por tipo	31
Figura 2.8 Representação dos percentuais de cobertura de teste.....	34
Figura 2.9: Número de defeitos novos encontrados por semana.....	37
Figura 2.10: Número de defeitos novos, abertos e fechados por semana.....	37
Figura 2.11: Percentuais de defeitos em função das ações tomadas	38
Figura 2.12: Curva S.....	41
Figura 2.13: Curva S.....	42
Figura 2.14: Zero Bug Bounce	43
Figura 2.15: Zero Bug Bounce.	43
Figura 2.16: Zero Bug Bounce	44
Figura 2.17: Relação entre ocorrências e defeitos	45
Figura 2.18: Quantidade de defeitos abertos X defeitos corrigidos por versão.....	46
Figura 2.19: Percentual de defeitos por fase em que foram detectados	47
Figura 2.20: Quantidade de defeitos por módulo em que foram detectados e por severidade	48
Figura 2.21: Percentual de defeitos por fase em que foram incluídos.....	49
Figura 2.22: Relação entre o tempo de teste estimado e o efetivamente utilizado.....	49
Figura 2.23: Distribuição dos defeitos por causa	52
Figura 3.1: Utilização de testes de software.	54
Figura 3.2: Barreiras ao uso de métricas de testes de software	55
Figura 3.3: Métricas de testes de software mais usadas	56
Figura 3.4: Gráfico das estimativas utilizadas.....	56
Figura 3.5: Relação entre o uso de métricas de teste e o tempo de uso de teste de software	57
Figura 3.6: Relação entre o uso de métricas de teste e a especialização da equipe de testes	58
Figura 4.1: Exemplo de Curva S.	61
Figura 4.2: Exemplo de Curva S	61
Figura 4.3: Ciclo de vida do processo de métrica de teste	63
Figura 4.4: Modelo Goal Question Metric (GQM).	66
Figura 4.5: Percentual dos erros por origem dos defeitos	74

Figura 4.6: Percentual dos erros revisados por origem dos defeitos	74
Figura 4.7: Percentual dos defeitos de código por tipo.	75
Figura 4.8: Relação entre as ocorrências e os defeitos encontrados.....	82
Figura 4.9: Índice de severidade dos defeitos.....	84
Figura 4.10: Tempo em horas entre os defeitos encontrados.	84
Figura 4.11: Cobertura dos testes	85
Figura 5.1: Exemplo de planilha Excel utilizada para estimativa do tempo de teste	96
Figura 5.2: Tempo de execução dos testes.	96
Figura 5.3: Lógica utilizada para o cálculo	97
Figura 5.4: Lógica para estimativa do tempo de planejamento dos testes	98
Figura 5.5: Estimativa de tempo de teste para os requisitos apresentados na Tabela 5.1.	100
Figura 5.6: Gráfico utilizado para gerenciamento da estimativa do tempo de execução dos testes.....	101
Figura 5.7: Matriz GUT.....	103
Figura 5.8: Relatório de análise de impacto dos defeitos pendentes.....	105
Figura 5.9: Relatório de retorno dos defeitos	107

LISTA DE TABELAS

Tabela 2.1: Descrição dos níveis de severidade dos defeitos.....	25
Tabela 2.2: Exemplo de distribuição dos defeitos por severidade	26
Tabela 2.3: Exemplo de análise de custos	29
Tabela 2.4: Exemplos de impactos dos defeitos.....	35
Tabela 2.5: Exemplos de métricas de teste básicas relacionadas a casos de teste.....	39
Tabela 4.1: Exemplo de uso de métricas de teste.....	59
Tabela 4.2: Determinando se o esforço de teste foi adequado	62
Tabela 4.3: Requisitos das métricas de teste	64
Tabela 4.4: Descrição da métrica referente ao número de casos de teste bloqueados. ..	68
Tabela 4.5: Exemplo de sumário das métricas de teste	70
Tabela 4.6: Número total de defeitos para cada origem de defeito	73
Tabela 4.7: Número total revisado de defeitos para cada origem de defeito.....	74
Tabela 4.8: Número total de defeitos de código para cada tipo	75
Tabela 4.9: Exemplo de justificativa das recomendações de mudanças para melhoria do processo	75
Tabela 4.10: Número de horas para confirmar, documentar, reportar e corrigir defeitos.	77
Tabela 4.11: Custo para confirmar, documentar, reportar e corrigir defeitos	78
Tabela 4.12: Número de defeitos encontrados.	83
Tabela 4.13: Severidade dos defeitos	83
Tabela 4.14: Análise dos dados disponíveis para a geração das métricas de teste.....	90
Tabela 4.15: Definição das métricas associadas aos dados capturados no processo de testes	91
Tabela 4.16: Relação entre os objetivos da empresa e as métricas de teste	92
Tabela 5.1: Tarefas a serem estimadas	97
Tabela 5.2: Planilha utilizada para estimativa do tempo de planejamento dos testes. ...	98
Tabela 5.3: Tempos estimados para a criação dos casos de teste.....	99
Tabela 5.4: Análise das estimativas da fase de testes.....	101
Tabela 5.5: Análise do tempo por sistema.....	102
Tabela 5.6: Relação das tarefas executadas nos testes	104

RESUMO

O desenvolvimento de software envolve diversas atividades de produção nas quais há uma grande possibilidade de ocorrerem falhas. Os testes de software são indispensáveis para remover defeitos e avaliar o grau de qualidade de um produto e de seus componentes. Apesar das métricas de teste de software serem reconhecidas como muito úteis para a verificação da efetividade e da eficiência de atividades do desenvolvimento de software, surpreendentemente não existem muitos trabalhos publicados que descrevam quais métricas de teste são realmente utilizadas nas empresas (notadamente nas empresas nacionais) e quais os procedimentos para a adoção de métricas de teste. Visando diminuir esta lacuna, neste trabalho são apresentados os fundamentos de métricas de teste, é feito um diagnóstico da situação das empresas de TI do Rio Grande do Sul, no que diz respeito à adoção de métricas de teste, e são investigadas algumas questões relacionadas à implementação de um programa de métricas de teste com base em exemplos extraídos de empresas reais.

Palavras-Chave: Métricas de Teste de Software, Métricas, Teste de Software.

ABSTRACT

The software development implicates several production activities in which there is a great possibility of occurring failures. The software testing are essential to remove defects and to evaluate a product and its components quality degree. Despite of software testing metrics being recognized as very useful to the verification of the effectiveness and the efficiency of the activity in software development, surprisingly there are not many published papers that describe which testing metrics are really used by companies (notedly national companies) and which are the procedures in order to adapt testing metrics. For the purpose of diminishing this gap, in this paper are presented the testing metrics basis, a diagnosis on Rio Grande do Sul IT's is made, concerning test metrics adaptation, and some matters related to the implementation of a testing metrics program based on some examples extracted from real companies.

Keywords: Software Testing Metrics, Metrics, Software Testing.

1 INTRODUÇÃO

Os sistemas de informação estão se tornando cada vez mais complexos e, em função disso, o mercado de teste de software também está evoluindo de forma a buscar a qualidade de toda essa estrutura. O teste de software pode ser considerado como uma fase do processo de desenvolvimento de software, cujo objetivo é atingir um nível de qualidade de produto superior. Os testes antecipam a descoberta de falhas e incompatibilidades, reduzindo o custo do projeto.

Os testes são fundamentais para o controle do projeto, e, por isso, é importante que sejam controlados de forma eficiente (HETZEL, 1987). As medições auxiliam no planejamento, controle e percepção de melhorias no processo de desenvolvimento de software (CRAIG & JASKIEL apud CAMACHO, 2008). Os testes de software são fontes inesgotáveis de métricas, pois uma grande quantidade de indicadores é gerada para que seja possível aferir a qualidade, assim como intervir nos processos para melhorar a cadeia de produção.

A questão é o como tratar todas as métricas geradas, pois, sem um objetivo de análise, são inúteis. Logo, é fundamental delimitar objetivos e estratégias, fazer uma leitura correta dos dados, para transformá-los em informação útil que leve à melhor tomada de decisão.

Há uma grande dificuldade em definir e coletar as métricas, e, por essa razão, na maioria das vezes, elas são ignoradas (CAMACHO). Sendo assim, o objetivo principal deste trabalho é motivar e auxiliar na expansão do uso de métricas de teste. Para isto, foram estudados os conceitos e as técnicas existentes para o uso de métricas de teste de software e a forma como algumas métricas são utilizadas na prática. No capítulo 2, serão apresentados os conceitos fundamentais associados a métricas de teste de software, focando em como proceder para inserir tais conceitos nas atividades de desenvolvimento de software. Visando uma compreensão da situação atual da adoção de métricas de teste e de características dos processos de teste praticados, foi realizada uma pesquisa entre empresas do mercado de TI do Rio Grande do Sul, cujo diagnóstico é apresentado no capítulo 3. O capítulo 4 contempla informações sobre o uso de métricas de teste na prática, incluindo uma proposta de abordagem para incorporar as métricas nos processos de teste. No capítulo 5, é comentado um exemplo de uso de métricas de teste de software. Finalmente, nas conclusões, são resumidas as contribuições deste trabalho, suas limitações e perspectivas de trabalhos futuros visando sua continuidade.

2 MÉTRICAS DE TESTE

Iniciaremos apresentando alguns conceitos relacionados a métricas, que são fundamentais para o entendimento do assunto a ser tratado.

Métricas são o processo em que números ou símbolos são conferidos a atributos de entidades de forma a caracterizá-las de acordo com regras claramente definidas. As métricas fazem um mapeamento do mundo empírico para o formal (FENTON & PFLEEGER apud KANER & BOND, 2004). Elas são uma medida quantitativa do grau em que um sistema, componente ou processo apresenta um determinado atributo. (SCHULMEYER & McMANUS apud PUSALA, 2006).

Medida é a atribuição empírica, objetiva de números, de acordo com regras derivadas de um modelo ou uma teoria, a atributos de objetos ou eventos com o intuito de descrevê-los (KANER & BOND).

Focando a análise nas métricas de software, podemos dizer que a métrica de software é a aplicação contínua de técnicas de medição ao processo de desenvolvimento de software e seus produtos, para fornecer informações gerenciais significativas e pontuais, possibilitando a melhoria dos processos e produtos (GOODMAN apud PUSALA).

Os termos medida, métrica, medição e indicador, costumam causar dúvidas quanto à correta utilização. Em função disso, relacionamos abaixo as definições de cada um deles.

- Medida – é a variável para a qual o valor é atribuído como resultado de uma medição (ISO/IEC 15939, 2002).
- Métrica – é um atributo (propriedade ou característica) mensurável de uma entidade (produto ou processo). No caso de um projeto, uma métrica pode ser, por exemplo, o seu tamanho, uma vez que pode ser medido (MAIA).
- Medição – é o ato de medir, de determinar uma medida. É o conjunto de operações com o objetivo de determinar o valor de uma medida (ISO/IEC 15939).
- Indicador – é a informação relacionada a uma medida, métrica ou combinação de métricas, que pode ser utilizada para compreender a entidade que está sendo medida. É uma métrica que provê uma visão dos processos de desenvolvimento de software e das atividades para melhoria deste processo (BRAUN, 2007).

Podemos dividir as métricas em diretas e indiretas. As métricas diretas ou básicas são resultantes de atributos observados, determinados habitualmente pela contagem, como, por exemplo, custo, esforço, número de linhas de código, entre outros. As métricas indiretas ou derivadas são obtidas a partir de métricas diretas, e citamos como exemplo a complexidade e a eficiência (GARCIA). Uma métrica direta é presumivelmente válida, pois não depende da medida de outros atributos, enquanto que as métricas derivadas dependem da validade das métricas diretas (KANER & BOND).

Quando as métricas são medidas diretamente, normalmente não têm um significado importante para o processo, como, por exemplo, o número de classes de um modelo analisado isoladamente. Uma das formas de adicionar valor a essa informação é através da combinação com outras métricas, como o número de operações e atributos de cada classe, possibilitando estimar o tamanho do sistema (MAIA).

Os enfoques das métricas são a produtividade e a qualidade. Quando medimos um processo, estamos avaliando a produtividade, e quando estamos medindo um produto, a qualidade é que está sendo avaliada. Além da qualidade e da produtividade, as métricas também podem ser orientadas à função e orientadas ao objeto, dependendo, nestes dois tipos, da modelagem e da análise do projeto. As métricas de qualidade incluem os custos com qualidade, ou seja, com a busca da conformidade com o que foi especificado pelo cliente, e os custos da falta de qualidade. As métricas de produtividade são fundamentais, pois todo projeto tem que ser produtivo. Elas abordam a produtividade da mão-de-obra aplicada, cujo objetivo principal é a redução dos custos, além da abordagem da produtividade em relação ao produto gerado, ou seja, o número de horas por artefato. Quanto às métricas orientadas à função, temos a análise de ponto por função, que é baseada na visão do usuário e mede o que é um sistema, o seu tamanho funcional, assim como a relação do sistema com os usuários e com outros sistemas. As métricas orientadas ao objeto são baseadas na análise detalhada do sistema e, para efetivá-las, é necessário acrescentar um peso às características das classes, gerando um fator de ajuste de complexidade orientado a objeto (MAIA).

As métricas são importantes para entender o comportamento e o funcionamento do produto ou do processo, para controlar os processos e serviços, para prever valores de atributos e para tomar decisões a partir de padrões, metas e critérios (GARCIA). As análises baseadas em métricas são mais eficientes do que as que utilizam informações subjetivas. Dados históricos das métricas também são usados para gerar estimativas, a partir de dados de projetos anteriores, além da possibilidade de comparação de projetos, visando ao aumento da qualidade e da produtividade do processo, que são exigências cada vez maiores das organizações.

Os fatores que determinam a importância das métricas de uma forma geral são totalmente adequados ao processo de desenvolvimento de software, e, por conseguinte, dos testes de software, pois envolvem principalmente um melhor conhecimento e um maior domínio dos processos. Além disso, a geração de estimativas adequadas, análise de tendências, maior precisão para avaliar eventuais mudanças nos processos ou para decidir por mudanças, diminuição dos custos e melhor atendimento ao cliente.

Os objetivos das métricas devem ser estabelecidos a partir das necessidades da organização. As métricas têm que ser definidas, priorizadas, documentadas, revisadas e atualizadas (BRAUN). Um programa de métricas envolve planejamento, medição, análise dos dados, tomada de decisão e implementação dessas decisões. É um processo contínuo e sempre sujeito a melhorias.

Para a elaboração de um plano de métricas, devem ser observadas as seguintes questões (GARCIA):

- Por que as métricas satisfazem o objetivo?
- Que métricas serão coletadas? Como serão definidas? Como serão analisadas?
- Quem fará a coleta? Quem fará a análise? Quem verá os resultados?
- Como será feito? Quais as ferramentas, técnicas e práticas que serão usadas para apoiar a coleta e a análise das métricas?
- Quando e com que frequência as métricas serão coletadas e analisadas?
- Onde os dados serão armazenados?

A implementação de um programa de métricas envolve uma série de requisitos, tais como a necessidade de comprometimento da gerência e de toda a equipe, a mudança cultural a que deve ser submetida a equipe, o custo envolvido, o fato de que os benefícios em muitos casos não são verificados imediatamente, e o cuidado na utilização das métricas (GARCIA). A escolha incorreta das métricas ou o uso incorreto das mesmas pode aumentar os problemas, ao invés de auxiliar no processo. Sendo assim, ao adotar as métricas, mesmo que as mais simples, todas as etapas devem ser seguidas para obter sucesso.

Alguns fatores são fundamentais, como o treinamento e o incentivo da equipe que vai trabalhar com as métricas, e a seleção de um conjunto de métricas coerentes, que agregam valor ao processo. Além disso, é importante que na implantação do processo sejam utilizadas métricas mais simples, para que a equipe possa entender a importância das métricas antes de aumentar a quantidade de dados a serem coletados.

A coleta dos dados é um fator crítico do processo e deve começar o mais cedo possível, de preferência na fase de definição dos requisitos. Na captura das métricas de teste, em muitos casos, os dados começam a ser capturados na fase de testes ou, então, na pior das hipóteses, quando o produto é liberado ao cliente. Essa decisão depende do projeto ou dos objetivos da organização (RIOS & MOREIRA).

Nos itens subsequentes, estudaremos especificamente as métricas de teste de software.

2.1 Conceitos Relacionados a Métricas de Teste de Software

As métricas de teste são um padrão de medidas muito útil para a verificação da efetividade e da eficiência de diversas atividades do desenvolvimento de software. Também são usadas para prover informações como estimativas do esforço necessário para o teste. São obtidas e interpretadas durante o processo de testes (BRADSHAW). É importante que elas sejam capturadas e utilizadas corretamente para que possam auxiliar na melhoria do processo de desenvolvimento do software através de informações objetivas e pragmáticas para iniciativas de mudanças do processo.

Um programa de medição dos testes deve ser muito bem planejado e gerenciado. É necessário identificar os dados dos testes que precisam ser coletados, como serão usados e quem é que decide. Os dados devem ser especificados para cada fase do ciclo de

testes. É importante que todos os planos de ação para melhorias no processo de testes efetuadas em função dos resultados das métricas sejam documentados.

As métricas de teste subdividem-se em métricas básicas e derivadas. As métricas de teste básicas são obtidas diretamente do esforço de teste, ou seja, são os dados brutos reunidos pelos analistas de teste, e são usadas para acompanhamento do status e da evolução do projeto. As métricas de teste derivadas são obtidas pelo gerente ou pelo líder de teste, através da conversão das métricas básicas em dados mais úteis que, combinados, podem ser usados para avaliar mudanças no processo (BRADSHAW). Como exemplos de métricas básicas temos a quantidade de casos de teste criados, executados, bloqueados, reexecutados, que passaram, que falharam e que estão sob investigação. As métricas derivadas mais utilizadas são o percentual dos testes concluídos, da cobertura dos testes, dos casos de teste que passaram ou que estão bloqueados, das falhas na primeira execução, dos defeitos¹, dos defeitos corrigidos, do retrabalho, da efetividade e da eficiência dos testes, a taxa de defeitos descobertos e o custo de remoção dos defeitos.

As métricas de teste são variadas. A questão não está em saber se devemos usá-las, mas sim, quais delas devemos usar. Normalmente, quanto maior a simplicidade da métrica, melhor. Por exemplo, pode ser interessante para um projeto em particular a utilização de uma métrica complexa, porém pode não ser prático em função dos recursos e do tempo necessário para capturar e analisar os dados. Além disso, o resultado pode não ser significativo ou útil para o processo atual de melhoria (BRADSHAW, 2004).

Ao utilizarmos métricas de teste, não podemos deixar de considerar a complexidade dos sistemas que estão sendo testados, ou seja, o tamanho do sistema. Por exemplo, não podemos comparar os defeitos encontrados em um sistema de estoque com os de um sistema de controle de uma hidrelétrica.

As principais medidas de um teste são a cobertura e a qualidade. A cobertura diz respeito à abrangência do teste, e a qualidade é uma medida de confiabilidade, estabilidade e desempenho do objetivo do teste. A avaliação da cobertura fornece uma medida para avaliar a conclusão dos testes, enquanto uma avaliação dos defeitos detectados indica a qualidade do software. Os defeitos são considerados como uma solicitação de mudança, pois o objetivo do teste não satisfaz os requisitos, e podem variar de simples contagens a estatísticas mais complexas. Qualquer atividade sistemática de teste baseia-se em ao menos uma estratégia de cobertura (RUP, 2001).

O maior propósito do teste é encontrar defeitos, porém, há um debate constante em função de que não há um padrão absoluto para medir esses defeitos. Habitualmente, as medidas são efetuadas em termos de status, prioridade, severidade, quantidade, tipo, duração, distribuição, custo para encontrar e corrigir, e origem do defeito.

É importante salientar que muitas das informações que são capturadas durante o processo de testes também podem ser obtidas após a aplicação ter sido liberada para o cliente. Existem métricas de teste que analisam, por exemplo, os defeitos encontrados em produção.

As tarefas relacionadas à inclusão de métricas nos testes são usualmente atribuídas ao líder de teste. A escolha das métricas para monitoração, controle da preparação e

¹ Neste trabalho, os termos falha (falta), erro e defeito estão sendo usados como sinônimos.

execução do teste, resolução dos defeitos e apontamento dos riscos, é efetuada durante o planejamento dos testes (MÜLLER et al., 2007). O papel dos gerentes de teste é fundamental para efetuar um planejamento adequado das métricas de teste, com um bom detalhamento dos dados a serem capturados pelos testadores durante vários estágios do ciclo de vida do produto. Alguns requisitos relacionados às métricas de teste podem não ser tão simples de definir na fase de planejamento, porém a maior quantidade de informações possíveis poderá auxiliar no trabalho dos testadores para a captura das métricas.

Existem alguns atributos que podem auxiliar os testadores no gerenciamento das métricas de teste. Os atributos são senso de antecipação, disciplina e uso de ferramentas. O senso de antecipação é baseado na experiência e significa pensar adiante, nos tipos de métricas que podem ser úteis em uma determinada fase do produto. Por exemplo, o testador pode esperar por questionamentos relacionados ao tempo previsto para a execução dos casos de teste ou, então, sobre o status dos componentes em teste, e, em função disso, pode estar sempre preparado com o material necessário, antes mesmo de ser questionado. A disciplina é um atributo que pode auxiliar os testadores a lidarem melhor com o fato de que o serviço de testar é uma tarefa repetitiva, ou até mesmo tediosa. O testador pode ter a motivação de encontrar um caminho melhor para executar o trabalho, através das métricas, tendo isso como desafio. Além disso, o uso de ferramentas também pode ajudar muito para gerenciar melhor as tarefas (MAGAZINE, 2003).

As métricas de teste pertencem a uma área relacionada a melhorias, porém, em muitas organizações, são relegadas a um segundo plano ou são até mesmo desconsideradas. Nos testes de software é comum que diferentes estágios se sobreponham, como, por exemplo, a criação de casos de teste e os testes em si (MAGAZINE). Quando os testadores são pressionados para cumprir o cronograma, a maior parte do esforço é direcionada para que não haja desperdício de tempo e para trabalhar de forma bastante produtiva. Nesse caso, as métricas de teste são fundamentais para otimizar o serviço.

A efetividade do processo de teste é avaliada através da coleta de dados durante todo o ciclo de vida do desenvolvimento. Quanto mais cedo for detectada uma falha que foi incluída na aplicação na fase de projeto, por exemplo, mais efetivo será o processo de teste (McGREGOR & SYKES, 2001). Sua eficiência é medida considerando a fase do desenvolvimento em que o defeito foi gerado e a fase em que foi detectado. Um processo de teste efetivo é aquele em que os defeitos são encontrados na mesma fase em que foram inseridos. Se os defeitos gerados na fase de projeto não forem detectados antes da fase de teste do código, a técnica de teste usada durante o projeto do sistema deve ser alterada para encontrar tais defeitos.

Na prática, testar todas as classes de um programa com todos os valores possíveis para garantir que cada uma está de acordo com o especificado é impossível. O teste exaustivo normalmente não é viável em função do tempo e dos recursos disponíveis, sendo assim, o que acontece na prática é que as classes são testadas suficientemente. Algumas medidas baseadas na cobertura dos testes podem ser aplicadas para fornecer um nível de confiança na qualidade da suíte de teste e, por consequência, dos testes.

O nível 4 do *Capability Maturity Model* (CMM) traz como processos-chave a análise da medição dos processos, e a utilização de métodos qualitativos para gerência da qualidade. As métricas cada vez mais estão ganhando importância junto às

organizações que valorizam a qualidade das aplicações. Enquanto isso, a implementação de programas estruturados de medida ocorre de forma lenta, principalmente na área de testes (PUSALA).

Apesar de muitas métricas de teste serem propostas, acabam sendo ignoradas ou usadas isoladamente, sem agregar valor ao processo. Medidas eficientes do processo de teste são fundamentais para avaliação da efetividade deste processo; são o primeiro passo para tornar os testes uma disciplina concreta.

2.2 Motivação para o Uso de Métricas de Teste de Software

A coleta de métricas é a melhor maneira de saber se um processo está sob controle e se os objetivos estão sendo atingidos, principalmente se o projeto for extenso e complexo. Com os testes, isso não é diferente. As métricas de teste devem ser capturadas para indicar o progresso do processo de testes. Por exemplo, o objetivo dos testes é localizar a maior quantidade de defeitos possível, porém a tendência é que, quanto mais testes forem feitos, menos defeitos passem a ser encontrados. Caso as métricas indiquem que os defeitos aumentam cada vez mais, há indícios de problemas no processo de desenvolvimento (MOCHAL, 2001).

Os testes fazem parte de um processo que deve ser medido e quantificado. O nível 3 do *Testing Maturity Model (TMM)* introduz atividades de controle e monitoração, para garantir que o processo de teste ocorra conforme planejado. Quando há desvio das atividades em relação ao planejado, a gerência pode agir para que as atividades retornem aos objetivos planejados. O progresso do teste é determinado através da comparação do andamento atual do esforço de teste, dos produtos do teste, dos custos e do cronograma. Uma das formas de avaliar o progresso é através das métricas de teste (BURNSTEIN; SUWANNASART; CARLSON). No nível 4, um dos principais focos é a medição acurada do processo de teste, que possibilita o controle e a monitoração dos processos.

A medição é a primeira etapa para o controle. Se não medimos, não podemos entender o processo. Se não entendemos o processo, não podemos controlá-lo. Se não controlamos, não é possível trabalhar no aperfeiçoamento do mesmo (MAIA).

As métricas de teste auxiliam os gerentes de projeto a identificar a posição em que se encontram no projeto e priorizar atividades de forma a diminuir os riscos de ultrapassar os prazos estabelecidos no cronograma.

Através das métricas de teste, é possível traduzir a visão do negócio em objetivos mensuráveis. O uso de métricas de teste normalmente indica um maior grau de maturidade da organização (SOARES & MARTINHO, 2006).

As medições baseadas nos testes buscam a prevenção e a otimização de tarefas empíricas, sendo que o acompanhamento das medidas garante a qualidade no ciclo de trabalho, evitando o retrabalho em estágios posteriores (ELIAS & WILDT, 2008).

Uma das métricas mais importantes dos testes diz respeito ao número de defeitos encontrados. Informações referentes ao custo e ao esforço para correção dos erros também devem ser consideradas, pois adicionam valor ao controle do processo de teste.

Os defeitos fornecem informações sobre a qualidade do produto, do processo e do projeto de teste. É importante fazer a correta interpretação dos dados. Para isso é

necessário definir os fatores de qualidade que são importantes para o projeto e associar tais fatores a métricas (COPSTEIN, 2006). As taxas de localização e correção dos defeitos possibilitam tomar decisões sobre a liberação ou não do produto ao cliente (PUSALA).

Relacionamos a seguir alguns objetivos do uso de métricas de teste:

- Analisar os defeitos: obter informações relacionadas às origens dos defeitos, à forma como foram detectados, quando foram detectados, entre outros.
- Analisar a eficácia e a eficiência dos testes e do processo de testes como um todo
- Avaliar a produtividade do processo.
- Analisar o retorno de investimento.
- Determinar o esforço para automação dos testes.
- Calcular o tempo e os recursos gastos com os testes.
- Avaliar o andamento do teste, em relação ao cronograma, através do status do teste.
- Planejar adequadamente os recursos, prazos e benefícios do processo de testes.
- Identificar áreas que necessitam de melhorias.
- Adequar as suítes de teste de acordo com o nível de cobertura necessário.
- Melhorar a exatidão das estimativas.
- Formar uma *baseline* para as estimativas.
- Auxiliar no gerenciamento do projeto e da execução dos testes.
- Auxiliar nos contratos de software.
- Auxiliar no relacionamento com os clientes.
- Auxiliar na melhoria do processo de desenvolvimento do software, através de dados quantitativos e qualitativos, que possibilitam identificar as melhores práticas, de forma mais objetiva.
- Avaliar os benefícios de novos métodos e ferramentas, através de evidências objetivas, pragmáticas.
- Embasar eventuais solicitações de novas ferramentas e treinamento.
- Avaliar o impacto na qualidade e na produtividade do produto ou do processo que eventuais variações podem causar.
- Viabilizar a tomada de decisão de forma ágil (avaliação de escolhas, comparação de alternativas e monitoramento de melhorias).
- Detectar tendências nos dados que mostrem a necessidade de mais testes em determinadas áreas.
- Identificar áreas de risco que necessitem de mais testes.

- Visualizar se o produto está pronto para liberação ao cliente.
- Indicar a qualidade de forma geral.

Para análise específica dos objetivos relacionados ao custo, tomamos, por exemplo, o custo dos defeitos de software para a economia dos Estados Unidos, que é estimado em \$59.5 bilhões por ano (BRADSHAW). Há uma estimativa de que o custo total poderia ser diminuído em \$22.2 bilhões através do aperfeiçoamento dos testes possibilitando a remoção dos defeitos mais rapidamente e de forma mais efetiva. Em função disso, o uso de métricas é fundamental, pois não é possível aperfeiçoar, melhorar, algo que não podemos medir.

As métricas em si não melhoram a produtividade, porém fornecem informações objetivas que podem ser usadas para essa melhoria, através do entendimento das mudanças necessárias. Tomemos como exemplo uma empresa em que os valores e os recursos humanos disponíveis para os testes são limitados. As métricas de teste podem auxiliar na definição das atividades de teste que produzem melhores resultados, ou seja, os métodos de teste mais efetivos, comparando o número de defeitos encontrados por hora em cada um dos tipos de métodos de teste. Outra forma de estabelecer prioridades nos testes é identificar os defeitos mais frequentes e caros, para determinar o foco dos testes (BRADSHAW).

Muitas vezes, os membros da equipe de projeto entendem que alterações no ciclo de vida do desenvolvimento podem acrescentar qualidade ao processo, assim como reduzir os custos, porém nem sempre eles têm condições de efetuar as modificações em função de que não têm evidência de onde elas devem ser feitas. Quando a equipe trabalha com métricas de teste, esse problema não ocorre, pois as métricas podem ser analisadas juntamente com outras informações disponíveis para definir onde as melhorias devem ser feitas.

Um exemplo que ilustra essa questão pode ser visto em uma equipe de teste que está no meio do período da fase de execução dos testes de um determinado projeto, e as métricas existentes estão sendo analisadas. Uma das métricas indica que menos de 50% dos casos de teste foram executados, o que pode ser um grande problema, considerando que já estão no meio do período previsto. O que explica essa situação é que 30% dos casos de teste estão bloqueados por causa de um defeito em um módulo específico. Dessa forma, com informações objetivas, o gerente de projeto decide antecipar a versão que seria liberada apenas em quatro dias e que resolve o problema que estava bloqueando os casos de teste.

Esse exemplo é importante de ser analisado, pois se as métricas de teste não fossem capturadas, o gerente de projeto não teria as informações necessárias para a tomada de decisão, a equipe de testes teria perdido quatro dias de teste, e, provavelmente, a confiança no trabalho realizado.

Não é possível acompanhar de forma significativa o status do projeto de teste se não houver conhecimento sobre o tempo gasto em cada tarefa para comparar com as estimativas efetuadas.

Enumeramos abaixo algumas perguntas que podem ser respondidas através do uso de métricas de teste de software:

- Quando parar de testar?
- Quanto tempo falta para terminar o ciclo de testes?

- Quanto já foi testado?
- Os testes serão concluídos dentro do prazo previsto?
- Quanto teste ainda tem que ser feito em determinada área?
- Já foi testado o suficiente?
- Qual o custo dos testes?
- Qual o custo para corrigir os defeitos?
- Quantos defeitos podemos esperar?
- Quais os tipos de defeitos encontrados?
- Quantos defeitos já foram corrigidos?
- Quais as áreas do software que têm mais ou menos defeitos?
- Quão estável é a funcionalidade que está sendo testada?
- Qual a técnica de teste que é mais efetiva?
- Estamos testando de forma difícil ou inteligente?
- Temos um programa de testes robusto ou uma suíte de testes fraca?
- Quais os defeitos prioritários?
- Qual o testador que encontrou mais defeitos?
- Quantos defeitos foram localizados por um determinado testador?
- Quantos defeitos foram encontrados pelo usuário?

Normalmente, é difícil para a equipe de testes responder tais perguntas. Pode ser uma tarefa bastante desconfortável, pois muitas vezes os testadores não estão preparados para apresentar os dados solicitados, ou então os dados disponíveis não são adequados, até mesmo porque, em muitos casos, não estavam cientes da necessidade de fornecer tais informações. Em função disso, o testador acaba deixando de lado as tarefas de teste para providenciar as informações solicitadas, e pode tomar bastante tempo, causando o atraso nas tarefas de teste, o que indica falta de planejamento e gerenciamento. Esse tipo de problema pode ser resolvido com gerenciamento adequado das métricas de teste, ou seja, as métricas são uma parte importante do trabalho da equipe de teste (MAGAZINE).

A pergunta “Quanto já foi testado?”, por exemplo, pode ser respondida através da métrica da cobertura dos testes, que é o percentual dos testes conhecidos que foram concluídos. Para isso é necessário um inventário de testes, que possibilita identificar o tamanho do conjunto de testes. Para a pergunta “Já foi testado o suficiente?” é necessário que a equipe de testes tenha feito inicialmente uma estimativa do quanto deve ser testado. Se os testadores não têm essa estimativa, então não sabem exatamente quanto precisam testar, em função disso, podem chegar à conclusão, mesmo que temporária, que testaram o suficiente.

Quanto ao aperfeiçoamento das estimativas através das métricas de teste, temos como exemplo o uso das informações relacionadas às horas gastas nos testes e o número de defeitos por função que podem ser usadas como guia ao gerar estimativas das atividades de teste. Devem ser considerados também o nível de experiência dos

testadores e a complexidade das funções. A divisão das estimativas dos testes em diferentes tarefas também facilita o trabalho, como preparar os casos de teste, testar e retestar.

Apenas a cobertura dos testes não é suficiente para garantir o sucesso do processo de testes, é necessário também um esforço de testes adequado. Ambos devem ser considerados no planejamento. A melhor maneira para determinar se a cobertura é adequada é através de medidas.

Apresentamos uma comparação interessante entre os jogos nos cassinos em *Las Vegas* e o processo de teste de software. Os jogadores que se baseiam na sorte são os que sustentam os cassinos, pois, normalmente, a sorte não é suficiente para ganhar o jogo. Um dos grandes hotéis em *Las Vegas* oferece escolas de jogos em que a principal orientação é para substituir o impulso da sorte por métodos de jogo, que são baseados simplesmente em técnicas de contagem e probabilidade. A idéia é bastante semelhante para a utilização de métodos nos testes. Normalmente, as pessoas não estão interessadas em métodos formais, até que a sorte acabe e tal necessidade seja vivenciada. Às vezes, essa necessidade pode ser extremamente prejudicial para uma empresa, por isso é fundamental avaliar bem a importância do uso de métodos. Para poder provar que os métodos funcionam melhor do que a sorte é necessário efetuar medições para mostrar os resultados da utilização de métodos. Os métodos e as métricas provem um valor que pode ser demonstrado para obter credibilidade e ter validade (HUTCHESON, 2003).

Os testadores têm dificuldade em reivindicar métodos quando a gerência nem mesmo acredita na necessidade dos testes. A melhor maneira de mostrar isso à gerência é através de informações de alta qualidade que possibilitem a tomada de decisões, com efeitos positivos ao processo. Para isso é necessário medir e manter o registro dessas medições, e, além disso, converter tais medidas em informações confiáveis e que tenham valor para a gerência. Uma outra forma de mostrar o valor dos métodos e métricas é mostrando o custo-benefício de usá-los ou não.

Em uma determinada organização, testadores treinados, que utilizavam métodos e métricas, encontravam defeitos a uma taxa de dois ou três a cada hora, enquanto que testadores sem treinamento e que não utilizavam métodos e métricas, encontravam três defeitos por dia na mesma aplicação. Além disso, 90% dos defeitos reportados pelos primeiros testadores eram corrigidos, enquanto que 50% dos defeitos apresentados pelos segundos testadores eram considerados pelos desenvolvedores como não reproduzíveis. Os testadores treinados recebiam um salário que era quase o dobro do salário dos testadores não treinados, porém, apesar disso, o custo para que os testadores treinados encontrassem um defeito era de \$13, enquanto que para os não treinados encontrarem o mesmo defeito o custo era de \$50 (HUTCHESON).

O uso de métricas de teste tem um custo, porém deve ser levado em consideração o fato de que o trabalho dos testadores se torna mais eficiente em função das métricas. Essas questões motivam a gerência para o uso de métricas de teste, ou seja, têm um trabalho muito melhor por um custo menor. Um dos erros de entendimento é de que os testadores não produzem nada além de possíveis defeitos (HUTCHESON). Realmente, eles não tomam nenhuma ação para corrigir os defeitos, porém tais defeitos devem ser removidos durante ou como resultado do esforço de teste, senão o processo de testes será considerado falho. Os métodos e métricas utilizadas pelos testadores adicionam valor ao produto final.

Os dados capturados hoje serão os dados históricos de amanhã, sendo assim, nunca é tarde demais para iniciar o processo de captura de informações em um projeto. Tais dados poderão ser utilizados em estimativas futuras. Sem dados históricos as estimativas são apenas suposições (PUSALA).

2.3 Métricas de Teste de Software Existentes

Relacionaremos a seguir as métricas de teste de software existentes, distribuídas em métricas de produto, de processo e de projeto.

As fontes de pesquisa das informações apresentadas estão relacionadas no Apêndice C.

2.3.1 Métricas de Produto

As métricas de produto servem para auxiliar no controle da qualidade do produto que está sendo testado.

Muitos relatórios são gerados a partir desse tipo de métrica, como, por exemplo, os relatórios de defeitos. Tais relatórios são muito importantes para a avaliação da qualidade do software.

A qualidade é uma medida de confiabilidade, de estabilidade e de desempenho do software, e se baseia fortemente na avaliação dos defeitos encontrados durante os testes, que variam de simples contagens a estatísticas mais complexas. Tais defeitos são um indicativo de necessidade de mudança, pois o objetivo do teste não satisfaz aos requisitos. A avaliação dos defeitos estima a confiabilidade do software atual e prevê como será a confiabilidade na continuação dos testes e a eliminação dos defeitos.

2.3.1.1 Número de Ocorrências

É o número total de ocorrências encontradas em um determinado período ou fase de teste. Uma ocorrência é uma informação da equipe de teste de que a aplicação está apresentando um comportamento indevido. A ocorrência pode ou não levar a uma alteração no software ou na documentação.

É um dos primeiros indicadores que podem ser obtidos dos testes, e fornece informações iniciais sobre a estabilidade do software. O número de erros descobertos, para ter algum significado para análise, deve ser combinado com outras informações.

O cálculo é efetuado a partir do número de ocorrências encontradas.

2.3.1.2 Status das Ocorrências

O status das ocorrências pode variar em função da ferramenta utilizada para localizar o defeito. Normalmente, os status mais utilizados são:

- Pendente de Solução – Registrado pela equipe de teste e esperando para ser tratado pelo desenvolvedor.

- Para ser Retestado – Resolvido pelo desenvolvedor e esperando para ser retestado pela equipe de teste.
- Fechado – A ocorrência foi retestada e aprovada pela equipe de teste.

O objetivo dessa métrica é acompanhar a evolução das ocorrências. A informação é útil para saber o número de ocorrências em cada status.

Nem todas as ocorrências registradas representam um defeito real, podendo ser apenas solicitações de melhoria, por exemplo, ou descrição de defeitos já relatados, assim como podem estar fora do escopo do projeto. No entanto, é importante observar e analisar a razão para tantos defeitos repetidos ou pendentes de confirmação.

2.3.1.3 Índice de Densidade de Defeitos

Apresenta a quantidade ou a taxa de defeitos encontrados.

Os defeitos são as ocorrências obtidas em um determinado período de teste e que resultaram em alteração do software ou da documentação. Ocorrências duplicadas e rejeitadas são eliminadas.

Os relatórios de distribuição de defeitos mostram as contagens de defeitos como uma função de um ou mais parâmetros de defeitos. Esse tipo de métrica pode indicar que alguns itens não foram especificados e permite acessar informações referentes à estabilidade e a confiabilidade do software.

O cálculo é efetuado através da contagem da quantidade de ocorrências que resultaram em modificação do software ou da documentação.

2.3.1.4 Índice de Severidade de Defeitos

É o índice que permite verificar se os defeitos são graves.

O nível de severidade de um defeito é uma métrica fundamental, que pode indicar um potencial impacto no negócio para o usuário final. O impacto no negócio é a relação entre o efeito para o usuário final e a frequência em que ocorre.

Provê indicações da qualidade do produto em teste. Defeitos de alta severidade sinalizam um produto de baixa qualidade, e vice-versa. Essa informação é importante para decidir sobre a liberação de um determinado produto ou versão baseados no número de defeitos e respectivos níveis.

O índice representa a média das severidades dos defeitos. Fornece uma medida direta da qualidade do produto – especificamente, confiabilidade, tolerância a falhas e estabilidade.

Cada defeito possui um nível de severidade e não existe um padrão para defini-lo. A magnitude da severidade do defeito está constantemente aberta para debate. É conveniente usar quatro níveis de prioridade.

Um número é atribuído a cada nível de severidade, sendo que a ordem pode ser alterada, conforme critérios locais, definidos por cada organização. Existem diversos padrões para definição das prioridades, sendo que sugerimos o modelo apresentado na Tabela 2.1.

Tabela 2.1: Descrição dos níveis de severidade dos defeitos

Nível	Descrição
1 – Crítico	Resolver imediatamente. O programa cessa a operação.
2 – Sério	Prioridade alta. Erro severo, porém a aplicação continua.
3 – Médio	Fila normal. Resultado inesperado ou operação inconsistente.
4 – Baixo	Prioridade baixa – design ou sugestão.

Fonte: HUTCHESON.

Cada ocorrência é multiplicada por seu nível de severidade, e todos os resultados são adicionados, formando um número total, que será dividido pelo número de defeitos. O resultado final é considerado o índice de severidade dos defeitos.

A avaliação do teste é efetuada a partir da distribuição dos defeitos nos níveis de prioridade. Tomemos como exemplo a distribuição apresentada na Figura 2.1, em que foi incluído um filtro para mostrar os defeitos em aberto. O gráfico mostra que os critérios não foram satisfeitos, pois há uma grande quantidade de defeitos críticos em aberto, e, em função disso, há um grande risco de que os usuários encontrem sérios defeitos no produto disponibilizado.

Podemos considerar como um exemplo de teste bem sucedido aquele em que não são encontrados defeitos críticos e no máximo cinco defeitos sérios.

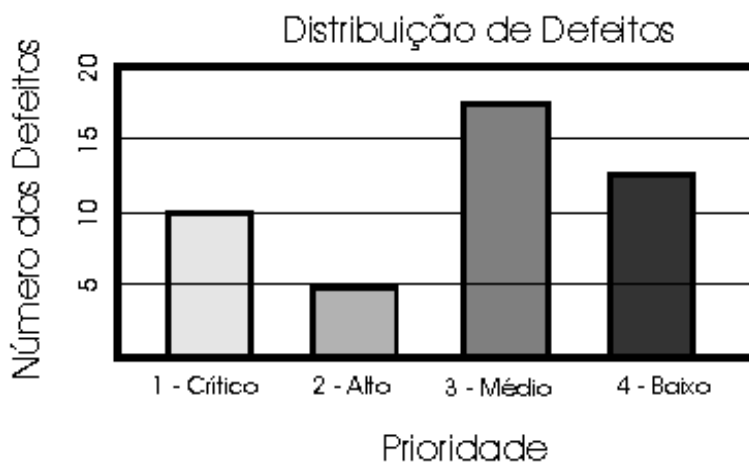


Figura 2.1: Distribuição dos defeitos por severidade (RUP)

Na Figura 2.2 temos um outro tipo de análise para a severidade dos defeitos, em que os índices de todo o período de teste são comparados com os índices do teste de regressão.

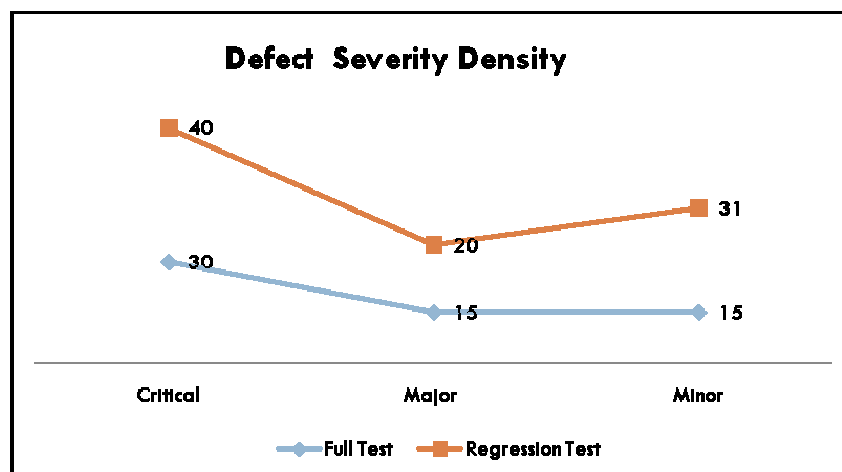


Figura 2.2: Densidade dos defeitos por severidade em relação a todo o período de teste e ao teste de regressão (CAMACHO)

O exemplo na Tabela 2.2 mostra um caso de estudo em que são contabilizados separadamente os erros encontrados e os relatados, por severidade.

Em algumas situações são reportados apenas erros que podem ser reproduzidos. Tal prática não é aconselhável porque possibilita que os erros mais difíceis, os que não são reproduzíveis, sejam ignorados, e, desta forma, repassados aos usuários.

No exemplo da Tabela 2.2, 50% dos erros mais severos não são reportados. Inevitavelmente, o suporte terá que tratá-los quando forem detectados pelos usuários.

Os erros mais severos representam 38% dos erros encontrados, ou seja, mais de um terço dos erros no produto são críticos. Haveria um grande risco em entregar o produto ao cliente com tais defeitos, pois a cada três defeitos encontrados pelo cliente, um poderia ser extremamente sério. Considerando que existem nove erros não reportados, a situação é mais grave, pois há grande probabilidade que tais erros sejam detectados em produção.

Tabela 2.2: Exemplo de distribuição dos defeitos por severidade

Nível	Erros encontrados	Erros reportados
1 – Crítico	18	9
2 – Sério	11	11
3 – Médio	19	19
4 – Baixo	0	0
Total	48	39

Fonte: HUTCHESON

O gráfico na Figura 2.3 mostra uma situação ideal, em que o índice médio de severidade dos defeitos diminui à medida que evolui o ciclo de teste.

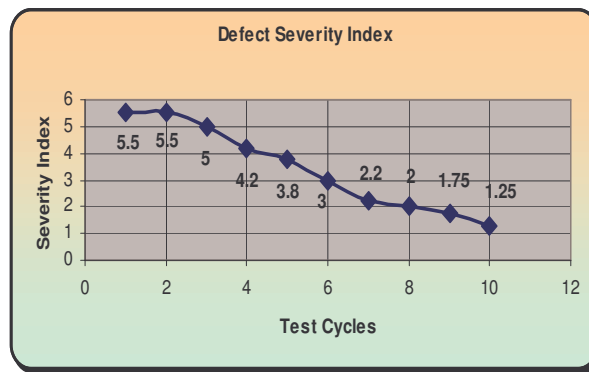


Figura 2.3: Índice de severidade dos defeitos (MAGAZINE)

É possível também utilizar um gráfico para acompanhar o índice de severidade dos defeitos reabertos, conforme Figura 2.4.

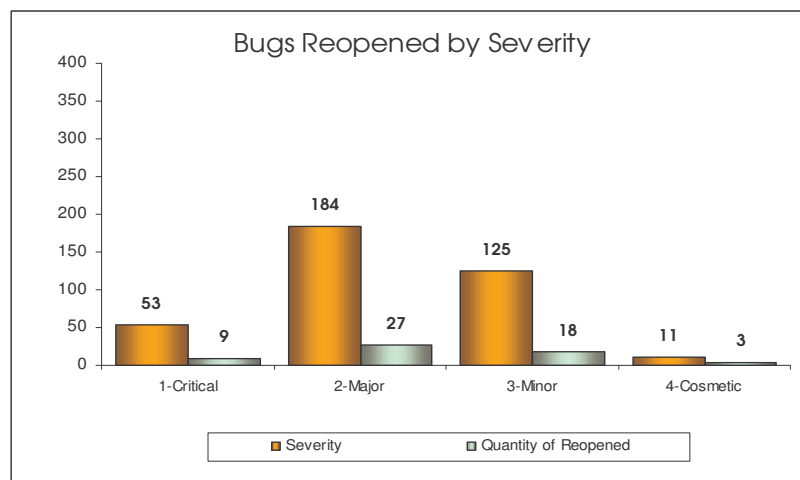


Figura 2.4: Defeitos reabertos por severidade (CAMACHO)

2.3.1.5 Tempo para Arrumar um Defeito

É uma métrica adequada para apresentar o índice de correção dos defeitos. É o esforço necessário para resolver um defeito, incluindo o diagnóstico e a correção.

O objetivo dessa métrica é fornecer uma indicação da manutenibilidade do produto e pode ser usada para estimar o custo de manutenção planejado.

Para calcular o número médio de horas para corrigir um defeito é necessário dividir o número de horas gasto para arrumar os defeitos pelo número de defeitos corrigidos no mesmo período. Pode ser usada outra unidade de tempo, como dias ou semanas.

Uma outra opção referente ao tempo gasto para arrumar os defeitos diz respeito ao tempo semanal gasto, ou seja, o número de horas gastas por semana para corrigir erros. Essa métrica fornece uma noção da complexidade relativa dos erros detectados nos testes. A tendência é que os defeitos mais difíceis de corrigir apareçam no início do processo de testes, de forma que, na medida em que o ciclo de testes evolua, o tempo de correção dos defeitos também seja menor.

A avaliação também pode ser feita comparando o índice de cada ciclo de testes, conforme gráfico da Figura 2.5.

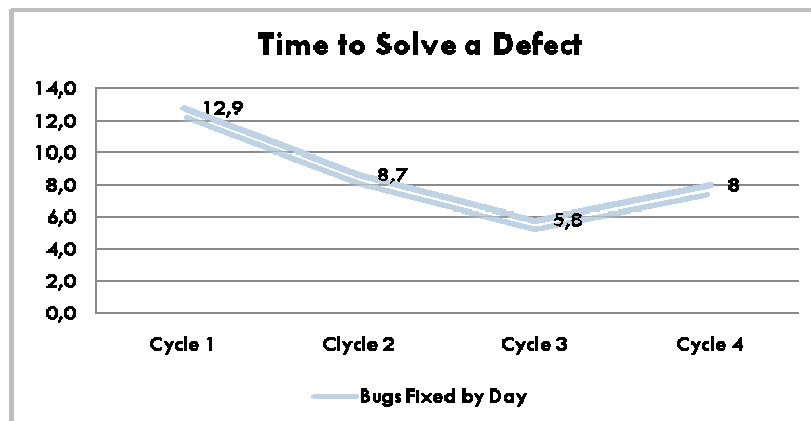


Figura 2.5: Tempo para corrigir um defeito por ciclo de teste (CAMACHO)

Outra métrica similar é a quantidade de horas para correção de defeitos por tipo de erro descoberto. Com o avanço do processo de teste é usual que o tempo necessário para corrigir erros típicos diminua. Caso sejam detectados defeitos que consomem maior tempo para correção, isso pode significar que os defeitos levam a problemas nas fases de projeto ou requisitos.

2.3.1.6 Tempo Médio para Encontrar um Defeito

É o esforço necessário para encontrar um defeito.

Mostra a velocidade em que os defeitos são encontrados, indicando a correlação entre o esforço de teste e o número de defeitos identificados.

O cálculo é feito a partir da soma da quantidade de horas gastas na execução dos testes e no registro dos defeitos, dividida pelo número de defeitos detectados no mesmo período. Os resultados podem ser apresentados em gráficos, como no exemplo da Figura 2.6.

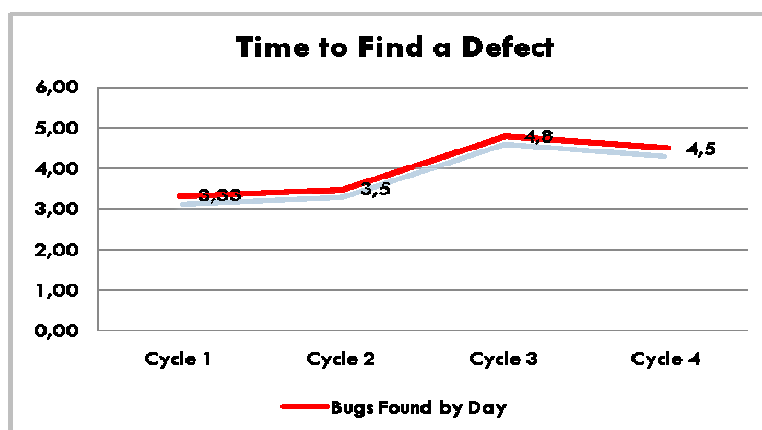


Figura 2.6: Tempo para encontrar um defeito por ciclo de teste (CAMACHO)

A taxa de identificação de defeitos é uma métrica derivada extremamente útil para medir o custo do teste e fornecer indicações da estabilidade do sistema, porém não tem utilidade se for analisada isoladamente. Na Tabela 2.3 apresentamos um exemplo da geração de informações significativas a partir da combinação da taxa de identificação de defeitos com o custo em encontrar defeitos. Esse tipo de informação é muito útil para analisar o custo dos testes, assim como, para avaliar se o sistema está pronto para ser liberado ao cliente. Os dados do exemplo foram obtidos a partir de testes efetuados durante cinco semanas.

Tabela 2.3 – Exemplo de análise de custos

	Primeira semana	Quarta semana
Quantidade de defeitos/hora	5,33	0,25
Custo para encontrar cada defeito	\$9.38	\$199.79
Defeitos reportados por hora	3,25	0,143
Custo para reportar	\$15.38	\$15.38
Custo para encontrar e reportar cada defeito	\$24.76	\$215.17

Fonte: HUTCHESON

Podemos verificar na Tabela 2.3 que no início do período de testes o custo em encontrar e reportar cada defeito é normalmente maior do que o custo de encontrar defeitos. À medida que diminui a quantidade de defeitos encontrados por hora o custo para encontrá-los aumenta significativamente, enquanto que o custo para reportá-los permanece o mesmo durante todo o período de testes. A diminuição na quantidade de defeitos encontrados à medida que o final dos testes se aproxima é normal. O custo para encontrar os defeitos aumenta, considerando que os testadores demoram mais tempo para encontrar os defeitos, mas são pagos por hora.

2.3.1.7 Quantidade de Falhas Encontradas no Produto

A quantidade de defeitos em produção é uma métrica importante para mostrar a efetividade global do processo de testes, assim como, uma idéia da estabilidade do sistema. Possibilita encontrarmos resposta à pergunta “O esforço de teste foi válido?”. O ideal seria que a aplicação não tivesse erros quando liberada para uso pelo cliente, porém normalmente isso não acontece na prática.

Essa métrica é usualmente definida pelos usuários do produto e reportada através do suporte ao cliente. Como são os clientes que reportam as falhas, não é usual que os defeitos sejam ignorados ou relevados, ou seja, a métrica é um bom indicador de performance do teste e de possíveis problemas em novas versões. Ultimamente, essa métrica tem sido considerada em termos de custos, perda de lucro e aumento no custo de desenvolvimento e suporte. Infelizmente, em algumas organizações, a equipe de testes não tem acesso a esse tipo de informação.

Métricas relacionadas podem ser capturadas após a aplicação ter sido entregue ao cliente, como por exemplo, número de defeitos detectados por semana e número de horas gastas para corrigir os defeitos detectados em produção. A tendência é que haja uma diminuição desses números com o passar do tempo.

Uma variação dessa métrica é o tempo que a aplicação fica indisponível ao cliente em função dos defeitos, mostrando os prejuízos causados ao negócio.

As quantidades de defeitos encontrados antes do produto ser liberado ao cliente ou até mesmo após, ou seja, encontrados pelos testadores ou pelos clientes, por si só são consideradas métricas fracas. O ideal é que sejam tratadas juntamente com outras métricas, como por exemplo, a severidade dos defeitos.

2.3.1.8 Tipos de Defeitos Encontrados

Para tornar os testes mais efetivos é importante saber os tipos de erros que podem ser encontrados na aplicação que está sendo testada, assim como a frequência relativa em que esses erros ocorreram no passado. Essas informações históricas podem ser úteis para fazer previsões quanto à qualidade do software, assim como aperfeiçoar o processo.

Os tipos de defeitos são diversos, podendo variar de um erro de entendimento da interface a erros de código, erros da base de dados, falhas sistêmicas, entre outros.

A literatura de testes de software é rica em classificações e taxonomias de falhas, porém, assim como na severidade, a classificação dos defeitos é feita de acordo com padrões locais.

Em um sistema conectado alguns tipos de defeitos são considerados falhas de sistema, ao invés de erro de código, por exemplo. Como exemplos de defeitos tratados como falhas de sistema, causados por falhas ou problemas de conexão, temos as falhas na rede, as falhas de comunicação, as unidades individuais dos dispositivos móveis que estão constantemente conectando e desconectando, os erros de integração, os componentes em falta ou com mau funcionamento, e os erros de sincronização e tempo. Esses tipos de falhas provavelmente serão verificados em produção, sendo assim, os testes que localizam esse tipo de defeito são importantes para evitar problemas no ambiente de produção.

Na Figura 2.7 os defeitos foram distribuídos por tipo, porém, para dar um outro enfoque à análise dos defeitos, a distribuição também pode ser feita considerando o número de defeitos por causa através do tempo e defeitos por módulo. É possível também calcular a densidade dos defeitos por categoria, através da relação entre o número de defeitos encontrados e o tamanho do programa.

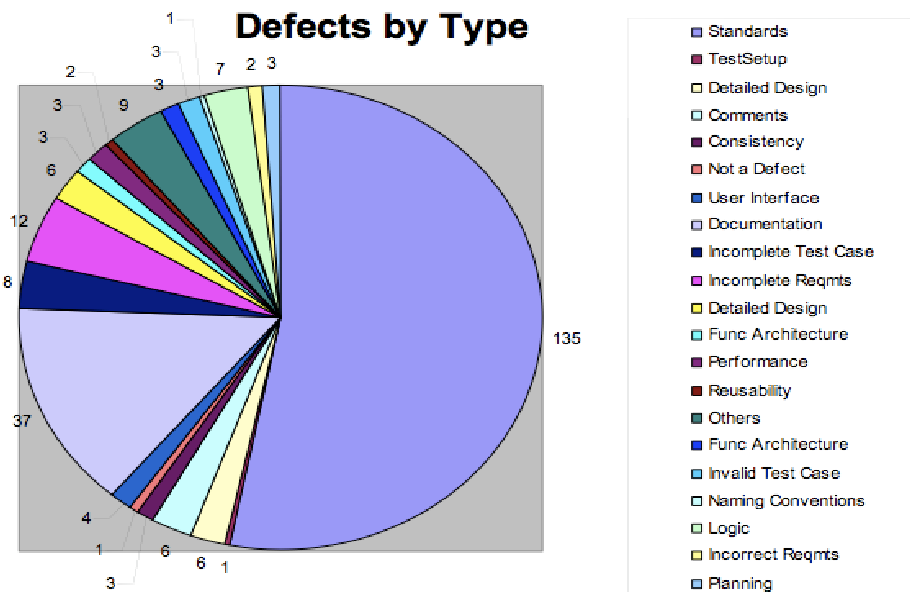


Figura 2.7: Distribuição dos defeitos por tipo (HUTCHESON)

2.3.1.9 Cobertura dos Testes

Cobertura mede o quanto um grupo de testes examina a capacidade de um determinado software. É a medida da abrangência do teste, indicando o nível de confiança atribuído ao teste.

Esta métrica permite verificar se todas as funcionalidades do sistema estão sendo testadas, ou seja, se o teste cobre todas as funcionalidades. Em um conjunto de itens a serem testados, a cobertura de teste é a porção que foi realmente testada.

Indica a integridade dos testes, sem tratar a efetividade dos testes, podendo ser usada como critério de parada dos testes.

Normalmente é apresentada como uma porcentagem, como por exemplo, 100% de cobertura nas declarações significa que todas as declarações em um programa foram testadas, diferentemente de afirmar que todo o sistema foi testado.

No teste unitário a cobertura de testes é uma medida absoluta, baseada em quantidades passíveis de serem contadas, porém no teste de sistema é considerada uma medida relativa, porque o número de testes para um sistema pode ser um conjunto infinito, já que os testes podem não encontrar todos os defeitos em um sistema, assim como nem mesmo os testadores poderão encontrar todos os testes para serem feitos em um sistema. Sendo assim, para que a cobertura possa ser uma medida útil no teste de sistema, é necessário um inventário de testes, que é uma lista de testes identificados para o sistema. A cobertura de teste de sistema mede quantos dos testes conhecidos foram executados. A validade da cobertura de teste de sistema depende da qualidade do inventário de testes.

As atividades sistemáticas de teste são baseadas em pelo menos uma estratégia de cobertura que orienta a geração dos casos de teste. Normalmente, a métrica de cobertura de teste pode ser baseada inicialmente nos requisitos, podendo variar em função das prioridades e objetivos do projeto.

As medidas de cobertura mais usadas são a cobertura de teste baseada em requisitos e em códigos (projeto e implementação). A cobertura de requisitos é medida pela quantidade de requisitos cobertos pelos casos de teste, enquanto que para a cobertura de código é necessário medir a extensão do código coberto através dos casos de teste. Se os requisitos estiverem bem especificados, a cobertura baseada em requisitos pode ser suficiente para análise da abrangência, como por exemplo, se todos os requisitos foram identificados, podemos estabelecer uma medida de abrangência de 75% dos requisitos verificados. Quanto maior o percentual de cobertura, melhor a situação do teste.

Um exemplo de cobertura baseada em requisitos é a verificação de casos de uso, enquanto que, para a cobertura baseada em códigos, temos como exemplo a execução de todas as linhas de código. No caso da cobertura baseada em códigos, a medida é efetuada em termos da quantidade do código fonte executada pelos testes, situação bastante usual para sistemas críticos.

O cálculo da cobertura considera os itens que estão cobertos pelos testes em comparação ao número total de itens do sistema. As medidas podem ser obtidas manualmente, através de equações específicas, ou via ferramentas de automação dos testes.

Adicionalmente, cobertura pode também ser representada nas seguintes relações:

- Quantidade de casos de teste desenvolvidos versus a quantidade de casos de teste planejados para execução.
- Quantidade de casos de teste planejados para execução versus a quantidade de casos de teste executados.
- Quantidade de casos de teste versus a quantidade de casos de teste executados.

2.3.1.9.1 Cobertura de Teste Baseada em Requisitos

A medida da cobertura de teste baseada em requisitos é efetuada várias vezes durante o ciclo de vida do teste, sendo possível obter, por exemplo, a cobertura de testes planejados, implementados, executados e bem-sucedidos.

Caso os requisitos sejam especificados por casos de uso, a cobertura será medida pelo número de casos de uso utilizados e a quantidade de cenários criados para cada caso de uso.

Equação utilizada para cálculo:

Cobertura de Teste = $T^{(p,i,x,s)} / RfT$, onde, T é o número de testes planejados (p), implementados (i), executados (x) ou bem-sucedidos (s), ou seja, T^p é o número de testes planejados expresso como procedimentos ou casos de teste, T^i é o número de testes implementados conforme expresso pela quantidade de procedimentos ou casos de teste para os quais existem scripts correspondentes, T^x é o número de testes executados expressos como procedimentos ou casos de teste e T^s é o número de testes executados expressos como procedimentos ou casos de teste que foram concluídos com êxito e sem defeitos.

RfT é o número total de requisitos do teste.

Um cálculo adicional pode ser efetuado em relação aos resultados das equações anteriores, comparando com a taxa de êxito dos testes. Podem ser definidos critérios de êxito, que serão atingidos ou não a partir dos resultados verificados, possibilitando prever a quantidade de esforço de teste que ainda é necessário.

2.3.1.9.2 Cobertura de Teste Baseada em Códigos

A medida da cobertura de teste baseada em código é efetuada a partir da quantidade de código executado durante o período de testes em comparação à quantidade total de código pendente de execução.

A cobertura pode ser baseada em fluxos de controle ou em fluxos de dados, sendo que no fluxo de controle são testadas linhas de código, condições de ramificação, caminhos que percorrem o código ou outros elementos do fluxo de controle do software, enquanto que no fluxo de dados o objetivo é testar se os estados dos dados permanecem válidos durante a operação do software, como, por exemplo, se um elemento de dados é definido antes de ser usado.

Caso algum código não seja executado no teste, os testadores poderão trabalhar junto com os desenvolvedores para determinar quais os casos de teste não considerados ou então se há alguma implementação do software cuja funcionalidade não tenha sido especificada.

Equação utilizada para cálculo:

Cobertura de Teste = $I^e / Tlic$, onde, I^e é o número de itens executados (instruções de código, ramificações de código, caminhos de código, pontos de decisão do estado de dados ou nomes de elementos de dados.)

$Tlic$ é o número total de itens do código.

Da mesma forma que no item anterior, um cálculo adicional pode ser efetuado em relação ao resultado da cobertura de teste, comparando com a taxa de êxito dos testes, para verificar se os critérios de êxito estabelecidos anteriormente foram atingidos ou não, fornecendo uma base para prever a quantidade de esforço de teste restante.

Há possibilidade de combinar cobertura com complexidade para estimar o esforço necessário para testar um determinado produto. Quanto mais complexo for o software, mais difícil será para atingir um nível específico de cobertura. Podemos citar o número e a complexidade dos métodos existentes nas classes e o número de linhas de código como exemplos de medidas de complexidade.

A Figura 2.8 mostra um exemplo de cobertura de testes em que são mostrados os percentuais realizados da cobertura de requisitos, cobertura de código, relação entre os casos de teste documentados e os executados, e a relação entre os casos de teste executados e os planejados, sendo que apenas o último dos itens atingiu o percentual ideal, de 100% de cobertura.

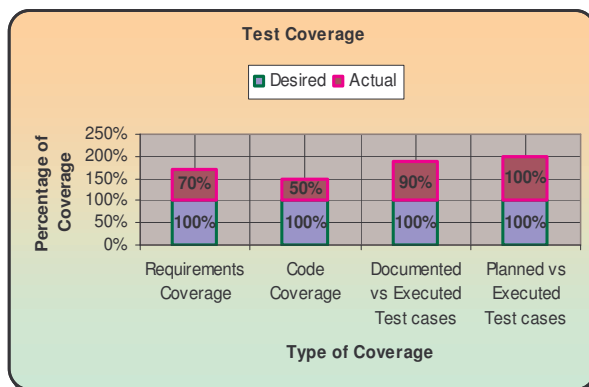


Figura 2.8: Representação dos percentuais de cobertura de teste (MAGAZINE)

2.3.1.9.3 Medida e Uso da Cobertura de Teste

Os dados de cobertura são coletados durante o processo de teste. As suítes de teste são criadas para alcançar alguns tipos de cobertura. Cada caso de teste tem uma razão para ser selecionado e pode estar diretamente relacionado à cobertura de um determinado aspecto do sistema. O grau da cobertura é alterado a partir da inclusão de novas funcionalidades no sistema.

A cobertura é utilizada no momento do lançamento do produto, ou seja, faz parte dos critérios de decisão para liberação do produto, que não deve ser efetuada enquanto os objetivos de cobertura não estiverem satisfeitos, apesar das pressões inerentes às datas previstas de entrega.

2.3.1.10 Efetividade de Caso de Teste

A efetividade de caso de teste é uma métrica que possibilita verificar se os casos de teste encontram defeitos, indicando se os mesmos são úteis, assim como o nível de estabilidade do software.

O cálculo é efetuado a partir da relação entre o número de casos de teste que resultaram em ocorrências registradas e o número total de casos de teste.

Uma outra análise quanto à efetividade dos casos de teste pode ser feita observando se os aspectos importantes do sistema foram bem testados através dos casos de teste selecionados. Esse tipo de análise é uma alternativa à cobertura tradicional de teste, pois assegura que os testes cobrem os atributos mais importantes da aplicação. Para isso podemos utilizar os impactos dos defeitos, cujas categorias são atributos do sistema que são afetados pelos defeitos. A questão é identificar nos testes os defeitos em cada área de impacto que devem ser listados nos relatórios dos testes de sistema, dentro das respectivas categorias.

Na Tabela 2.4 são apresentadas informações adicionais para o testador. Não há um algoritmo específico a ser seguido, porém a lista apresentada pode ser útil na avaliação da cobertura durante a elaboração do planejamento dos testes. A dificuldade existente nesse tipo de análise é que não há garantia de que o programa simplesmente não impacta uma determinada área ou se os testes é que não foram suficientemente minuciosos.

Tabela 2.4: Exemplos de impactos dos defeitos

Impacto do defeito	O que fazer se não forem detectados defeitos que causam esse tipo de impacto
Capacidade	Se defeitos funcionais não forem encontrados, reavaliar os critérios de cobertura por casos de uso.
Usabilidade	Determinar se todas as maiores telas foram exercitadas.
Performance	Considerar se o produto foi suficientemente submetido ao <i>stress</i> . Caso não tenha sido, aumentar a carga, senão, passar.
Confiabilidade	Não há outras ações específicas que não sejam avaliar a confiabilidade, baseados nos resultados de toda a suíte de testes.
Mantenabilidade	Não há ação específica.
Documentação	Determinar se a inspeção tem documentação de usuário correlata e ajuda on-line.
Portabilidade	Garantir que todos os objetivos razoáveis tenham sido incluídos na análise.
Padronização	Identificar interfaces padrão. Os testes baseados nessas interfaces foram aplicados? Se tiverem sido, passar.
Integridade/Segurança	Se um nível de segurança para todos os diretórios tiver sido alcançado, então passar.

Fonte: MCGREGOR & SYKES

2.3.1.11 Efetividade e Eficiência dos Testes

O processo de teste é efetivo quando encontra defeitos, e é eficiente quando os encontra com a menor quantidade de recursos possível.

A efetividade do teste é uma métrica que possibilita verificar o quão adequados foram os testes, a habilidade do conjunto de testes em encontrar defeitos, em suma, a qualidade do conjunto de testes.

Uma cobertura de teste adequada não necessariamente precisa que o conjunto de testes atinja uma alta taxa de cobertura em relação ao inventário de testes, que pode ser muito grande para ser exercitado totalmente. Desta forma, o objetivo é buscar o menor conjunto de testes, que provavelmente irá encontrar o maior número de defeitos dentro do tempo definido.

Normalmente, é mais simples definir um conjunto amplo de testes do que conseguir tempo e recursos suficientes para executá-lo completamente ou para reportar todos os problemas. Em função disso, o número de testes que devem ser executados é quase sempre maior do que o número que realmente pode ser tratado dentro do tempo e com os recursos disponíveis.

A forma mais comum de saber se os testes foram adequados é através da quantidade e do tipo dos defeitos encontrados pelo usuário. A efetividade do teste em encontrar defeitos pode ser medida através da relação entre o número de defeitos encontrados pelo conjunto de testes e os defeitos encontrados no produto. Alguns defeitos podem ser encontrados pelos testadores e pelos usuários, porém são considerados apenas uma vez na medida da efetividade do teste. Outra questão a ser observada é que a efetividade dos testes considera apenas os defeitos encontrados, sem se preocupar se os mesmos foram

corrigidos. A métrica que considera os defeitos identificados e tratados é relacionada à performance.

Tomemos como exemplo uma suíte de testes que cobre 50% do inventário de teste, porém encontra 98% dos defeitos possíveis de serem localizados no sistema. Nesse caso, temos uma cobertura de testes adequada em que os 50% dos testes do inventário que foram executados eram os mais importantes. Qualquer acréscimo na quantidade de testes executados provavelmente traria um benefício mínimo ao processo.

Todos os testes executados durante o período de testes fazem parte da suíte de testes mais importante, porém os testes que encontram defeitos possuem maior valor, e fazem parte de um subconjunto de testes. Além da efetividade nos testes, são considerados os que encontram falhas, que são defeitos que podem voltar a ocorrer apesar de aparentemente terem sido corrigidos.

Quando é possível identificar no esforço de testes o subconjunto mencionado, e instanciá-lo no ambiente de produção, os testadores acrescentam um bom retorno nos investimentos dos testes.

2.3.1.12 *Defeitos por Quantidade de Linhas de Código (kloc)*

É o número de defeitos encontrados em 1.000 linhas de código, e indica a qualidade do produto testado. Essa métrica pode ser usada como base para estimar defeitos em fases futuras ou novas versões.

O cálculo é efetuado a partir da relação entre o número de defeitos encontrados e o número de linhas de código (milheiros).

2.3.1.13 *Situação ou Tendência dos Defeitos em Função do Tempo*

Mostra a quantidade de defeitos em diversos status, em um determinado período de tempo, assim como o número acumulado de defeitos através do tempo. O status dos defeitos pode variar em novos, abertos, concluídos ou executados e corrigidos.

As tendências dos defeitos podem ser apresentadas através de relatórios e gráficos mostrando a contagem dos mesmos, por status, como uma função de tempo, podendo ser cumulativos ou não. Os relatórios de tendências possibilitam a identificação das taxas de defeitos, fornecendo uma visão do estado do teste, uma idéia geral de como está o processo de teste.

As análises de defeitos combinadas com a cobertura de teste podem ser úteis para os critérios de conclusão do teste.

O exemplo na Figura 2.10 mostra o padrão da tendência dos defeitos no ciclo do teste. Ocorre um aumento rápido no número de defeitos no início do ciclo, atinge um pico e diminui mais lentamente com o passar do tempo. Através da análise desse tipo de gráfico podem ser detectados problemas como, por exemplo, atraso no projeto, quando um ciclo de testes com previsão de duração de quatro semanas tem uma quantidade de defeitos que continua aumentando na terceira semana.

O ideal é que a equipe de testes não pare de testar enquanto a quantidade de erros não estiver próxima de zero.

Na Figura 2.9 temos uma situação em que os defeitos estão sendo corrigidos imediatamente e as correções são validadas nos ciclos posteriores, desta forma, a taxa de conclusão dos defeitos segue o mesmo padrão da taxa de detecção dos defeitos. Caso contrário, haveria um problema na correção dos defeitos, que teria que ser avaliado em termos dos recursos para correção e validação dos defeitos ou para reaplicação dos testes.



Figura 2.9: Número de defeitos novos encontrados por semana (RUP)

A Figura 2.10 mostra um esforço de teste bem sucedido em que os novos defeitos são descobertos e abertos no início do projeto, diminuindo com o tempo. Os defeitos concluídos aumentam com o passar do tempo, considerando que os defeitos em aberto são corrigidos e verificados. Tendências muito diferentes das apresentadas na Figura 2.10 podem indicar problemas, sinalizando a necessidade da aplicação de recursos adicionais em áreas específicas do desenvolvimento ou dos testes. Por exemplo, se houver uma quantidade significativa de defeitos antigos não resolvidos pendentes de validação, há uma grande probabilidade de que os recursos de teste não estão sendo suficientes.

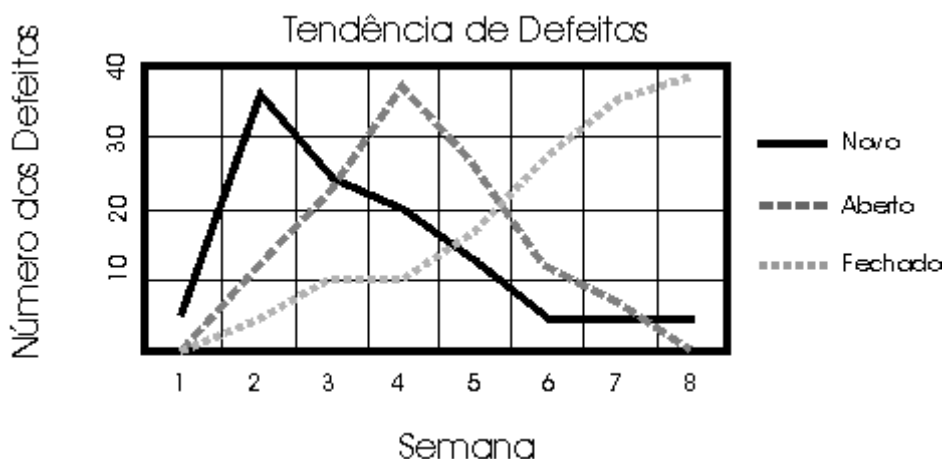


Figura 2.10: Número de defeitos novos, abertos e fechados por semana (RUP)

Outro tipo de relatório que pode ser apresentado utilizando essa métrica é o que mostra o tempo de permanência dos defeitos. É apresentada a distribuição dos defeitos através do tempo de permanência de um defeito em um determinado estado.

2.3.1.14 *Providências Adotadas em Relação aos Defeitos*

Uma outra forma de analisar o status dos defeitos é apresentada na Figura 2.11, onde os defeitos são contabilizados em função das providências adotadas para correção. Essa informação pode ser obtida no final de um determinado ciclo de teste.

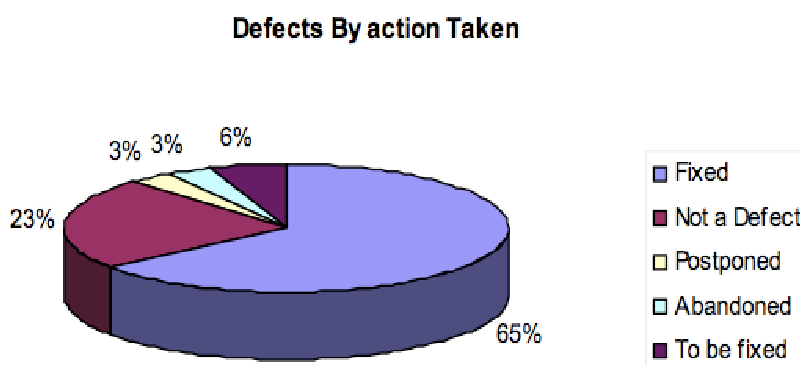


Figura 2.11: Percentuais de defeitos em função das ações tomadas (SOARES & MARTINHO)

As informações referentes aos defeitos corrigidos e pendentes de correção podem ser comparadas com os dados referentes aos testes planejados e executados. Desta forma, podemos obter um indicativo de problemas, caso o número de testes não completados aumentem ao mesmo tempo em que o número de defeitos a serem corrigidos. Com base nessa informação, e na estimativa para concluir o processo de testes, podem ser alocados mais recursos para completar o projeto de teste.

2.3.1.15 *Métricas Adicionais*

Relacionamos a seguir algumas métricas adicionais voltadas ao produto.

- Defeitos por módulo
- Defeitos por tecnologia utilizada
- Defeitos por unidade organizacional
- Defeitos por funcionário
- Defeitos por hora de introdução

2.3.2 Métricas de Processo

As métricas de processo servem para auxiliar no controle da qualidade do processo de testes.

2.3.2.1 Número de Casos de Teste

O ideal seria que todo projeto capturasse as métricas de teste relacionadas aos casos de teste listadas na Tabela 2.5, que são consideradas suficientes para a maior parte das equipes de teste que estão iniciando um programa de métricas. São métricas básicas que auxiliam no acompanhamento do processo.

Quantidade de casos de teste criados
Quantidade de casos de teste executados
Quantidade de casos de teste que passaram
Quantidade de casos de teste que falharam
Quantidade de casos de teste sob investigação
Quantidade de casos de teste bloqueados
Quantidade de casos de teste reexecutados
Tempo de execução dos casos de teste

Tabela 2.5: Exemplos de métricas de teste básicas relacionadas a casos de teste (BRADSHAW)

A relação entre o número de casos de teste concluídos e o número total de casos de teste permite que a equipe de teste acompanhe a quantidade de serviço que já foi concluída e o que ainda está pendente. O número de casos de teste executados em um determinado período de tempo também possibilita avaliar o status dos testes.

O número de casos de teste bloqueados indica os casos de teste que não puderam ser executados durante o período de teste devido a problemas na aplicação ou no ambiente.

2.3.2.2 Taxa de Falhas na Primeira Execução dos Casos de Teste

É o percentual de casos de teste que falharam na primeira execução. É usado para determinar a efetividade dos processos de análise e desenvolvimento.

A comparação dessa métrica entre projetos mostra os impactos que as mudanças no processo causam à qualidade do produto no final da fase de desenvolvimento.

2.3.2.3 Custo dos Testes

2.3.2.3.1 Custo de Testar X Custo de Não Testar

Os custos dos testes normalmente consideram os gastos com os salários dos testadores, os equipamentos utilizados, sistemas, softwares, e outras ferramentas. O custo dos testes não tem muita serventia se não for usada uma outra métrica para comparação, como, por exemplo, o custo de não testar. A questão é que estabelecer o custo de não testar pode ser extremamente difícil.

O cálculo dos custos é simples se tivermos boas métricas de projeto e pode ser baseado em um teste ou em um conjunto de testes. Normalmente são utilizadas a moeda corrente ou unidades de tempo para cálculo do custo. O cálculo posterior dos custos é uma métrica muito simples, e é bastante útil para usos futuros, em comparações para orçamentos, em fases de estimativa.

É uma métrica que possibilita mostrar à gerência o custo necessário para atender o tempo e os recursos adequados, de forma que o teste atinja um nível de cobertura ideal.

As métricas de performance auxiliam a acompanhar o custo atual do teste, assim como o esforço que ainda é necessário.

Os testes nunca encontram e removem todos os defeitos de um produto, porém reduzem o número de defeitos e o risco de problemas no produto entregue ao cliente. Se a equipe de testes tiver bons registros, há possibilidade de prever o percentual de defeitos de determinados tipos que o teste pode remover. Esse tipo de informação pode auxiliar na estimativa dos custos de não testar.

Também é possível estabelecer uma comparação entre o custo para corrigir um defeito encontrado no período de testes e o custo para corrigir o defeito quando o produto já foi entregue ao cliente. Essa análise exige um profundo conhecimento do processo de suporte ao cliente e o acompanhamento dos custos das áreas de suporte e desenvolvimento. O custo do defeito encontrado quando o sistema já está em produção inclui o custo de corrigir o defeito e o custo das pessoas e recursos ociosos ou subutilizados em função do tempo necessário para a manutenção do defeito no sistema.

2.3.2.3.2 Custo/Esforço Total de Teste X Custo/Esforço Total de Desenvolvimento

É o percentual de custo ou esforço do teste em relação ao custo ou esforço total de desenvolvimento.

Essa métrica fornece uma perspectiva do tempo gasto em desenvolvimento de um determinado projeto em relação a outros projetos. Por exemplo, se a média de tempo gasto em teste nos projetos é de 25% e o projeto em análise usou 35% do tempo em teste, pode haver indicação de uma certa ineficiência.

A métrica que relaciona a quantidade de horas gasta no desenvolvimento com a quantidade de defeitos encontrados provê uma medida do custo do processo. Os números podem variar localmente em função de ferramentas utilizadas para os testes, assim como do nível de cobertura desejado.

2.3.2.3.3 Outras Análises de Custo

O custo dos testes pode ser tratado de diversas formas. Pode ser distribuído por causa ou por módulo, pode ser calculado o custo médio para identificar um defeito, assim como pode ser feita uma comparação entre o que foi gasto com o que foi previsto.

2.3.2.4 Curva S

A Curva S é um exemplo de gráfico utilizado para a gestão dos testes, bastante útil para uso em um programa de métricas de teste, onde podem ser apresentados, por exemplo, os testes já concluídos ou o número de defeitos encontrados durante a fase de

teste. Essa curva pode ser comparada com uma curva teórica, que representa a situação ideal do processo de teste, de forma a identificar facilmente os riscos e/ou problemas envolvidos na liberação da aplicação para o cliente. Além da comparação com a curva teórica, há possibilidade de relacionar os dados com casos de teste realizados com sucesso em projetos anteriores.

Nas Curvas S temos uma representação cumulativa do esforço utilizado nas atividades do processo de teste, permitindo uma gestão objetiva, com estimativas de tempo e recursos necessários para o sucesso do projeto. Fornece um feedback visual imediato do progresso do esforço de teste, possibilitando identificar se os testes estão ocorrendo dentro do cronograma previsto, ou então, se já foram efetuados testes suficientes.

A curva teórica representa um desempenho uniforme e ideal do progresso dos testes, e é calculada da seguinte maneira.

$$\frac{(Day\ Number / Total\ Days\ in\ Test\ Effort)}{(Day\ Number / Total\ Days\ in\ Test\ Effort) + e^{(3-8 * Day\ Number / Total\ Days)}}$$

O gráfico em análise normalmente toma o formato de um S, pois o processo de execução dos testes normalmente inicia lento, aumentando o ritmo à medida que o teste continua, quando maiores defeitos são encontrados e mais questões são liberadas para teste. No final, o ritmo diminui, pois a maior parte dos defeitos já foi consertada e os casos de teste de menor prioridade são executados. Em resumo, a execução do teste inicia lenta, aumenta significativamente no meio do esforço de teste e volta a ficar lenta no final.

É aconselhável que sejam criados gráficos separados para cada objetivo, sempre mostrando a curva teórica para comparar com a curva atual. Na Figura 2.12 é apresentado um exemplo de curva S, que mostra os casos de teste que passaram, possibilitando o acompanhamento do progresso do esforço de teste.

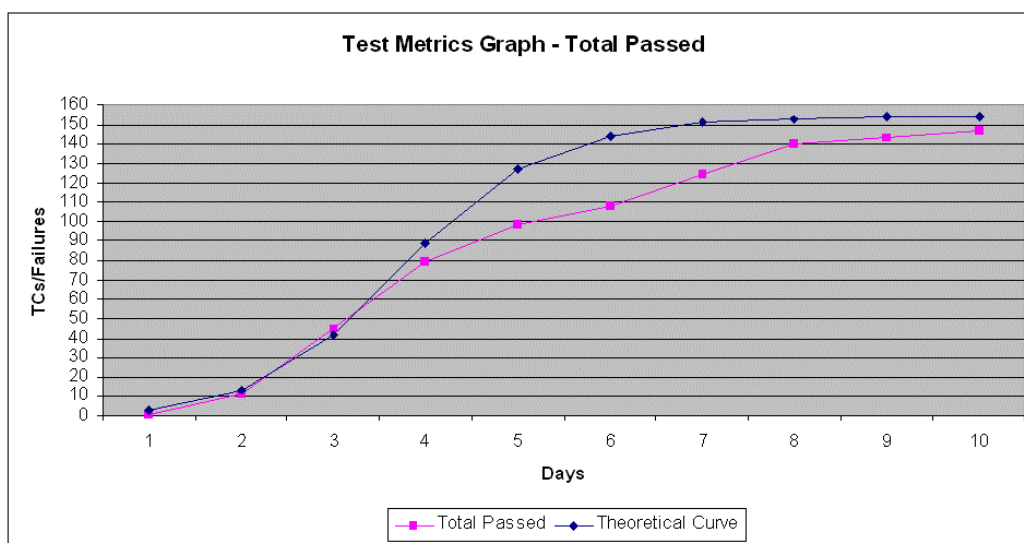


Figura 2.12: Curva S (BRADSHAW)

O gráfico da Figura 2.13 é usado para mostrar os defeitos, sendo útil na identificação de eventuais riscos na liberação do sistema.

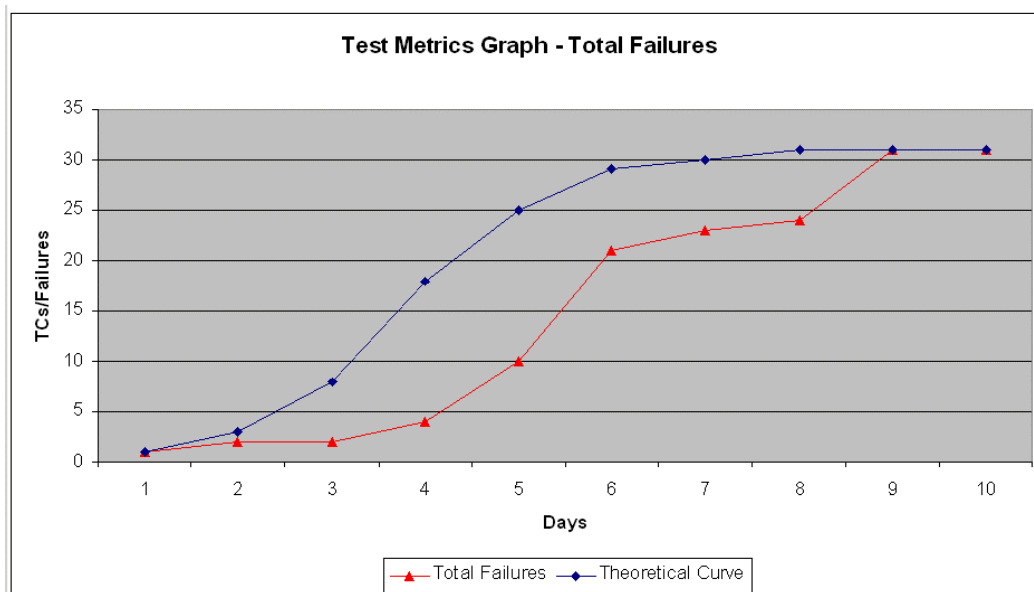


Figura 2.13: Curva S (BRADSHAW)

2.3.2.5 Curva Zero Bug Bounce

Assim como a Curva S, a *Zero Bug Bounce* constitui uma ferramenta gráfica eficaz na gestão e na monitoração dos testes, permitindo a detecção dos problemas em tempo hábil, de forma a desenvolver ações corretivas adequadas. O estado do projeto é apresentado através de métricas claras e objetivas, que auxiliam a gestão do risco.

No final de cada dia o número de defeitos pendentes de correção é representado graficamente. No final de cada fase ocorrem saltos nas quantidades de defeitos, sendo que a amplitude e a duração dos saltos sinalizam a estabilidade da aplicação. Os saltos ocorrem quando a equipe de testes descobre mais defeitos para correção. O conjunto dos picos forma a ondulação, que vai diminuindo cada vez mais em amplitude até a aplicação ser considerada suficientemente estável para ser liberada à produção.

Através da monitoração é possível determinar os defeitos que precisam ser corrigidos, quando são corrigidos, assim como, se o produto está em um nível aceitável de estabilidade para ser liberado à produção. Tal nível pode ser considerado como o ponto, nas atividades de teste, em que todos os defeitos em aberto foram corrigidos, e o ponto em que foi definida a taxa de defeitos descobertos pela equipe de testes.

As Figuras 2.14 e 2.15 mostram o gráfico *Zero Bug Bounce*.

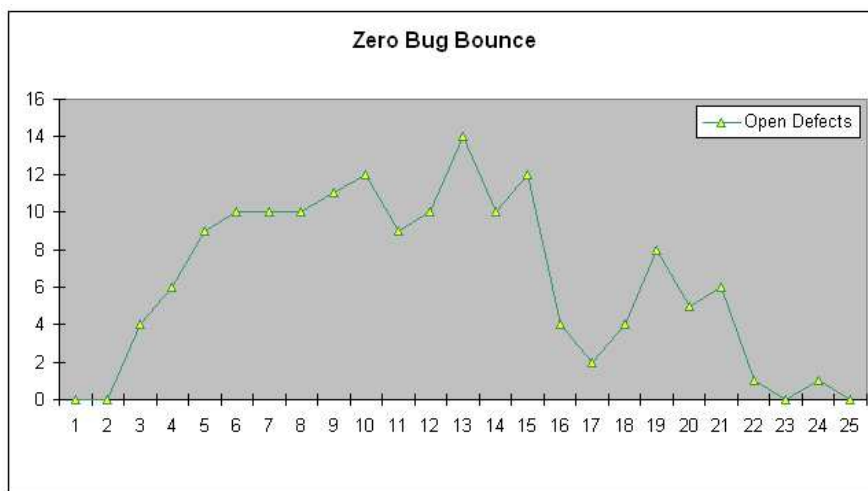


Figura 2.14: *Zero Bug Bounce* (SOARES & MARTINHO)

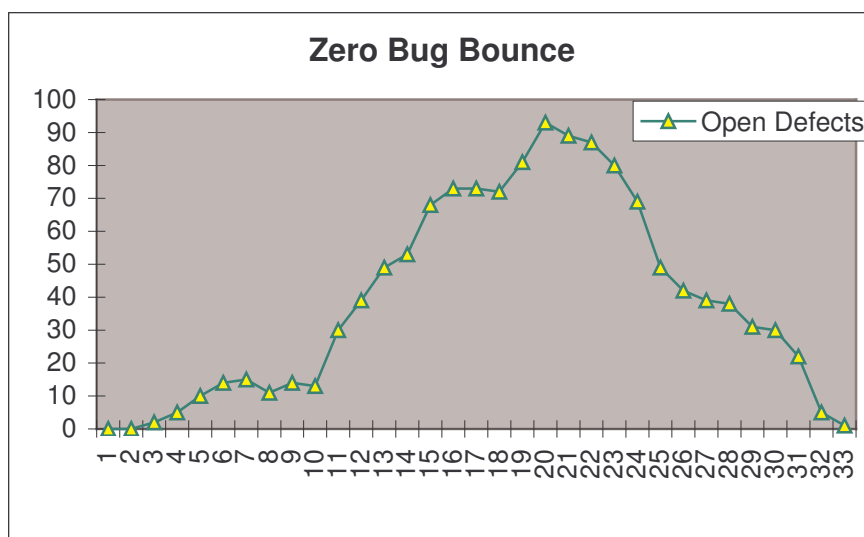


Figura 2.15: *Zero Bug Bounce* (SOARES & MARTINHO)

Na Figura 2.16 temos um exemplo de gráfico *Zero Bug Bounce* em que a figura revela uma situação com potencial de risco, em função do atraso da equipe de desenvolvimento em corrigir os defeitos detectados. Tal aplicação não poderia ser liberada para produção no prazo previsto de 10 dias, pois há uma grande possibilidade de ainda encontrarem defeitos no sistema.

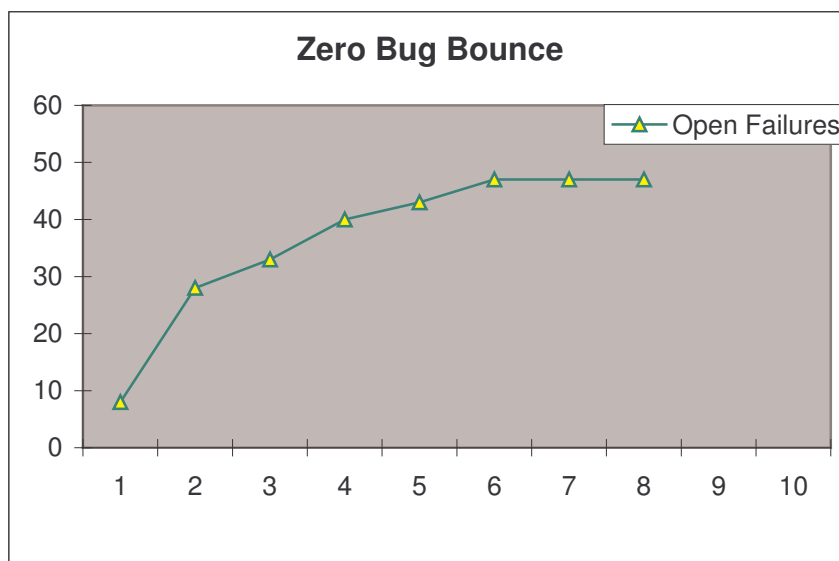


Figura 2.16: *Zero Bug Bounce* (SOARES & MARTINHO)

2.3.4.2.6 Densidade dos Defeitos Residuais

É uma estimativa do número de defeitos não resolvidos que foram para a fase de produção, pois, apesar do trabalho de remoção dos defeitos nas fases de teste, alguns poucos ainda podem permanecer.

O objetivo dessa métrica é alcançar um nível de defeitos que seja aceitável para o cliente.

2.3.2.7 Relação entre Defeitos e Ocorrências

É a relação entre o número de ocorrências que resultaram em alteração no software e o número de ocorrências apresentadas pelos testadores.

O propósito dessa métrica é apresentar uma indicação do nível de compreensão dos engenheiros de teste e dos engenheiros do software a respeito do produto, assim como, é uma informação indireta da efetividade do teste.

O cálculo é feito através da divisão entre o número de ocorrências que resultaram em alteração no software e o número total de ocorrências registradas. É válido para cada tipo de teste, durante e depois da fase de testes.

O ideal é que todas as ocorrências fossem convertidas em defeitos, com uma relação de 1:1 ou, da mesma forma, um percentual de 100%.

Na Figura 2.17 apresentamos um exemplo de gráfico com percentuais.

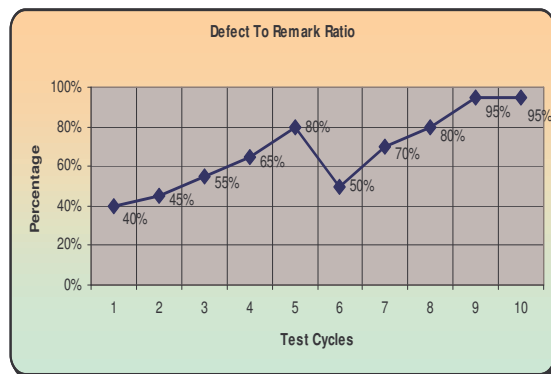


Figura 2.17: Relação entre ocorrências e defeitos (MAGAZINE)

2.3.2.8 Taxa de Ocorrências Válidas

Métrica que mostra o percentual de ocorrências válidas em um certo período de tempo, indicando a eficiência do processo de teste.

O número total de ocorrências válidas é formado pela soma do número de defeitos, das ocorrências duplicadas e do número de ocorrências que serão resolvidas na próxima fase ou release.

O cálculo é efetuado a partir da relação entre o número de ocorrências válidas e o número total de ocorrências encontradas.

2.3.2.9 Taxa de Problemas Encontrados na Correção de Defeitos

É o percentual de defeitos resolvidos que resultaram na criação de novas ocorrências na correção dos defeitos existentes.

Essa métrica mostra a efetividade do processo de correção dos defeitos, assim como fornece sinais indiretos sobre a manutenibilidade do software.

O cálculo é efetuado a partir da relação entre o número de defeitos que geraram novas ocorrências e o número total de defeitos corrigidos. O cálculo também pode ser feito por tipo de teste, fase de teste ou período de tempo.

2.3.2.10 Defeitos Encontrados X Defeitos Corrigidos

Há um mito na indústria que todos os defeitos encontrados durante a fase de testes são corrigidos antes do produto ser liberado para uso pelo cliente. Isso não acontece na prática e muitos dos defeitos entregues ao cliente são classificados como difíceis de reproduzir. Um estudo dos problemas de produção mostra que dois terços dos problemas verificados pelos usuários já haviam sido detectados no processo de teste do sistema. Tais defeitos migraram para produção porque no esforço de teste não foi possível reproduzi-los.

A questão principal desse problema é a pressão para entregar o produto dentro do prazo previsto. O risco de não atender o prazo tem maior importância do que o risco em entregar um sistema com defeitos difíceis de reproduzir ou mal compreendidos.

É questionado se há uma falha quando em um esforço de teste 98% dos defeitos são encontrados e apenas 50% corrigidos. A resposta é que, se não há uma estimativa do risco em entregar esses defeitos junto com o produto, a gerência não tem informações suficientes para decidir da maneira correta, e a pressão em entregar o produto no prazo previsto acaba sendo um fator decisivo.

O índice de defeitos encontrados e corrigidos pode ser analisado em função das versões do software, conforme exemplo apresentado na Figura 2.18.

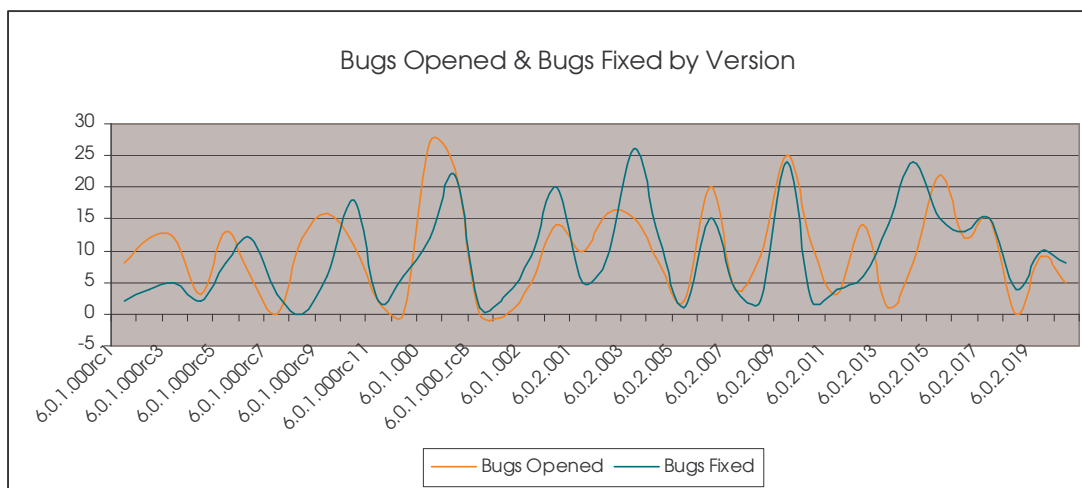


Figura 2.18: Quantidade de defeitos abertos X defeitos corrigidos por versão (CAMACHO)

2.3.2.11 Ocorrências Pendentes de Correção

Essa métrica é baseada no número de ocorrências que ainda estão pendentes de correção pela equipe de desenvolvimento. Indica como os engenheiros de software estão lidando com o esforço de teste.

O cálculo é efetuado a partir do número de ocorrências pendentes de solução.

2.3.2.12 Defeitos Encontrados X Defeitos Estimados

É a relação entre o número de defeitos encontrados durante todo o ciclo de vida do desenvolvimento e o número de defeitos estimados inicialmente. Mostra a efetividade das estimativas efetuadas, e pode ser usada também para estimar o número de defeitos nas próximas fases.

O cálculo é efetuado a partir da relação entre o número de defeitos encontrados pelo número de defeitos estimados, e pode ser usado durante e no final de cada fase.

A próxima métrica apresentada é específica para identificar a probabilidade de defeitos em um sistema.

2.3.2.13 Probabilidade de defeitos

A probabilidade de defeitos é obtida a partir da identificação de indicadores que sejam importantes para cada função, como, por exemplo, mudanças na funcionalidade desde a versão anterior, tamanho da função, complexidade e qualidade da documentação do projeto. Cada um dos indicadores recebe um índice por função, assim como um peso para administrar a diferença entre os indicadores.

A probabilidade de termos um defeito pode ser calculada por função e ser comparada com as demais funções do sistema. Tal informação deve ser tratada juntamente com a consequência do defeito para cada função.

2.3.2.14 *Ocorrências Resolvidas Que Ainda Não Foram Retestadas*

Essa métrica diz respeito ao número de ocorrências que já foram corrigidas, cujos testes precisam ser refeitos pela equipe de testes. Indica como os engenheiros de teste estão atendendo o esforço da equipe de desenvolvimento na solução dos defeitos.

O cálculo é efetuado considerando o número de ocorrências que foram resolvidas.

2.3.2.15 *Mudanças no Escopo*

É o número de alterações feitas no escopo do teste e indica a estabilidade ou a volatilidade dos requisitos e do processo como um todo.

É a relação entre o número de itens alterados no escopo e o número total de itens.

2.3.2.16 *Fase em Que o Defeito Foi Encontrado*

É um atributo do defeito que mostra a fase em que a ocorrência foi encontrada. Indica se os defeitos estão sendo encontrados nas fases corretas, conforme definido na estratégia de teste, ou seja, se não estão migrando para as fases subsequentes.

O cálculo é efetuado a partir do número de defeitos encontrados nas respectivas fases.

A Figura 2.19 apresenta um exemplo de gráfico com o percentual dos defeitos encontrados em cada fase.

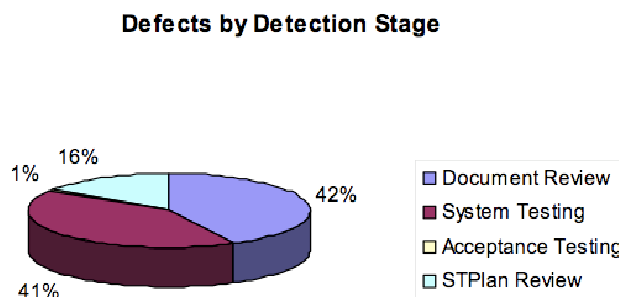


Figura 2.19: Percentual de defeitos por fase em que foram detectados

2.3.2.17 Densidade de Defeitos por Unidade

A densidade de defeitos por unidade possibilita a identificação das funcionalidades mais problemáticas. São gerados relatórios de distribuição dos defeitos por origem indicando onde os defeitos foram encontrados. Normalmente, a maior densidade dos defeitos ocorre nos módulos mais novos, assim como nos módulos experimentais. Sendo assim, é importante ter claro o fato de que altos índices de densidade de defeitos não necessariamente significam que alguém está fazendo mal o seu trabalho. É importante observar através dessa métrica que existem defeitos que precisam ser removidos.

A Figura 2.20 contém um gráfico que mostra um exemplo de concentração de defeitos em quatro módulos de um sistema durante a fase de testes. Esse tipo de gráfico é uma ferramenta bastante simples e eficiente para determinar onde é necessário concentrar recursos de desenvolvimento e teste em um projeto, para, por exemplo, possibilitar que o produto seja entregue no período previsto.

A densidade dos defeitos deve ser monitorada através do esforço de teste. Na Figura 2.21 foi considerada também a severidade dos defeitos, e, perto do final do período de testes, programadores extras foram alocados para trabalhar no módulo *Com Ap*, em função dos diversos defeitos sérios em aberto, e no módulo *Browser Ap*, por causa do grande número de defeitos em aberto.

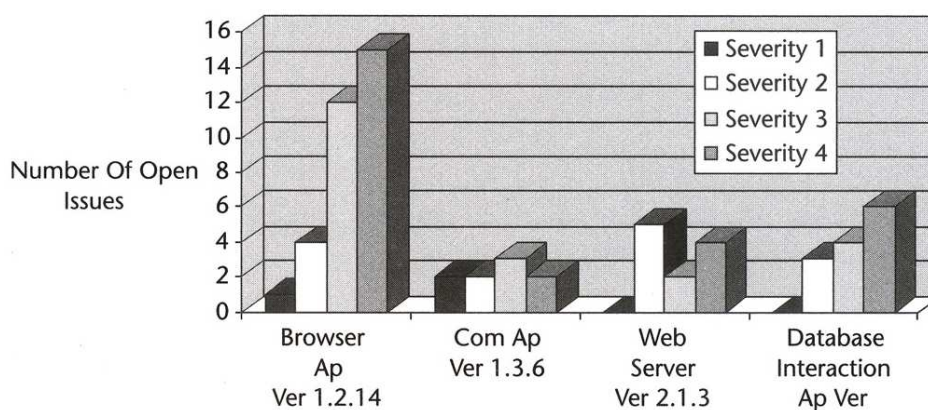


Figura 2.20: Quantidade de defeitos por módulo em que foram detectados e por severidade (HUTCHESON)

Apesar desse tipo de gráfico ser útil aos testadores, ele está entre os gráficos raramente publicados, pois há um certo temor de que esse tipo de métrica possa ser usado para prejudicar algum membro da equipe. Dessa forma, é necessário ter cautela para que a métrica não seja usada indevidamente.

2.3.2.18 Defeitos por fase em que foram injetados

Uma variação da métrica apresentada no item anterior diz respeito à fase em que os defeitos foram inseridos no sistema.

A Figura 2.21 apresenta um exemplo de gráfico com a distribuição dos defeitos por fase do processo de desenvolvimento.

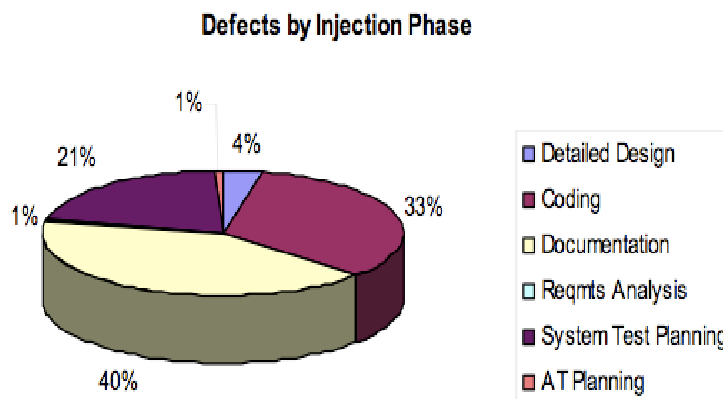


Figura 2.21: Percentual de defeitos por fase em que foram incluídos

2.3.3 Métricas de Projeto

2.3.3.1 Tempo de Teste Estimado X Tempo de Teste Efetivamente Utilizado

Essa métrica refere-se ao tempo calculado no planejamento dos testes em relação ao tempo realmente gasto nos testes. Mostra se os testes foram estimados corretamente, ou seja, fornece medidas da performance do planejamento dos testes.

O cálculo é efetuado a partir da relação entre o esforço efetivamente gasto e o planejado.

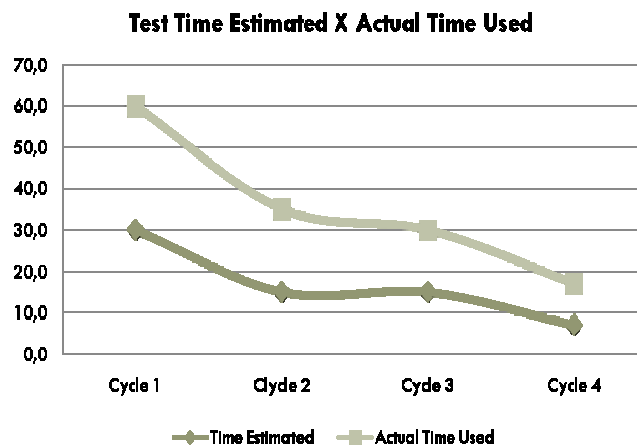


Figura 2.22: Relação entre o tempo de teste estimado e o efetivamente utilizado (CAMACHO)

Uma variação dessa métrica pode considerar o custo estimado em relação ao custo efetivo dos testes.

2.3.3.2 Fator de Segurança

Normalmente, as previsões são feitas a partir de métodos e medidas. O fator de segurança é uma métrica que mede a eficácia de uma estimativa efetuada, aplicando essa medida a uma previsão atual, para torná-la mais acurada.

Apesar dessa métrica não ser muito utilizada nos testes de software, ela foi incluída no escopo deste trabalho por considerarmos importante para a qualidade do processo, pois auxilia na estimativa adequada do tempo necessário para os testes.

Não há conhecimento de algum fator de segurança estabelecido para estimativas de testes de software. Normalmente, as gerências de projeto acrescentam alguns dias a mais na programação, que permite assimilar alguns eventos não programados. Essa prática não é aconselhável, pois é arbitrária. Os fatores de segurança devem ser determinados baseados em erros ocorridos em estimativas anteriores, efetuando os ajustes necessários. Mesmo que não haja medidas no processo para chegar a uma estimativa, é possível estabelecer um fator de segurança para estimativas futuras que sejam semelhantes.

Um exemplo de uso do fator de segurança na estimativa dos testes é um esforço de teste cuja estimativa foi de 14 semanas, e que, na realidade foram necessárias 21 semanas para conclusão do trabalho. Nesse caso, a estimativa é calculada dividindo o tempo real pelo estimado ($21/14 = 1,5$). Na próxima estimativa efetuada, considerando a utilização de métodos similares, devemos multiplicar a nova estimativa pelo fator 1,5. A partir daí, teremos uma estimativa ajustada à realidade da organização.

Nem todos os fatores de segurança são determinados analiticamente. Por exemplo, em uma determinada organização, apesar do fator de segurança estar entre 1,5 e 2, às vezes acabava ultrapassando o valor de 3,5, que era a projeção inicial de um determinado gerente. De acordo com esse profissional, cujas estimativas, na maior parte das vezes, estavam corretas, cada pessoa precisa de seu próprio fator de segurança, e, normalmente, ele utilizava a própria experiência para calcular.

Em suma, o que importa não é como o fator de segurança é determinado, e sim que sua utilização auxilia no aperfeiçoamento das estimativas. Ninguém conhece tudo e nenhum método é perfeito. Não há vergonha em produzir uma estimativa que é inicialmente inadequada. O importante é reconhecer as deficiências e poder corrigi-las antes que venham a se tornar um problema.

Normalmente os fatores de segurança são rejeitados por muitas gerências, pois querem estimativas de tempo mais curtas. Sendo assim, para poder persuadir a gerência para o seu uso é necessário apresentar bons resultados nos cálculos efetuados, que possibilitem a tomada correta de decisões gerenciais.

2.3.3.3 Tempo Necessário para Executar um Teste

Essa medida é uma métrica fundamental, necessária para estimar o tempo necessário para a execução dos testes planejados, assim como o *setup* dos testes e atividades de *cleanup*, que também devem ser considerados. Esses itens podem ser incluídos como parte do tempo necessário para o teste ou separadamente.

Teoricamente, a soma de tempo necessário para executar todos os testes planejados é importante para estimar o tamanho global do esforço de teste, mas também deve ser considerado o número de tentativas efetuadas para a execução do teste de forma adequada e confiável.

O tempo geralmente é calculado em minutos ou horas gastas por teste, assim como o número de horas necessárias para concluir um conjunto de testes.

2.3.3.4 Tempo Disponível para o Esforço de Teste

O tempo disponível para o esforço de teste é uma das métricas de teste mais firmemente estabelecida e publicada. O valor deve estar sempre decrescendo.

A estimativa do tempo é efetuada em semanas e normalmente medida em minutos.

2.3.3.5 Taxa de Esforço de Teste

O esforço gasto em todo o projeto de software inclui diversas fases do projeto como, por exemplo, requisitos, projeto, codificação e testes. A taxa de esforço de teste é o esforço gasto especificamente para o teste em relação ao projeto como um todo.

O objetivo na captura desse tipo de métrica é obter indicações no nível de investimentos necessários para os testes. Além disso, essa métrica pode ser útil para estimar projetos futuros similares.

O cálculo é efetuado a partir da divisão do esforço total do teste pelo esforço total do projeto.

2.3.3.6 Categoria dos Defeitos

A categoria do defeito é um atributo do mesmo em relação aos atributos de qualidade do produto.

Essa métrica pode ser calculada dividindo os defeitos que fazem parte de uma determinada categoria pelo número total de defeitos. As informações obtidas podem ser utilizadas para auxiliar na tomada de decisões sobre o andamento do processo de testes e onde focar o tempo da equipe de testes, assim como determinar os aspectos do processo que precisam ser aperfeiçoados.

As falhas verificadas nos testes dizem respeito a defeitos no software. Esses defeitos devem ser analisados para que se possa determinar quando foram introduzidos no software.

Tomemos por exemplo situações em que existem muitos erros em função de uma documentação pobre. Nesse caso, o uso da métrica pode indicar tais problemas, mostrando que os erros são provenientes de uma causa comum, que pode ser atacada de forma mais eficiente.

Um outro enfoque dessa métrica é analisar os erros por subsistema. Se um número pequeno de subsistemas apresentar alto número de erros, isso pode ser um indicativo de que tais áreas precisam ser mais bem gerenciadas.

A Figura 2.23 mostra um exemplo de gráfico que distribui os defeitos por causa.

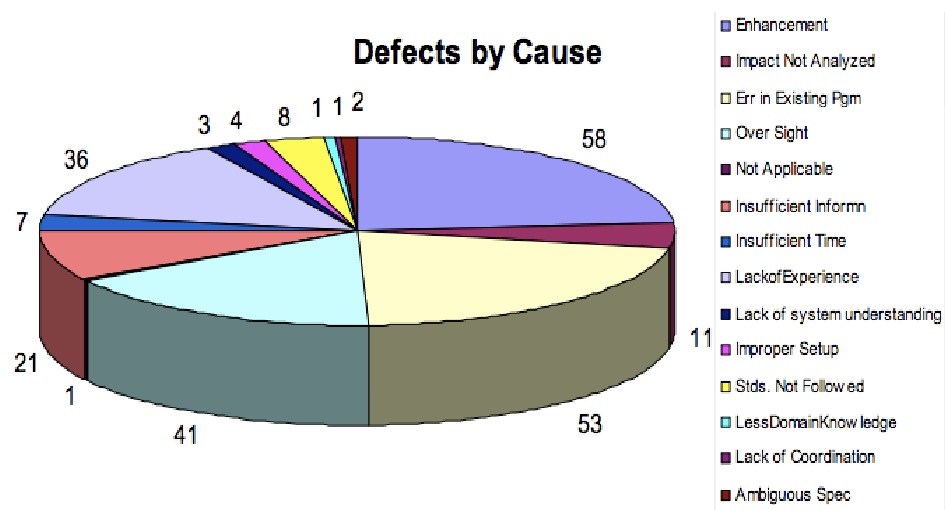


Figura 2.23: Distribuição dos defeitos por causa

3 ADOÇÃO DE MÉTRICAS: DIAGNÓSTICO EM EMPRESAS DE TI DO RIO GRANDE DO SUL

O objetivo principal deste trabalho é investigar o uso de métricas de teste de software através da apresentação de conceitos e exemplos que relacionam o custo e o benefício das métricas. Essa necessidade é baseada no fato de que o uso de métricas de teste de software não é uma prática usual nas empresas gaúchas de desenvolvimento de software, conforme pudemos constatar em pesquisa efetuada junto a diversas empresas do ramo.

Através da pesquisa, foram exploradas várias questões referentes ao teste de software, apesar de o objetivo final serem as métricas de teste de software. As áreas investigadas na pesquisa foram:

- Informações referentes à organização
 - Nome da empresa
 - Nome e função do funcionário responsável pelo preenchimento do questionário
- Metodologias de teste de software
 - Equipe de teste
 - Documentação do processo de teste
 - Momentos em que o teste é executado dentro do processo de desenvolvimento do software
 - Níveis e tipos de testes
 - Catálogo de erros
- Ferramentas de automação de teste de software
- Educação e treinamento em teste de software
- Métricas de teste de software
 - Tipos de métricas e estimativas usadas
 - Barreiras que levam à não utilização de métricas

O questionário foi criado em um formulário do *GoogleDocs*, conforme APÊNDICE A, sendo a solicitação para preenchimento encaminhada via e-mail a 178 empresas gaúchas que desenvolvem software, ao mesmo tempo em que foi enviada aos usuários do Grupo de Usuários de Testes de Software (GUTS) – Sociedade dos Usuários de

Informática e Telecomunicações do RS (SUCESU). As empresas gaúchas que desenvolvem software foram identificadas na Internet, através de buscas no *Google*.

A coleta de dados da pesquisa durou aproximadamente três meses. No primeiro mês, houve uma quantidade pouco significativa de respostas, sendo necessário o reenvio do e-mail às empresas, buscando sensibilizá-las sobre a importância do trabalho. Após reiteradas tentativas, obtivemos resposta de 30 empresas, que foi considerada uma amostragem significativa para o objetivo do trabalho.

Os gráficos nas Figuras 3.1, 3.2, 3.3, 3.4, 3.5 e 3.6 mostram o resultado de alguns itens específicos do resultado da pesquisa, sendo que a apresentação completa, gerada no *GoogleDocs*, pode ser consultada no Apêndice B.

A Figura 3.1 diz respeito à utilização de testes de software nas empresas, e ratifica a ideia de que a rotina de testes já está bastante difundida, pois apenas quatro das 30 empresas consultadas não trabalham com testes, e uma grande parte delas utiliza testes há mais de três anos.

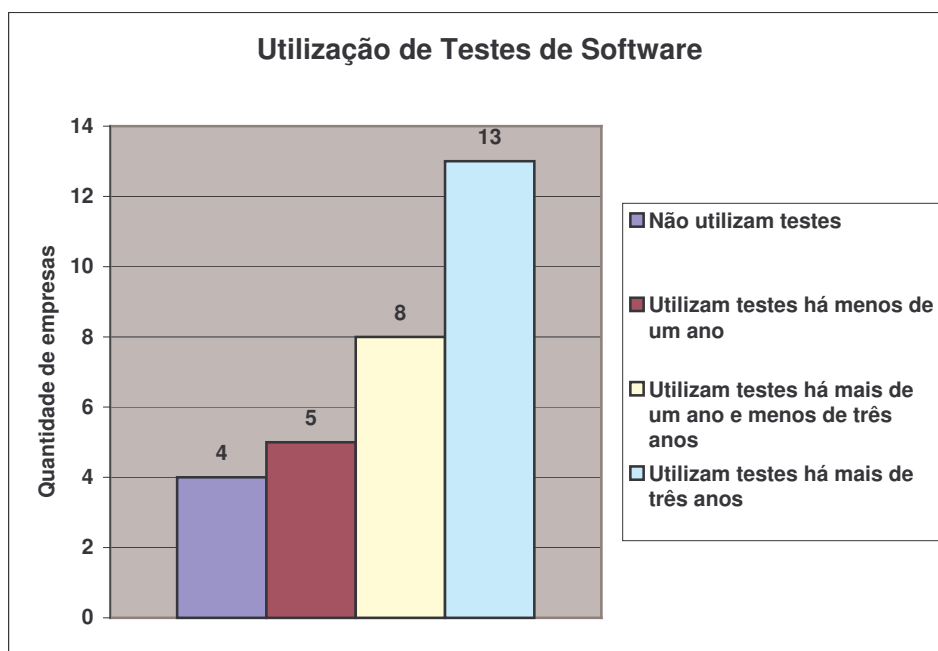


Figura 3.1: Utilização de testes de software

Uma das questões importantes que foi avaliada através da pesquisa, e que é o foco principal deste trabalho, é o fato que, embora a maioria das empresas já use teste de software, 59% delas ainda não trabalham com métricas de teste, em função das barreiras apresentadas no gráfico da Figura 3.2. É um percentual significativo, considerando a importância das métricas de teste dentro do processo de desenvolvimento de software.

A Figura 3.2 mostra que duas empresas não usam métricas porque há falta de informações sobre recursos disponíveis. O custo, a dificuldade no uso e a ideia de que as métricas não são úteis ou que o custo/benefício não compensa, foram citados apenas uma vez como barreiras para o uso das métricas de teste, enquanto o consumo de tempo e a falta de conhecimento sobre métricas foram apontados como barreiras por quatro empresas. Por fim, temos quatro empresas que alegam não ter barreiras. Esta informação pode ter diversas interpretações, porém, a mais coerente de todas, é que

essas empresas nunca tiveram interesse em usar métricas por falta de conhecimento do assunto, o que nos leva mais uma vez ao objetivo principal desse trabalho, que é motivar o uso de métricas de teste através da apresentação dos benefícios das mesmas, e de informações e orientações para utilização. Apesar de serem 15 as empresas que não usam métricas de teste, as barreiras apresentadas no gráfico somam 19, pois uma mesma empresa pode marcar mais de uma opção na pesquisa.

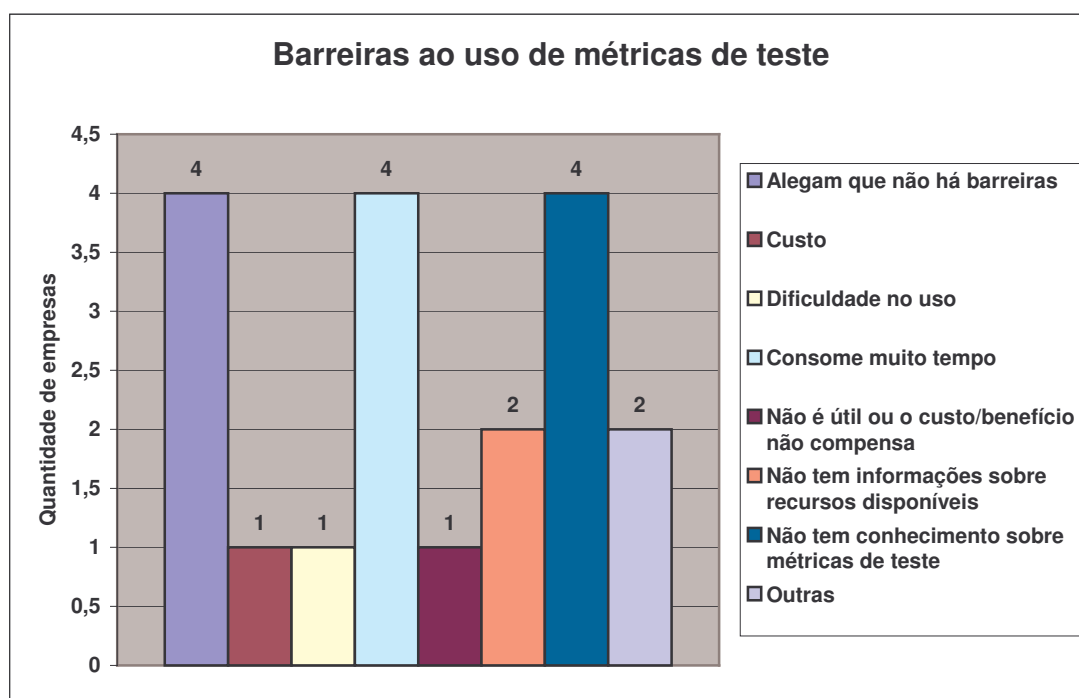


Figura 3.2: Barreiras ao uso de métricas de testes de software

A Figura 3.3 apresenta as métricas de teste mais utilizadas nas empresas. Das 11 empresas pesquisadas, e que utilizam métricas de teste, nove informaram que utilizam a métrica que avalia a quantidade de defeitos encontrados e corrigidos, oito empresas usam o índice de densidade dos defeitos, e seis utilizam a métrica que possibilita verificar a efetividade dos casos de teste. As métricas referentes ao índice de severidade dos defeitos, da distribuição dos defeitos por funcionalidade e a cobertura de teste foram selecionadas por cinco empresas. Da mesma forma que na Figura 3.2, apesar de serem 11 as empresas que usam métricas de teste, as métricas apresentadas no gráfico somam 39, pois uma mesma empresa pode marcar mais de uma opção na pesquisa.

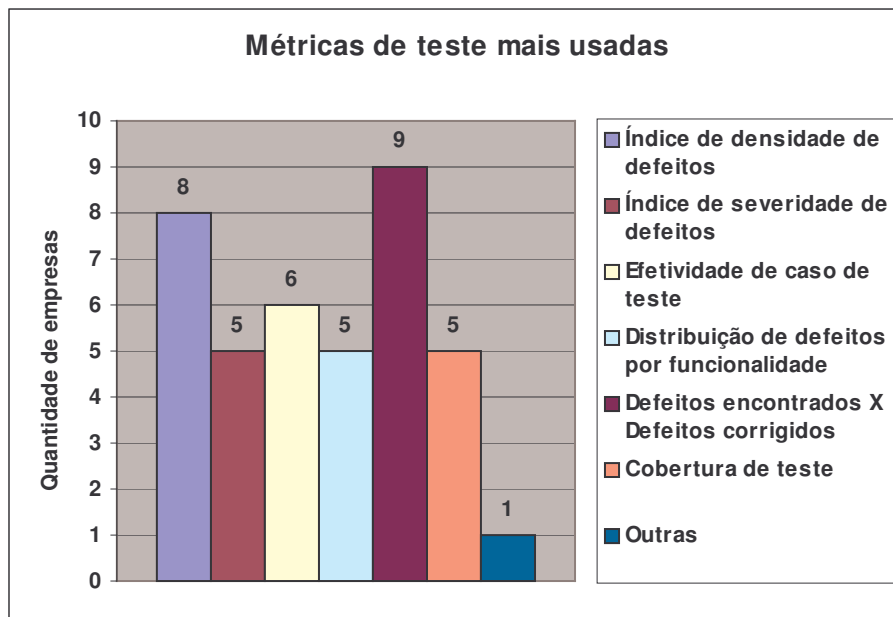


Figura 3.3: Métricas de testes de software mais usadas

A pesquisa mostrou também a utilização de estimativas para os testes de software, conforme gráfico na Figura 3.4. As métricas de projetos anteriores para estimar o esforço do teste são usadas por dez empresas. As métricas para medir o progresso do teste são utilizadas por oito empresas, para poder estimar o prazo de conclusão dos testes, ou estimar se os testes serão concluídos na totalidade, caso seja mantido o prazo inicial estabelecido. A empresa que usa uma outra estimativa, além das especificadas, respondeu que se baseia no conhecimento empírico do trabalho dos testadores para estimar o tempo necessário para os testes de um determinado projeto. Nesse caso, a empresa usa também as métricas de projetos anteriores para embasar as estimativas, assim como a experiência da equipe para estimar o tempo necessário para os testes.

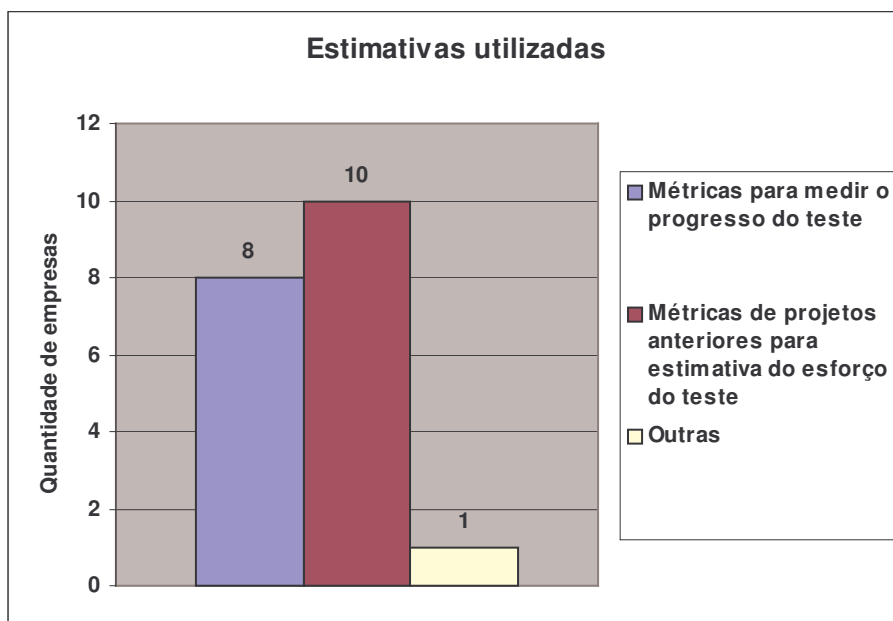


Figura 3.4: Gráfico das estimativas utilizadas

No gráfico apresentado na Figura 3.5, podemos verificar o tempo em que as empresas que utilizam métricas de testes trabalham com testes de software. Das 11, cinco delas trabalham com testes há mais de três anos; duas empresas, há menos de um ano; e quatro empresas no intervalo de um a três anos. A partir desta informação, podemos observar que o uso de métricas de teste de software não tem relação com o tempo que a empresa trabalha com testes, pois, por exemplo, há duas empresas que trabalham com métricas e utilizam testes de software há menos de um ano.

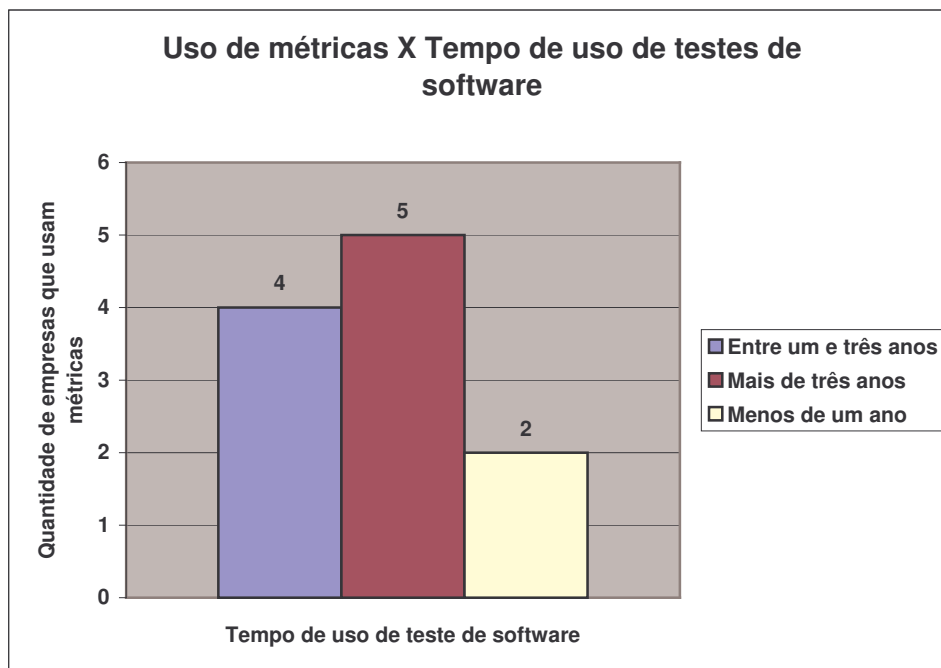


Figura 3.5: Relação entre o uso de métricas de teste e o tempo de uso de teste de software

Uma outra questão a ser analisada é que o uso de métricas de teste não está relacionado ao fato de que os profissionais da equipe de teste têm formação específica na área de testes. No gráfico da Figura 3.6, que mostra as 11 empresas que trabalham com métricas de teste de software, temos a informação de que em sete empresas apenas alguns profissionais têm especialização em testes, em duas empresas todos os profissionais têm formação, e nas duas últimas empresas os profissionais não têm nenhuma especialização em testes de software. O fato de que em duas empresas são utilizadas métricas de teste, apesar de os profissionais não terem formação específica em testes, nos leva a questionar a complexidade atribuída às métricas por algumas empresas, como barreira para o uso.

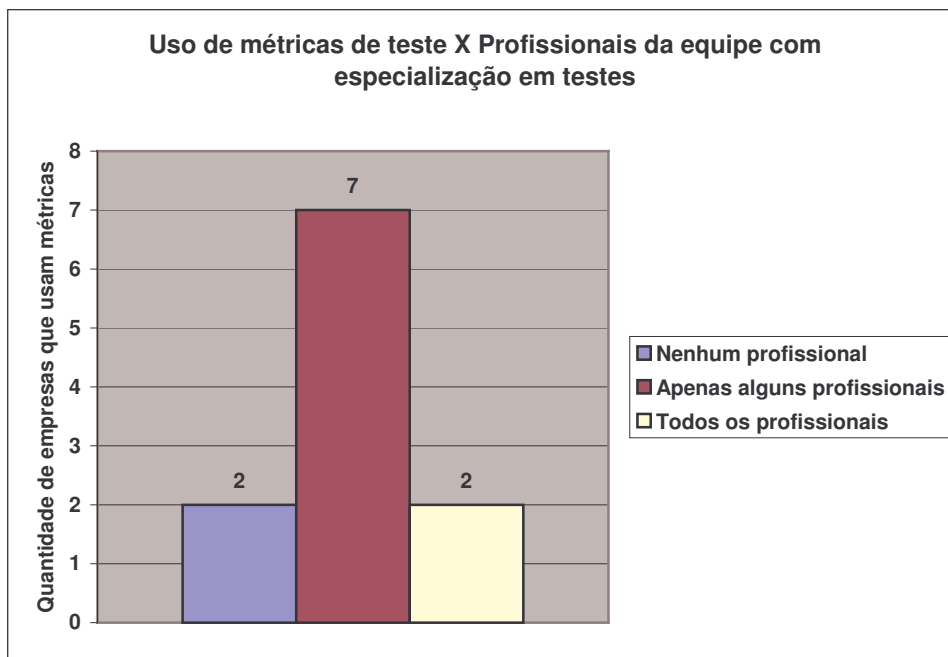


Figura 3.6: Relação entre o uso de métricas de teste e a especialização da equipe de testes

A pesquisa demonstra que, antes de tentarmos implementar algum tipo de melhoria nas métricas de teste de software, é necessário expandir o uso básico das métricas ao maior número possível de empresas desenvolvedoras de software. Analisando as informações obtidas, podemos supor que a razão para que as métricas de teste não sejam tão utilizadas seja a falta de conhecimento, tanto dos benefícios das métricas como da utilização das mesmas, já que, não necessariamente, ocorre em função do pouco tempo do uso de testes nas empresas ou à formação específica na área de testes da equipe. Considerando tais fatores, precisamos apresentar da maneira mais simplificada possível as métricas existentes, os benefícios e as dificuldades inerentes à implantação e ao uso das mesmas.

4 USO DE MÉTRICAS DE TESTE NA PRÁTICA

A essência dos testes é encontrar e fornecer informações. Os resultados do esforço de teste são utilizados pelos *stakeholders* para verificar o andamento do projeto. Os executivos utilizam as informações dos testes para tomar importantes decisões sobre o produto. Em função disso, muitas organizações investem em equipes independentes de teste. A questão a ser analisada, nesse caso, é a forma como as informações são apresentadas, pois em muitos casos auxiliam na tomada de decisões críticas. Por exemplo, em um projeto que esteja a dois dias do prazo para entrega ao cliente, em que são reportadas apenas informações referentes aos defeitos em aberto e às tendências dos defeitos, esquecendo de métricas importantes como cobertura dos testes, decisões podem ser tomadas sem conhecimento suficiente dos fatos (MAGAZINE, 2008).

Na Tabela 4.1, temos um exemplo de uso de métricas de teste em uma empresa, que demonstra como as métricas podem prover informações para o entendimento dos tipos de mudança no processo que podem melhorar a qualidade e reduzir os custos dos projetos (BRADSHAW, 2007).

Tabela 4.1: Exemplo de uso de métricas de teste

Contexto	<ul style="list-style-type: none"> - Departamento de TI de uma fábrica de caminhões - Tem pouco ou quase nenhum teste nos seus projetos - Ambiente de desenvolvimento em um <i>mainframe</i> COBOL/CICS
Objetivos	<ul style="list-style-type: none"> - Migrar para uma tecnologia mais avançada - Estabelecer um processo de testes - Treinar os novos membros que farão parte da equipe de testes
Ações	<ul style="list-style-type: none"> - Um programa de métricas de teste foi criado juntamente com a inclusão de testes no departamento. - A equipe de testes foi orientada quanto às métricas de teste a serem capturadas.
Primeiro Projeto (V)	<ul style="list-style-type: none"> - Desenvolvido em <i>Visual Basic</i> e HTML, com acesso através da <i>web</i>, com um <i>browser</i> padrão. - Criados 355 casos de teste. - A taxa de falha na primeira execução foi de 30,7%. - A taxa de falha geral foi de 31,4%. - O custo para corrigir os defeitos foi de \$519,000.
Segundo Projeto (T)	<ul style="list-style-type: none"> - Foram criadas revisões nos requisitos e nas especificações, pois quanto mais cedo a equipe de teste se envolve no processo, maior

	<p>é a melhoria no processo de desenvolvimento.</p> <ul style="list-style-type: none"> - O XML passou a ser utilizado também no ambiente de desenvolvimento, o que causou aumento na complexidade do processo. - Criados 345 casos de teste. - As equipes permaneceram as mesmas do Projeto V. - A taxa de falha na primeira execução foi de 17,9%. - A taxa de falha geral foi de 18%. - O custo para corrigir os defeitos foi de \$346,000.
Conclusões	<ul style="list-style-type: none"> - Houve melhoria significativa nas taxas de falha no Projeto T em relação ao Projeto V. - A redução da taxa de falha na primeira execução foi 41,7%. - A redução da taxa de falha geral foi de 42,7%. - A redução na taxa de falha geral gerou uma redução de custos de aproximadamente \$170,000 (33,3%), em função da redução dos custos com o retrabalho.

FONTE: BRADSHAW.

Das informações apresentadas na Tabela 4.1, concluímos que o uso de métricas de teste bem definidas para auxiliar no esforço de teste pode melhorar significativamente a forma de relatar os defeitos encontrados, com objetividade. A equipe e os gerentes de teste têm bastante dificuldade em comunicar de forma empírica aos gerentes e à equipe de projeto, ou outras partes interessadas, os impactos causados por atrasos, defeitos, ou mudanças de escopo.

Um outro exemplo selecionado, referente ao uso de métricas, e que é bastante utilizado para a gestão dos testes, é o gráfico com a Curva S. Na Figura 4.1, temos um gráfico em que o progresso dos testes não está adequado, pois a quantidade de casos de teste concluídos não está aumentando com o passar do tempo, ou seja, não está acompanhando a tendência da curva teórica. Existem diversas causas para esse tipo de problema, como muitos testes bloqueados, em função de defeitos que impedem a realização de vários casos de teste, realocação de recursos para outras tarefas no meio do processo e alteração na equipe de testes. Entre as ações para solucionar o problema, sugerimos a reavaliação da prioridade de execução dos casos de teste e a utilização de mais recursos ou a formação de uma equipe mais experiente.

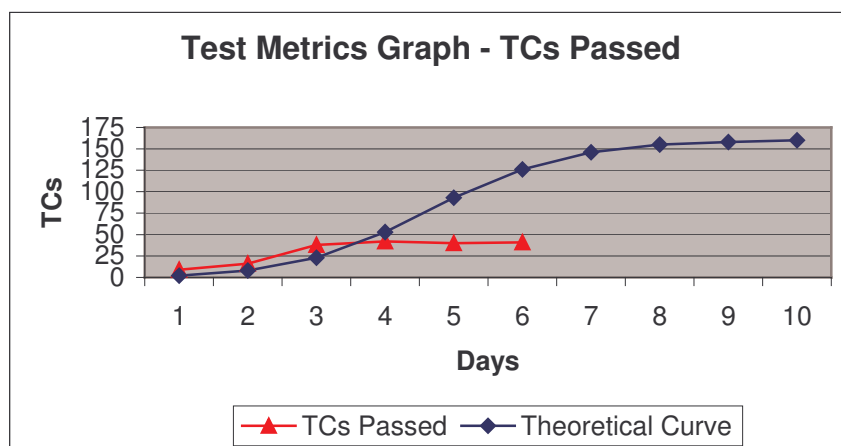


Figura 4.1: Exemplo de Curva S (BRADSHAW).

Um segundo exemplo de Curva S é apresentado na Figura 4.2. O gráfico retrata a quantidade de defeitos em função do tempo. A curva teórica mostra que a tendência é que haja um aumento gradual da quantidade de defeitos no início do processo de testes, estabilizando no final. No exemplo, houve um aumento abrupto da quantidade de defeitos no início do processo. As causas prováveis para a ocorrência são o erro de estimativa e a liberação de defeitos corrigidos que ainda têm problemas para serem resolvidos. As sugestões de correção para esse caso são uma reavaliação das estimativas e o aumento do esforço em testes unitários ou revisões de código.

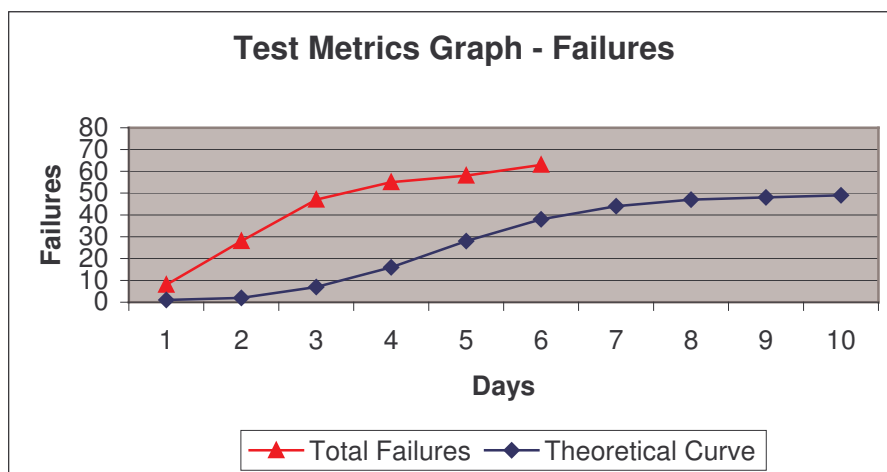


Figura 4.2: Exemplo de Curva S (BRADSHAW).

O próximo exemplo de uso de métrica de teste é referente à performance do teste e auxilia a responder à pergunta “Vale a pena testar?” (HUTCHESON). A boa performance do esforço de teste deve ser confirmada, e é medida no final do processo, através dos dados capturados durante todo o período pelos testadores. Além desta confirmação, é necessário mostrar que os testes adicionam valor ao processo, para justificar os gastos. O objetivo do esforço de teste é entregar o produto ao cliente com o menor número de defeitos possível, encontrando e removendo os defeitos antes do produto ser liberado para uso em produção. A performance do esforço de teste é calculada através da relação entre o número total de defeitos encontrados e corrigidos durante os testes e todos os defeitos verificados no sistema.

Na Tabela 4.2, são relacionadas métricas referentes a dois casos de estudo conduzidos no mesmo ambiente. Analisando a média de chamadas do usuário nos primeiros 90 dias de uso da aplicação, concluímos que o caso 1 foi mais eficiente do que o caso 2, pois foram apenas 5 chamadas em comparação às 30 chamadas do caso 2. Este exemplo mostra que não podemos analisar as métricas isoladamente, como, por exemplo, a cobertura dos testes que no caso 2 é de 100%, enquanto que no caso 1 é de apenas 67%. Neste caso, provavelmente o inventário de testes do caso 2 é insuficiente, além da baixa taxa de correção dos defeitos, que foi apenas de 50%. É importante diferenciar se o esforço de teste como um todo foi suficiente, ou se a cobertura de testes foi suficiente, pois os defeitos podem ter sido encontrados através de um conjunto de testes, mas não corrigidos durante o esforço de teste. O número de defeitos sérios (severidade 1) reportados em produção, o custo do esforço de teste e o custo dos defeitos encontrados em produção também foram considerados para avaliar se o esforço do teste foi adequado.

As medidas são utilizadas para fazer previsões. Os resultados são medidos para verificar se as previsões estavam corretas e fazer ajustes para as próximas estimativas. No caso do exemplo da Tabela 4.2, as informações serão usadas futuramente para ajustar a cobertura dos testes e os requisitos para correção dos defeitos.

Tabela 4.2: Determinando se o esforço de teste foi adequado.

Caso de estudo	Cobertura dos testes	Média de chamadas do usuário nos primeiros 90 dias	Taxa de correção dos defeitos	Taxa de performance nos seis primeiros meses após liberação ao usuário	Defeitos de severidade 1 reportados em produção	Defeitos de severidade 2 reportados em produção
1	67%	5	70%	98%	0	6
2	100%	30	50%	75%	7	19

FONTE: HUTCHESON.

No capítulo 5, apresentaremos um exemplo de uso de métricas de teste em uma empresa da cidade de Porto Alegre, Rio Grande do Sul, com o objetivo de trazer de forma simples e detalhada a evolução do programa de métricas dentro de um determinado setor da área de TI.

4.1 Implementação de um Programa de Métricas de Teste de Software

Existem diversos fatores relevantes para estabelecer um programa de métricas de teste, tais como, a definição do objetivo para o desenvolvimento e a captura das métricas, a identificação e definição das métricas úteis para serem coletadas, e análise do resultado das métricas para efetuar mudanças no processo (BRADSHAW).

Existem diversas métricas de teste, e, em função disso, as organizações têm dúvidas na hora de escolher as métricas a serem utilizadas. As métricas podem servir para diferentes propósitos. A título de ilustração, para avaliar o trabalho da equipe de testes podemos utilizar métricas como o número de defeitos encontrados, número de casos de teste criados, ou o número de testes executados em um determinado período (PIROZZI).

As métricas de teste podem ser aplicadas em diversos estágios do projeto, porém isso depende do foco das métricas, da maturidade dos dados de métricas de teste da organização e do tipo de projeto (BRADSHAW).

O ciclo de vida do processo de métricas de teste é formado pelos seguintes passos, ilustrados na Figura 4.3, que garantem uma padronização adequada do processo (SOARES & MARTINHO).

- Identificar os processos chave que são mensuráveis
- Identificar as métricas de teste que são relevantes
- Definir e classificar as métricas
- Identificar e refinar os mecanismos de captura dos dados
- Comunicar o processo à equipe
- Capturar e verificar os dados
- Analisar e processar os dados
- Reportar as métricas de teste
- Melhorar os processos

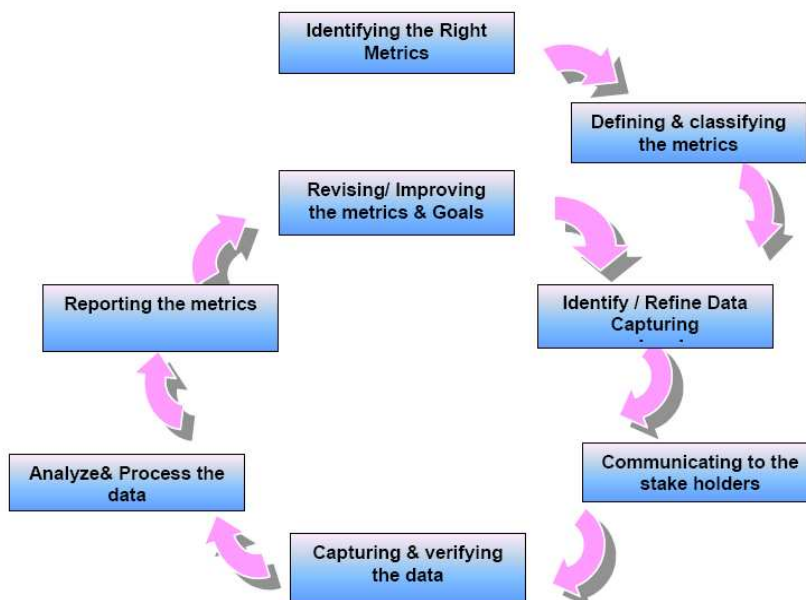


Figura 4.3: Ciclo de vida do processo de métrica de teste (SOARES & MARTINHO).

Apresentaremos alguns pontos que devem ser observados na definição de um programa de métricas de teste e na monitoração dos benefícios obtidos a partir das melhorias no processo de testes (BRADSHAW; BORYSOWICH, 2006).

4.1.1 Identificação das Métricas de Teste

Usualmente, as métricas de teste são identificadas no início do projeto, e devem ser quantificáveis, fáceis de coletar, simples, significativas e não ameaçadoras.

O processo de identificação das métricas deve ser iniciado listando os problemas a serem resolvidos e os objetivos. Na sequência, devem ser determinados os itens a serem medidos e os padrões de medida utilizados, de forma a atingir os objetivos. Deve ser avaliada a relação custo x benefício de cada métrica.

Os requisitos das métricas de teste, formulados durante a identificação das métricas, devem ser documentados, assim como o mecanismo definido para coletar as informações das métricas de teste, conforme exemplo na Tabela 4.3. Os requisitos consistem no objetivo, nas métricas de teste a serem capturadas, em como as métricas fornecem as informações necessárias, e nas definições das métricas de teste.

O exemplo da Tabela 4.3 pode ser usado como um guia para especificar o processo de métricas de teste e uma ferramenta para treinar a equipe que provê os dados.

Tabela 4.3: Requisitos das métricas de teste

Requisitos das Métricas de Teste		
Objetivo	Determinar as tendências e as causas dos erros a partir dos defeitos encontrados durante o teste de sistema, para identificar os defeitos mais comuns e minimizar tais defeitos no projeto atual e em projetos futuros.	
Métricas a coletar	<ul style="list-style-type: none"> - Número de defeitos - Origem de cada defeito - Origem de cada defeito de código 	
Como determinar	Dividir o número de defeitos para cada tipo de origem de defeito pelo número total de defeitos. Para os defeitos de código, dividir o número de defeitos de cada subtipo pelo número total de defeitos de código.	
Definições	Defeito	Uma falha que causa o funcionamento incorreto ou incompleto do sistema. Apenas defeitos identificados como problemas reais. Erros cosméticos e de usabilidade não são incluídos.
	Origem do defeito, especificada através dos códigos de defeitos relacionados abaixo.	Área em que ocorreu o defeito especificado através dos códigos de defeito.
	Especificação Funcional	Código para especificação funcional, quando a função descrita

		nas especificações estiver incorreta, não clara ou incompleta.
	Projeto	Código para o projeto, quando estiver incorreto, não claro ou incompleto.
	Arquitetura	Os componentes da arquitetura selecionados não estão conforme o esperado ou o planejado.
	Interface com usuários	Código para especificação, quando a interface com os usuários descrita nas especificações, ou seja, telas, relatórios, entre outros, estiver incorreta, não clara ou incompleta.
	Projeto de banco de dados	Resultados inesperados devido ao projeto do banco de dados. Por exemplo, definição incorreta de chave primária ou tipo de dado incorreto.
	Erro de código, especificado através dos subtipos relacionados abaixo.	O código não está de acordo com as especificações detalhadas no projeto.
	Erro lógico	A lógica está incorreta, não clara ou incompleta.
	Erro de cálculo	O cálculo está incorreto, não claro ou incompleto.
	Interface entre unidades de programas	Quando o controle é recebido ou dado a uma entidade externa ao programa, um procedimento está incorreto, não claro ou incompleto, resultando em transferência incorreta de dados.
	Manipulação de dados	Mecanismo de manipulação de dados, como declaração de dados ou estrutura de dados incorreta, não clara ou incompleta.
	Tratamento dos erros	Um procedimento de tratamento dos erros incorreto, não claro ou incompleto.
Mecanismo de coleta	Os programadores frequentemente especificam a origem do erro no relatório de defeitos quando corrigem os erros identificados no teste de sistema. Adicionar subtipos para os erros de código no formulário de relato das falhas.	

Fonte: BRADSHAW.

O modelo *Goal Question Metric* (GQM) é bastante útil na identificação das métricas de teste, pois é uma abordagem sistemática para integrar os objetivos no processo. É baseado na idéia de que deve existir uma necessidade clara associada a cada métrica. O modelo GQM consiste nos seguintes passos (SOARES & MARTINHO):

- Identificar as pessoas interessadas na medição.
- Selecionar alguns objetivos do projeto ou da organização.
- Expressar os objetivos da maneira mais quantitativa e mensurável possível.
- Descobrir o que deve ser feito para atingir cada um dos objetivos. Encontrar a pergunta que verifica se o objetivo foi atingido.
- Definir o que deve ser medido para quantificar o progresso, ou seja, identificar a métrica que permite responder cada uma das questões.

Principalmente no início da implantação de um programa de métricas, os objetivos não devem ser muito complexos e de longo prazo, sob pena de causar impacto na motivação da equipe. Com o passar do tempo e com a maturidade da organização para o uso de métricas, os objetivos podem se tornar mais desafiadores (MARINHO).

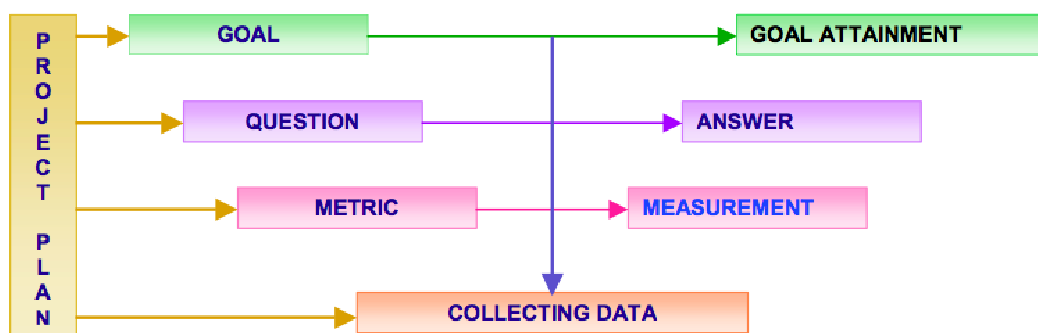


Figura 4.4: Modelo *Goal Question Metric* (GQM) (SOARES & MARTINHO).

Exemplo de uso do Modelo GQM (CAMACHO):

- Meta – Ter todos os defeitos corrigidos antes do software ser liberado para uso.
- Perguntas – Quantos defeitos temos atualmente? Qual é a cobertura dos testes? Qual é a efetividade dos casos de teste?
- Métricas selecionadas – Índice de defeitos, Cobertura dos testes e Efetividade dos casos de teste.

4.1.1.1 Quantificável

Os métodos utilizados para as medições devem ser concisos, quantificáveis e devem estar dentro de um padrão. Para determinar a densidade dos defeitos, por exemplo, temos que identificar as métricas para obter tal informação e o padrão adotado para medida. A métrica de teste é o número de defeitos, e deve ficar claro o que constitui um

defeito. O método de medição é a quantidade de linhas de código e deve obedecer a um padrão, como 1.000 linhas, por exemplo. Além disso, para garantir a consistência da contagem do número de linhas, é necessário definir se os comentários do código serão desconsiderados.

4.1.1.2 Fácil de Coletar

O processo de coleta de dados não pode tomar muito tempo dos testadores. Devem ser capturadas apenas as informações mais úteis, e, se possível, utilizando ferramentas de automação.

Caso não seja possível automatizar a coleta dos dados, as planilhas podem ser de muita ajuda para registro das medidas. Planilhas podem ser utilizadas tanto para cálculos das estimativas quanto para acompanhar o processo de testes, com o registro dos números obtidos (HUTCHESON).

4.1.1.3 Simples

A simplicidade normalmente é a melhor opção. Inicialmente, devem ser capturadas as métricas mais simples, que também são as mais solicitadas aos analistas de teste, quais sejam, o número de casos de teste a serem executados, o estado atual de cada caso de teste, e o tempo e a data de execução. Uma forma de iniciar o processo de captura de modo apropriado é através da formalização de processos reunindo dados já disponíveis.

As métricas devem ser objetivas e confiáveis, o que inclui clareza, definições que não sejam ambíguas e explicações sobre as mesmas, buscando o melhor entendimento e a facilidade na captura, evitando uma carga muito grande para a equipe de teste. Tomemos como exemplo a métrica que informa o número de casos de teste bloqueados, apresentada na Tabela 4.4, em que a definição e a explicação da métrica são bem elaboradas, com o objetivo de detectar bloqueios indevidos. A descrição da métrica elimina confusões do tipo se os testes foram ou não bloqueados intencionalmente pelos analistas de teste. Além disso, a partir da definição, um caso de teste pode ser considerado como bloqueado ou não; nunca ambos. O analista de teste não poderá aplicar outros critérios subjetivos para marcar um caso de teste como bloqueado. A informação deve ser repassada a todos os envolvidos com o uso de métricas de teste.

Tabela 4.4: Descrição da métrica referente ao número de casos de teste bloqueados

Métrica	Definição	Explicação
Número de casos de teste bloqueados	É o número de casos de teste distintos que não puderam ser executados durante o período de testes em função de problemas na aplicação ou no ambiente.	Define o impacto de alguns <i>issues</i> conhecidos na habilidade da equipe de teste para execução dos casos de teste remanescentes. Um teste bloqueado é aquele que não pode ser executado em função de um problema no ambiente. Um caso de teste também é bloqueado quando é conhecido que um determinado defeito causa falha de sistema. Em função dos possíveis atrasos envolvidos, é útil saber quantos testes não poderão ser executados até que as correções no programa sejam recebidas e verificadas.

Fonte: BRADSHAW.

4.1.1.4 Significativa

A informação coletada deve apresentar um propósito significativo e ser relevante para quem a está capturando. Por exemplo, buscar o número de defeitos e o tempo utilizado em cada fase de teste com o objetivo de determinar a forma mais efetiva de minimizar os defeitos com o menor custo, esclarecendo que as informações serão úteis para destacar os métodos de teste que produzem bons resultados, de maneira a focar em atividades produtivas.

As métricas de teste devem prover feedbacks objetivos para as equipes. A métrica não deve ser coletada se não for significativa, pois a equipe gasta tempo desnecessariamente, sem melhorias para o processo de desenvolvimento.

4.1.1.5 Não Ameaçadora

As métricas de teste não devem ser usadas com o propósito de avaliar os empregados, pois se os mesmos se sentirem ameaçados poderão reportar os dados de forma incompleta e imprecisa.

4.1.2 Obtenção das Métricas de Teste

O primeiro passo para garantir que as informações obtidas sejam acuradas e consistentes é fornecer definições claras e compreensíveis sobre as mesmas.

O momento de captura das métricas de teste depende das métricas identificadas. Por exemplo, se as métricas são resultado de inspeções no projeto, então a captura deve ser feita no estágio de projeto estrutural. Caso as métricas sejam baseadas na análise de defeitos resultantes do teste de sistema, então devem ser capturadas durante o desenvolvimento.

Formas efetivas para a obtenção de métricas de teste incluem o treinamento da equipe, o uso de mecanismos de coleta confiáveis, a apresentação e a obtenção de feedbacks.

4.1.2.1 Treinamento da Equipe

Para que os dados coletados pela equipe sejam adequados, é necessário garantir que as definições das métricas de teste foram devidamente compreendidas, assim como a necessidade do uso das mesmas. Uma forma de facilitar essa compreensão é através de um breve treinamento da equipe, explicando sobre os dados a serem capturados e seus respectivos usos, revisando as definições das métricas e requisitos, citando exemplos de usos das métricas, e aplicando exercícios para garantir o entendimento.

4.1.2.2 Uso de Mecanismos de Coleta Confiáveis

O ideal é que sejam utilizadas ferramentas de automação para minimizar o esforço na captura das métricas, porém isso nem sempre é possível. Sendo assim, para que a coleta dos dados efetuada pela equipe de testes seja confiável, é fundamental garantir que os procedimentos adotados pela equipe sejam simples e estejam devidamente compreendidos. O uso de formulários é bastante eficiente para auxiliar na coleta de dados manual.

A melhoria contínua na metodologia de captura dos dados auxilia na redução do esforço para a coleta e na eliminação de dados inexatos. A origem dos dados imprecisos deve ser identificada para cada métrica, e ações corretivas devem ser tomadas para eliminar tais ocorrências (PUSALA).

4.1.2.3 Apresentação e Obtenção de Feedback

É aconselhável que seja feita uma revisão por amostragem das métricas de teste reportadas. A amostra deve ser elaborada considerando a quantidade de métricas e de testadores.

Após a revisão das amostras, deve haver uma reunião com a pessoa responsável pela coleta das métricas para esclarecer eventuais dúvidas, assim como apresentar e obter feedback.

4.1.3 Análise das Métricas de Teste

A análise das métricas de teste é formada por diversas etapas, incluindo a classificação dos dados em categorias, os cálculos e a análise para a definição das tendências. Caso o objetivo da análise seja o aperfeiçoamento do projeto atual, o processo pode ser feito continuamente, ou se o objetivo for melhorar projetos futuros, a análise pode ser deixada para o final do período de captura das métricas, utilizando a perspectiva histórica obtida no final do projeto.

A análise contínua pode ser comparativa ou cumulativa. Para ilustrar a análise comparativa, temos a monitoração da situação de uma determinada aplicação, que é feita através do cálculo e da comparação do número de defeitos diários, para identificar alguma tendência. A análise cumulativa teria serventia, por exemplo, na determinação dos defeitos mais frequentes e dispendiosos, onde tais informações acumuladas seriam utilizadas na identificação das áreas problemáticas.

4.1.3.1 Classificação dos Dados em Categorias

Os dados capturados devem ser classificados em categorias, de acordo com os objetivos definidos na avaliação das métricas de teste. Por exemplo, se o objetivo for determinar as origens mais frequentes dos defeitos, os dados deverão ser classificados por origem do defeito.

4.1.3.2 Cálculos

Devem ser utilizados percentuais e médias para comparar os dados obtidos. Tomemos como exemplo o percentual de cada categoria de defeito por origem, onde são informados as origens dos defeitos e o número total de defeitos. O cálculo consiste na divisão de cada tipo de defeito por origem e o número total de defeitos, para determinar a categoria com o maior percentual de ocorrências.

4.1.3.3 Análise para Definição de Tendências

As métricas de teste indicam tendências. Por exemplo, a tendência que a maior parte dos defeitos se refere ao código, mas o maior custo para corrigir é dos defeitos de especificação funcional. O custo é maior nesse caso porque devem ser revistos os processos de análise e projetos. Quanto aos erros de código, podem ser necessárias outras informações, como a identificação dos programas onde foram localizados e a complexidade dos mesmos.

4.1.4 Aplicação das Métricas de Teste

Aplicar as métricas de teste significa transformar os resultados da análise das métricas em ações de melhoria do projeto, sempre priorizando ações preventivas do que as corretivas. A aplicação pode ser feita utilizando um sumário das métricas de teste coletadas, fazendo e justificando recomendações de mudanças para melhoria do processo, comunicando os resultados e definindo as ações a serem tomadas.

4.1.4.1 Sumário das Métricas de Teste Coletadas

Deve ser elaborado um sumário das métricas coletadas, dos cálculos efetuados, das tendências observadas, dos planos de pesquisa, das descobertas da pesquisa e dos problemas e causas identificados.

A Tabela 4.5 apresenta um exemplo de sumário das métricas de teste, em que são documentadas as informações obtidas a partir das métricas de teste coletadas.

Tabela 4.5: Exemplo de sumário das métricas de teste

Sumário das Métricas		
Sumário das informações coletadas	Origem do defeito	Verificar o número total de defeitos para origem de defeito (Tabela 4.6)

	Defeitos de código	Verificar o número total de defeitos de código para cada tipo (Tabela 4.8)
Cálculos	Percentual de origem de defeitos	Verificar o gráfico de origem de defeito (Figura 4.5)
	Percentual de defeitos de código	Verificar o gráfico dos defeitos de código por tipo (Figura 4.7)
Tendência	As origens de defeito mais frequentes são de código (a maior parte dos defeitos), seguidas da especificação funcional. Os tipos de defeitos de código mais frequentes são de tratamento dos erros (a maior parte), seguidos de manipulação de dados e interface entre unidades de programas, que têm o mesmo número de ocorrências.	
Plano de pesquisa	Defeitos de código – determinar quais programas que apresentam erros. Comparar os detalhes dos documentos do projeto e o código. As circunstâncias são as mesmas para os defeitos de tratamento dos erros? (Mesmas questões para a manipulação de dados e interface entre unidades de programas). Conversar com os programadores para determinar possíveis causas dos defeitos.	
	Defeitos de especificação funcional – revisar os defeitos para determinar o que estava incorreto.	
Descobertas da pesquisa	Defeitos de código – um programa apresentou dois defeitos, um lógico e outro de interface entre unidades de programas, e o outro programa teve dois problemas de tratamento dos erros. O restante dos defeitos estava em programas diferentes. Para todos os defeitos de tratamento dos erros, faltavam checagens dos erros. O documento detalhado de projeto especificava “incluir checagem dos erros para todas as condições de erro prováveis”. Foi decidido que a definição dos erros seria feita pelos programadores, baseado na forma em que o programa seria escrito. Quatro das ocorrências de manipulação de dados eram baseadas no uso de <i>hard coding</i> quando uma tabela poderia ser usada. Um dos erros era falha de inicialização de uma variável. O erro de interface entre unidades de programas resultou de duas transferências de dados incorretas e três faltantes. A decisão também foi de fazer com que os programadores decidam quais as informações que precisam ser passadas para outro programa. Os programadores informaram que não haviam recebido nenhum feedback referente à qualidade dos programas.	
	Defeitos de especificação funcional – cinco dos oito defeitos de especificação funcional foram causados por falta de clareza nas informações. Para dois, dos cinco defeitos, a	

	<p>informação estava incluída na descrição da narrativa, mas não constava no <i>layout</i> da tela. Para três, dos cinco defeitos, a informação foi mal interpretada pelo designer. Os outros três defeitos eram de processos esquecidos que nunca haviam sido especificados. Os três defeitos foram relacionados como solicitações de mudança, causando mudanças nos percentuais.</p>
Problemas	<p><u>Problema 1</u> – Alto volume de defeitos de tratamento dos erros e defeitos de interface entre unidades de programas.</p> <p><u>Causa 1</u> – Falta de definição de tratamento dos erros e interface entre unidades de programas durante o projeto. Os problemas não são sempre checados antes de liberar para o teste de sistema.</p> <p><u>Problema 2</u> – Existência de confusão entre o uso de tabelas e <i>hard coding</i>.</p> <p><u>Causa 2</u> – O procedimento é explicado nos padrões de desenvolvimento. A aderência aos padrões necessita de reforço e conferência. Os programas não são sempre checados antes de liberar para o teste de sistema.</p> <p><u>Problema 3</u> – Algumas das especificações funcionais estão sendo mal interpretadas.</p> <p><u>Causa 3</u> – As especificações funcionais que foram mal interpretadas foram escritas em um nível muito genérico. Futuramente, as especificações funcionais terão que ser revistas para garantir a existência de especificidade suficiente para prevenir múltiplas interpretações. Existe um checklist de revisão de procedimentos padrão, porém esse item não está incluído.</p> <p><u>Problema 4</u> – Há alguma confusão na definição de um defeito e uma solicitação de mudança.</p> <p><u>Causa 4</u> – O pessoal que documenta os erros não tem conhecimento da definição de solicitação de mudança.</p> <p><u>Problema 5</u> – Inconsistência na especificação funcional, entre a descrição da narrativa e o <i>layout</i> da tela.</p> <p><u>Causa 5</u> – Descuido na parte de revisão das especificações funcionais. O documento foi revisado e aprovado sem identificar tais inconsistências. Existe um checklist padrão para revisão dos procedimentos, porém esse item não foi incluído.</p>
Recomendações	<p>Defeitos de código – A especificação de projeto detalhada deve incluir listas dos prováveis erros e dos tipos de dados necessários para outros programas e provenientes de outros programas. Enfatizar que as tabelas devem ser usadas mais do que <i>hard coding</i>. Após os testes unitários, os líderes da equipe de desenvolvimento devem fazer revisões no programa e dar</p>

	feedbacks aos programadores.
	Defeitos de especificação funcional – Em projetos futuros, devem ser feitas revisões nas especificações funcionais para garantir que as mesmas sejam consistentes e não muito genéricas. Esses dois itens devem ser acrescentados no checklist padrão para revisão dos procedimentos. A diferença entre defeito e solicitação de mudança deve ser esclarecida aos responsáveis pela documentação dos erros.
Justificativa	Nenhuma justificativa adicional é dada para enfatizar o uso de tabelas e a diferença entre um defeito e uma solicitação de mudança, porque são ações simples que não precisam de tempo adicional. Entretanto, a adição de listas dos prováveis erros e dos tipos de dados necessários para outros programas e provenientes de outros programas precisa ser incluída na especificação detalhada de projeto. Esta recomendação pode ser incorporada aos projetos futuros com pouco esforço adicional. A recomendação de adicionar uma checagem para a especificidade requerida ao desenvolvimento e design acurados, e consistência entre o relato e os <i>layouts</i> de telas e relatórios, pode ser incorporada no checklist padrão de revisão de procedimentos com pouco esforço. A recomendação que os líderes de equipe façam revisões e providenciem feedbacks aos programadores é embasada no documento específico para as justificativas das recomendações.

Fonte: BORYSOWICH.

Tabela 4.6: Número total de defeitos para cada origem de defeito

Número Total de Defeitos Para Cada Origem de Defeito					
Especificação funcional	Projeto	Arquitetura	Interface com usuários	Projeto de banco de dados	Código
8	3	0	6	2	25

Fonte: BORYSOWICH.

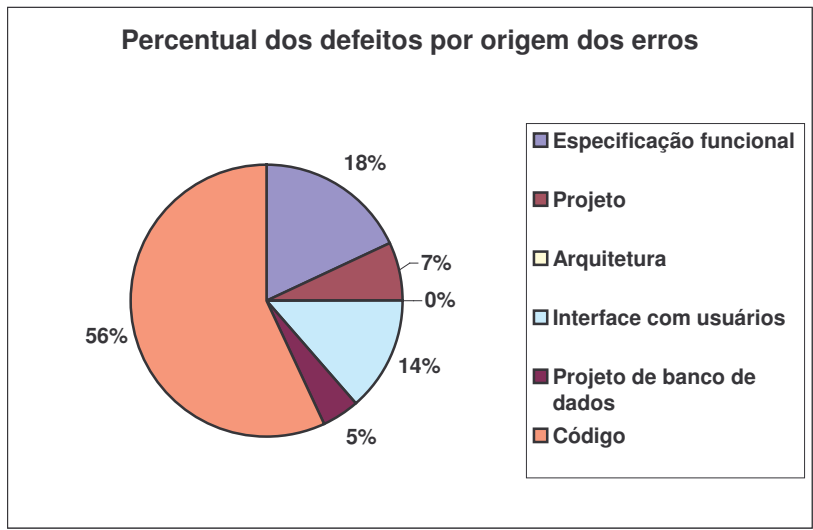


Figura 4.5: Percentual dos erros por origem dos defeitos (SOARES & MARTINHO).

Tabela 4.7: Número total revisado de defeitos para cada origem de defeito

Número Total Revisado de Defeitos Para Cada Origem de Defeito					
Especificação funcional	Projeto	Arquitetura	Interface com usuários	Projeto de banco de dados	Código
5	3	0	6	2	25

Fonte: BORYSOWICH.

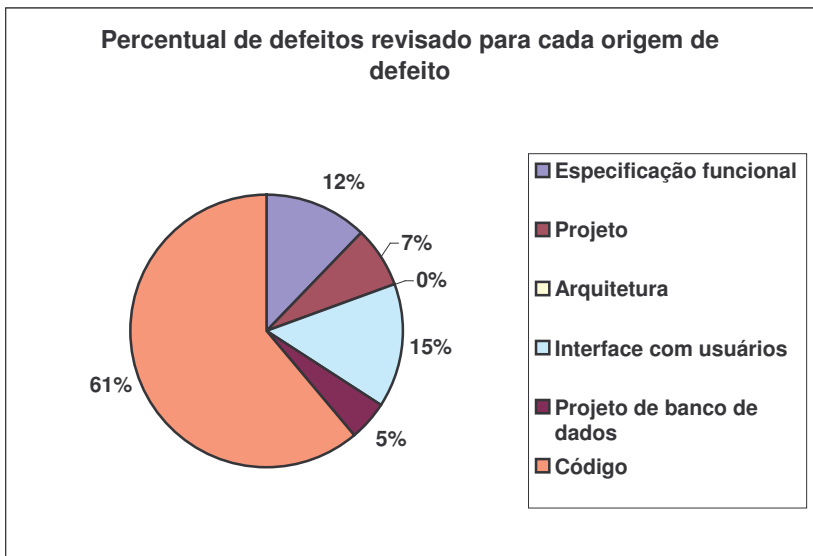


Figura 4.6: Percentual dos erros revisados por origem dos defeitos (SOARES & MARTINHO).

Tabela 4.8: Número total de defeitos de código para cada tipo

Número Total de Defeitos de Código Para Cada Tipo				
Lógica	Cálculo	Interface entre unidades de programas	Manipulação de dados	Tratamento dos erros
3	2	5	5	10

Fonte: BORYSOWICH.

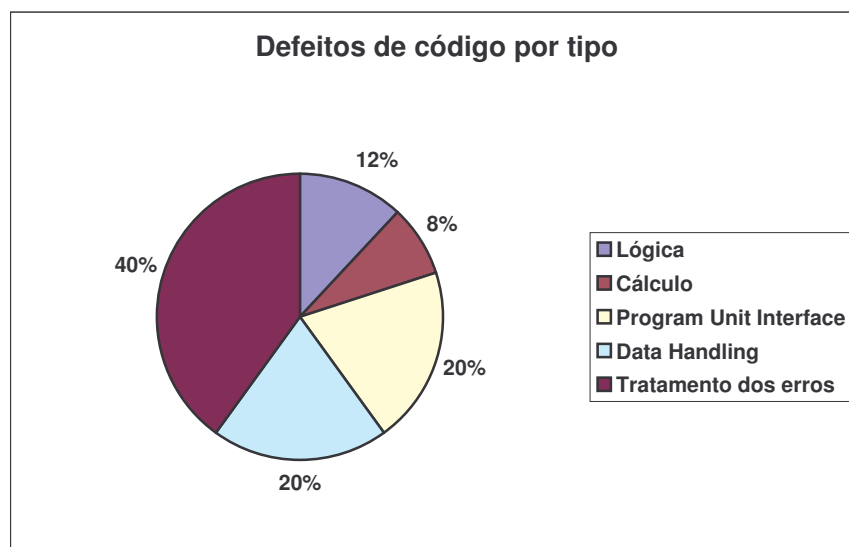


Figura 4.7: Percentual dos defeitos de código por tipo (SOARES & MARTINHO).

4.1.4.2 Recomendação e Justificativa de Mudanças para Melhoria do Processo

No sumário das métricas de teste devem ser incluídas recomendações para correção e prevenção dos problemas identificados. As recomendações devem ser justificadas, seguindo os passos apresentados na Tabela 4.9.

Tabela 4.9: Exemplo de justificativa das recomendações de mudanças para melhoria do processo

Justificativa das Recomendações		
Recomendação	Os líderes da equipe devem fazer revisões dos programas e prover <i>feedbacks</i> aos programadores antes de liberar os programas para os testes de sistema.	
Estabelecer regras	Densidade média dos erros antes de liberar para o teste de sistema (sem revisão de programa anteriormente)	7 em 1.000 linhas de código não comentadas (ULOC)
	Média de erros encontrados durante a	50%

	revisão do programa	
	Tempo médio necessário para conduzir uma revisão do programa	400 ULOC por hora
	Tempo médio necessário para confirmar, documentar, e reportar defeitos durante a revisão de programa *	0,5 hora
	Tempo médio necessário para confirmar, documentar, e reportar defeitos durante o teste de sistema *	1 hora
	Tempo médio necessário para corrigir durante a revisão do programa **	3 horas
	Tempo médio necessário para corrigir durante o teste de sistema **	7 horas
Suposições	Quantidade aproximada de linhas de código não comentado no sistema	40.000 ULOC
	Revisores do programa	Dois líderes da equipe
	Valor médio do recurso – líder da equipe	\$60 por hora
	Valor médio do recurso – testador	\$50 por hora
	Valor médio do recurso – programador	\$30 por hora
Cálculos	Tempo para executar a revisão de programa em todo o código	40.000 ULOC / 400 ULOC por hora = 100 horas
	Custo para conduzir a revisão de programa em todo o código	100 horas x \$60 por hora = \$6,000
	Número de defeitos antecipados	7 x 40 (40.000/1.000) = 280
	Número de defeitos a serem encontrados durante a revisão de programa pelo líder da equipe	0,5 x 280 = 140 defeitos
	Número de horas para confirmar, documentar e reportar os defeitos durante a revisão de programa	0,5 hora x 140 = 70 horas
	Número de horas para confirmar, documentar e reportar os defeitos durante o teste de sistema	1 hora x 140 = 140 horas
	Número de horas para corrigir os defeitos durante a revisão do programa	3 horas x 140 = 420 horas
	Número de horas para corrigir os defeitos durante o teste de sistema	7 horas x 140 = 980 horas
	Custo para confirmar, documentar, reportar e corrigir os defeitos durante a revisão do programa	(70 x \$60) + (420 x \$30) = \$16,800

	Custo para confirmar, documentar, reportar e corrigir os defeitos durante o teste de sistema	$(140 \times \$50) + (980 \times \$30) = \$36,400$
	Custo total para adicionar o esforço de revisão de programa	$\$6,000$ (tempo do líder da equipe para a revisão) + $\$16,800$ (custo para confirmar, documentar, reportar e corrigir os defeitos) = $\$22,800$
	Lucro obtido ao encontrar os defeitos através da revisão do problema em relação ao teste de sistema	$\$36,400 - \$22,800 = \$13,600$

Fonte: BORYSOWICH.

Na Tabela 4.9, podemos observar que a média de tempo para confirmar e documentar os defeitos é menor na revisão do programa, pois o processo de documentação é mais informal, e o revisor tem a vantagem de poder consultar o código para confirmar o defeito, ao invés de apenas repetir ações do sistema. Além disso, a revisão também previne defeitos, pois o feedback é dado aos programadores durante o desenvolvimento, possibilitando que os erros não sejam repetidos. A média de tempo para corrigir os defeitos também é menor na revisão de programa, pois o revisor consegue identificar exatamente onde o erro foi originado, economizando o tempo do programador na localização da causa do erro. A explicação da correção do defeito também é mais informal durante o processo de revisão do programa.

Tabela 4.10: Número de horas para confirmar, documentar, reportar e corrigir defeitos

Número de Horas Para Confirmar, Documentar, Reportar e Corrigir Defeitos			
Tipo de teste	Confirmado, documentado e reportado	Corrigido	Total
Revisão de programa	70	420	490
Teste de sistema	140	980	1.120

Fonte: BORYSOWICH.

Tabela 4.11: Custo para confirmar, documentar, reportar e corrigir defeitos

Custo Para Confirmar, Documentar, Reportar e Corrigir Defeitos			
Tipo de teste	Confirmado, documentado e reportado	Corrigido	Total
Revisão de programa	\$4,200	\$12,600	\$16,800
Teste de sistema	\$7,000	\$29,400	\$36,400

Fonte: BORYSOWICH.

4.1.4.4 Comunicação dos Resultados

Após a conclusão do sumário das métricas de teste e eventuais justificativas, as informações devem ser repassadas à gerência para aprovação.

4.1.4.4 Ações a Serem Tomadas

Caso os resultados sejam aprovados pela gerência, as ações devem ser tomadas. Dependendo do tipo de ação, é aconselhável que a implementação seja feita em uma pequena escala, para validação, e, posteriormente, estender a toda a organização. O sucesso obtido com o uso das métricas de teste deve ser comunicado à equipe.

A melhoria do processo deve ser contínua. Os objetivos devem ser constantemente revistos, *feedbacks* devem ser solicitados regularmente aos *stakeholders*, as métricas podem ser alteradas em função das necessidades, novas métricas podem ser criadas, os *templates* devem ser refinados, o esforço para captura e relato das métricas deve ser mínimo. É necessário acompanhamento constante (PUSALA).

4.2 Como Relatar as Métricas de Teste de Software

As informações obtidas a partir do esforço de teste são utilizadas pelos *stakeholders* para acompanhar o andamento do projeto, assim como pelos executivos que precisam decidir sobre a liberação do produto para o mercado, e, talvez, essa seja uma das razões para as organizações investirem em equipes independentes de teste (MAGAZINE).

É certo que fornecer informações é um dos papéis importantes dos testes, porém é fundamental a forma como tais dados são apresentados, pois isso é definitivo na tomada de decisões críticas com tranquilidade ou não. Tomemos um exemplo em de um projeto que está há dois dias da liberação do produto ao cliente, e o relatório de status dos testes informa apenas sobre os defeitos em aberto e as tendências dos defeitos, sem informações importantes como cobertura dos testes. Relatórios com informações incompletas podem levar a tomadas de decisões sem o conhecimento total dos fatos (MAGAZINE).

Durante as reuniões de revisão, a equipe responsável deve examinar todas as métricas disponíveis, e usar essas informações para identificar a origem dos problemas existentes, pois as métricas dão uma visão do que acontece durante o esforço de teste. Se as métricas forem capturadas durante projetos distintos, poderão ser feitas

comparações entre o projeto corrente e o histórico de projetos anteriores, possibilitando identificação de eventuais tendências no projeto, principalmente quando forem implementadas mudanças de projeto. É interessante também observar se as métricas são típicas de projetos da organização, quando comparadas com o padrão do mercado (BRADSHAW).

A seguir relacionamos alguns pontos importantes que devem ser observados pelos responsáveis pela geração dos relatórios de status dos testes, baseados nas métricas de teste (MAGAZINE).

4.2.1 Público que Fará Uso do Relatório de Status dos Defeitos

É necessário identificar as pessoas que farão uso do relatório contendo os status dos defeitos. Os relatórios não devem ser encaminhados para listas de distribuição de forma indiscriminada, pois muitos dos destinatários ignorarão o relatório recebido, podendo ser considerado como simples *spam*. Em empresas muito grandes, a distribuição correta dos relatórios pode ser uma tarefa extremamente difícil, porém, no mínimo deve haver um conhecimento sobre os papéis das pessoas dentro do projeto, para antecipar as expectativas que as pessoas terão a respeito do relatório recebido.

Uma boa alternativa é listar os papéis das pessoas que podem receber o relatório, tais como, gerente de produto, gerente de projeto, gerente de desenvolvimento, diretor de desenvolvimento, equipe de teste, equipe de desenvolvimento, vice-presidência, entre outros. Esse é o primeiro passo na criação de uma lista de distribuição adequada.

4.2.2 Antecipação de Questões Úteis aos Destinatários

É fundamental analisar as questões que são significativas para as pessoas que vão ler o relatório. O objetivo principal dos relatórios de status do projeto é responder às mais diversas perguntas relacionadas ao status feitas pelos *stakeholders* do projeto, que variam de acordo com a fase do projeto. O relatório precisa responder às diversas questões com clareza e objetividade. A elaboração de um relatório com informações apropriadas auxilia na diminuição do tempo gasto em reuniões.

Relacionamos a seguir exemplos de questões que devem ser esclarecidas em um relatório de status do projeto para a fase de execução dos testes. A lista foca em alguns status que são importantes para serem analisados na fase de execução dos testes.

- Gerente de desenvolvimento
 - Qual é a taxa de defeitos? Ela está diminuindo ou aumentando?
 - Quais os componentes que apresentam os defeitos de maior gravidade?
 - Quais os componentes que possuem defeitos que estão bloqueando os testes?
- Gerente de projeto
 - Os testes estão ocorrendo dentro do prazo previsto?
 - Há algum defeito que esteja bloqueando os testes de forma significativa?

- Se existem riscos, quais os planos para mitigá-los?
- Gerente de produto
 - Existe algum defeito que precise da atenção da gerência de produto?
 - Quais os defeitos que estão sendo adiados?
- Gerente de teste
 - Qual o percentual da cobertura de teste que está concluída?
 - Qual o percentual de testes encontrados baseados em uma determinada metodologia de testes?
 - Qual a razão para que os defeitos estejam aumentando?
 - Qual é a taxa de correção dos defeitos?

4.2.3 Uso de Objetividade nas Respostas às Questões

As questões apresentadas no relatório de status dos testes devem ser esclarecidas de forma objetiva.

O primeiro passo é definir a lista de questões a serem apresentadas no relatório. Na sequência, devem ser projetadas as métricas que permitirão responder tais questões. Cada questão deve ser avaliada a partir dos dados fornecidos nas métricas e respondida da forma mais objetiva possível, baseada em fatos, evitando adicionar opiniões pessoais ao relatório.

4.2.4 Distribuição das Informações no Relatório

A posição das informações no relatório é vital, sendo que as mais importantes devem ser incluídas no início do relatório.

Deve-se ter o cuidado de colocar as informações que envolvem riscos no início do relatório, com os itens positivos na sequência. Tal distribuição dos dados evita o esquecimento de informações importantes. Relatórios que são considerados de boa qualidade devem alertar os destinatários sobre alguma informação essencial à primeira vista.

Uma boa sugestão é fazer um sumário dos status, antes de entrar em detalhes sobre os mesmos.

4.2.5 Tratamento dos Itens Relacionados aos Riscos

As pessoas responsáveis pela geração dos relatórios devem se sentir encorajadas para poderem mencionar itens relacionados aos riscos, evitando qualquer espécie de temor em apresentar más notícias. Se o processo não está adequado, a equipe responsável deve ter conhecimento disso, de forma a saberem corretamente dos riscos, ou seja, da situação geral.

Caso necessário, deve ser usada uma linguagem eficaz para passar a mensagem necessária, não descuidando nunca do profissionalismo.

4.3 Problemas Verificados na Utilização de Métricas de Teste de Software

As métricas de teste de software são bastante úteis ao processo de teste, porém é importante observar algumas dificuldades relacionadas ao assunto. As questões tratadas variam de problemas gerenciais e de relacionamento a observações específicas sobre como o uso isolado das métricas pode fornecer informações equivocadas.

Tomemos por exemplo uma situação em que a equipe conclui os testes, são geradas métricas e o líder de teste publica um relatório certificando a qualidade da aplicação. Entretanto, a maior parte do que foi publicado estava errado, e o sistema apresentou falhas em produção, gerando bastante constrangimento da equipe de teste com os clientes e com os gerentes. O problema, nesse caso, foi que as métricas de teste avaliadas sugeriam uma boa qualidade da aplicação, pois a relação entre as ocorrências e os defeitos era de 1:1, o índice de severidade dos defeitos diminuía a cada semana e o tempo para encontrar os defeitos estava aumentando. Situações como essa não são raras e podem existir múltiplas razões relacionadas aos testes que explicam o ocorrido, como má qualidade dos testes, práticas de teste pobres ou processo de teste ineficiente. Porém, é importante observar o papel do líder ou do gerente de testes e sua habilidade em analisar as diferentes métricas para compreender o que elas significam, ou até mesmo o que está subentendido nas métricas (MAGAZINE).

Muitas vezes, por falta de entendimento, os líderes e os gerentes de teste confiam em algumas métricas para medir a qualidade da aplicação que está sendo testada. Não que isso seja um problema, porém é necessário tomar alguns cuidados antes de aceitar totalmente o que as tendências sugerem. Relacionamos a seguir algumas propostas de cuidados a serem adotados ao analisar a qualidade de uma aplicação (MAGAZINE).

- As métricas de teste por si só não fornecem a percepção adequada da real qualidade da aplicação.
- As métricas de teste não devem ser analisadas isoladamente. Diferentes métricas devem ser comparadas e estudadas para fornecer um parecer confiável dos testes.
- O estudo da origem dos problemas como parte da análise das métricas provê resultados mais confiáveis.
- Uma análise sistemática e completa das métricas de teste é importante para que sejam consideradas como ferramentas confiáveis para medir a qualidade da aplicação.
- As tendências de algumas métricas de teste podem ser analisadas por diversos ciclos de teste, enquanto outras são relativas a um ciclo de teste específico.

Apresentaremos outros exemplos para ilustrar alguns problemas de interpretação das métricas de teste (MAGAZINE). No primeiro caso, temos uma organização cujo programa de métricas de teste inclui a coleta das seguintes métricas: relação entre defeitos e ocorrências, índice de severidade dos defeitos, tempo médio para encontrar

um defeito e cobertura dos testes. Os gráficos das Figuras 4.8, 4.9, 4.10 e 4.11 foram gerados pela equipe de testes a partir dos testes de um determinado produto.

Ao analisarmos o gráfico da Figura 4.8, podemos deduzir que há uma tendência favorável, pois nove dos dez ciclos de teste apresentaram taxa crescente, significando que a maior parte dos registros foram convertidos em defeitos, ou seja, o número de ocorrências inválidas, duplicadas, foi diminuindo a cada ciclo de teste. A questão é que a análise não pode ficar restrita à tendência favorável apresentada pelo gráfico, sendo necessário observar alguns fatores que podem mostrar uma situação diferente. A cobertura dos testes deve ser considerada, pois se ela estiver baixa, confiar apenas na relação entre os defeitos e as ocorrências pode resultar em uma análise pobre. Por exemplo, se estivermos utilizando uma cobertura de requisitos e apenas 70% deles estiverem cobertos pelos testes, é preciso observar se os 30% não cobertos não dizem respeito às funcionalidades mais importantes da aplicação. A severidade dos defeitos também traz informações importantes, pois as ocorrências podem tratar apenas de defeitos cosméticos, e, nesse caso a relação favorável do gráfico não indica a qualidade da aplicação. O gráfico não traz informações sobre o número de defeitos. Sendo assim, se a aplicação tiver 1.000 defeitos e a equipe de teste encontrou apenas 100, também teremos problemas se nos limitarmos a analisar o gráfico da Figura 4.8, já exposto anteriormente. Por último, temos a classificação dos defeitos, que também não está contemplada no gráfico. Digamos que 90% dos defeitos detectados sejam técnicos. É possível que a equipe de teste tenha feito um bom trabalho de detecção de defeitos técnicos, em detrimento dos defeitos de funcionalidade, por exemplo.

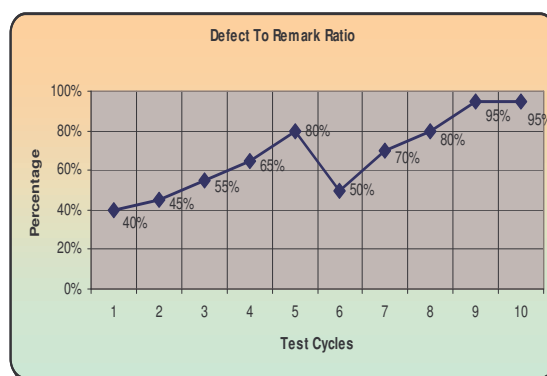


Figura 4.8: Relação entre as ocorrências e os defeitos encontrados (MAGAZINE)

O gráfico da Figura 4.9 também indica uma tendência favorável para a qualidade do produto, pois o índice de severidade dos defeitos está diminuindo de forma significativa a cada ciclo de teste, ou seja, menos defeitos críticos e de alta severidade estão sendo reportados. O problema está na análise isolada do índice de severidade dos defeitos. Para uma análise mais significativa, um dos fatores a serem considerados também é o número de defeitos encontrados em cada ciclo, conforme Tabela 4.12. Verificando o índice de severidade do ciclo 2 em relação ao ciclo 1, temos a impressão de que o índice do ciclo 2 é favorável, porém se analisarmos detalhadamente a tabela, a situação é diferente. No ciclo 1, temos 15 defeitos nas severidades 1 e 2, enquanto no ciclo 2 são 20 defeitos. Isso quer dizer que a qualidade do ciclo 2 é pior do que a do ciclo 1, apesar de o índice de severidade no ciclo 2 ser melhor. A cobertura dos testes tem um impacto

semelhante, pois uma cobertura de testes mais baixa com índice de severidade decrescente pode indicar que a tendência não é positiva.

Tabela 4.12: Número de defeitos encontrados

Severidade dos defeitos	Quantidade de defeitos no ciclo 1	Quantidade de defeitos no ciclo 2
S1	5	5
S2	10	15
S3	50	30
S4	100	100
Índice severidade	1,52	1,43

FONTE: (MAGAZINE)

Um outro exemplo de problema em considerar apenas o índice de severidade dos defeitos é apresentado na Tabela 4.13, em uma situação com dois ciclos de teste, cujos índices de severidade são 2,42 no primeiro ciclo e 2,92 no segundo. Considerando este índice, temos a idéia de que o ciclo 1, com índice de severidade menor, sinaliza melhor tendência do que o ciclo 2, porém, o ciclo 2 é bem melhor do que o ciclo 1 no total de defeitos com severidade 1 e 2.

Tabela 4.13: Severidade dos defeitos

Severidade dos defeitos	Quantidade de defeitos no Ciclo 1	Quantidade de defeitos no Ciclo 2
S1	4	0
S2	4	0
S3	42	75
S4	27	2
Índice de severidade	2,42	2,92

FONTE: (MAGAZINE)

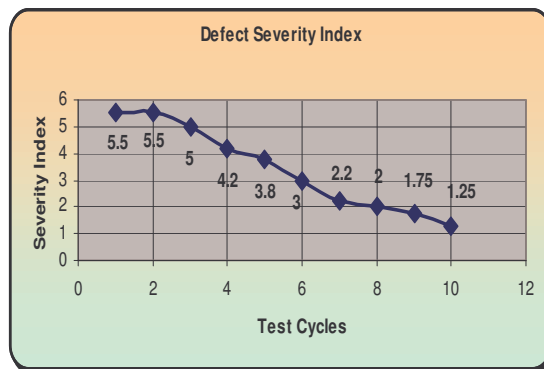


Figura 4.9: Índice de severidade dos defeitos (MAGAZINE)

O gráfico da Figura 4.10 indica que o tempo entre cada defeito encontrado está aumentando a cada ciclo de testes, ou seja, que a equipe de testes está demorando cada vez mais para encontrar defeitos, e, por consequência, a qualidade da aplicação está melhor do que nos ciclos iniciais. Podemos verificar que no primeiro ciclo os testadores demoravam 5 minutos para encontrar um defeito, enquanto no décimo ciclo esse tempo aumentou para 55 minutos, o que indica uma boa tendência. Porém, é necessário considerar a severidade dos defeitos encontrados antes de efetuar conclusões sobre a qualidade da aplicação. Podemos ter uma situação em que os testadores detectam apenas defeitos de severidade baixa no primeiro ciclo e defeitos de severidade crítica no décimo ciclo, o que altera a idéia da qualidade da aplicação, pelo fato de encontrar defeitos críticos no ciclo final.

Outra questão importante a ser analisada no gráfico da Figura 4.10 já mostrada no item 2.3.1.4 diz respeito à necessidade de avaliar o tipo dos defeitos em conjunto com o tempo entre os defeitos encontrados. Por exemplo, se os testes efetuados durante o ciclo 1 foram de interface e no ciclo 10 os testes foram de transações do banco de dados, consequentemente os defeitos no ciclo 1 foram identificados em um menor espaço de tempo do que no ciclo 10. O padrão é de que uma simples transação do banco de dados é mais importante do que 10 defeitos na interface, detectados no mesmo espaço de tempo. Outra possibilidade seria que a cobertura do ciclo 1 fosse de 90% dos requisitos, enquanto o ciclo 10 cobrisse apenas 10% dos requisitos. Uma maior cobertura com certeza levaria a um maior número de defeitos detectados.

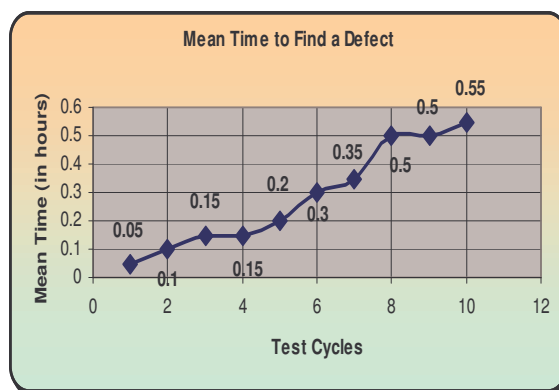


Figura 4.10: Tempo em horas entre os defeitos encontrados (MAGAZINE)

Finalizando, temos o gráfico da Figura 4.11 que trata da métrica relacionada à cobertura dos testes e indica que a cobertura dos requisitos está adequada aos 70%, a cobertura de código está em 50% e necessita de melhorias, 90% dos casos de teste documentados foram executados, e 100% dos casos de teste planejados foram executados. De forma geral, a cobertura dos testes é boa, apenas com necessidade de melhoria na cobertura de código.

Temos diversas questões a serem analisadas na Figura 4.11. A primeira delas é que a informação da cobertura dos requisitos no percentual de 70% não indica se as funcionalidades mais críticas ou as mais simples estão sendo cobertas pelos testes. O gráfico também não menciona os requisitos funcionais ou não funcionais. Se os 30% faltantes estiverem concentrados nas partes mais importantes da aplicação, o índice de 70% é inadequado. Com relação à cobertura de código de 50%, é necessário explicitar melhor a que se refere o percentual. Além disso, a definição de que tal percentual é suficiente depende se a aplicação em teste é crítica ou não. Se a aplicação tiver funcionalidades críticas, o percentual de 50% é insuficiente; porém, se aplicação for simples, desenvolvida para uso interno da organização para um controle básico de recursos humanos (RH), por exemplo, a cobertura pode ser adequada.

Na Figura 4.11 já apresentada, há a informação de que 90% de casos de teste documentados foram executados, o que parece ser um bom percentual. Para considerar completamente tal informação para definir a qualidade da aplicação, é necessário analisar se os casos de teste documentados são de qualidade, assim como, verificar se cobrem todos os requisitos funcionais críticos. Quanto ao percentual de 100% de execução dos casos de teste planejados, também é sugerida a atenção para a robustez dos casos de teste planejados, que depende da habilidade da equipe em planejar bons testes.

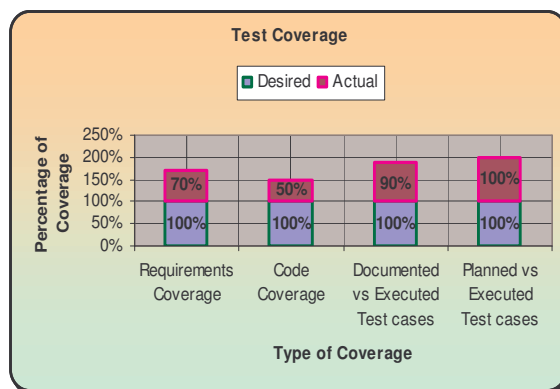


Figura 4.11: Cobertura dos testes (MAGAZINE)

Outra questão a ser observada é que muitas das tarefas mencionadas são realizadas por pessoas. Os softwares são planejados, concluídos, testados, gerenciados e utilizados por pessoas, e, em função disso, o sistema pode ser afetado por questões inerentes ao comportamento humano, que muitas vezes são ignoradas (KANER & BOND). Se levarmos em conta tais variáveis, teremos um desafio a lidar e com muitas dimensões a serem analisadas, apesar do caráter objetivo das medições.

É usual o fato de que empresas estabelecem programas de métricas e pouco acontece em relação aos programas, ou então incluem as métricas apenas para estarem de acordo

com os critérios do *Capability Maturity Model* (CMM). Isso ocorre também para as métricas de teste e pode ser interpretado como falta de maturidade e profissionalismo das empresas ou resistência aos altos custos dos programas de métricas. Em alguns casos, isso é correto; porém, em outros, a resistência ocorre porque as métricas prejudicam mais do que ajudam, quando, por exemplo, um projeto ou empresa são gerenciados baseados nos resultados de métricas inadequadas ou mal interpretadas, com distorções das medidas (KANER & BOND).

Quanto mais as organizações entendem a necessidade do uso de métricas, mais desafios surgem para a implementação das mesmas de forma adequada. Buscando evitar armadilhas comuns no uso de métricas de teste, relacionamos alguns aspectos que devem ser considerados em sua utilização (PUSALA):

- É necessário o comprometimento da gerência para qualquer iniciativa de melhoria nos processos existentes.
- Devem ser selecionadas apenas as métricas de teste mais importantes e que adicionem valor ao processo. Novas métricas podem ser incluídas no decorrer do projeto, assim que a equipe começar a sentir os benefícios do uso de métricas.
- As métricas devem ser coletadas durante todo o processo. Coletar poucas métricas e quando o processo já estiver adiantado não fornece informações corretas para tomar decisões apropriadas.
- As métricas de teste precisam estar relacionadas aos objetivos do processo de testes; caso contrário, não há razão para coletá-las. Métricas que não serão usadas não devem ser coletadas.
- Definições ambíguas das métricas podem ser perigosas, considerando que as pessoas podem interpretá-las de formas diversas, gerando resultados incorretos.
- As métricas não devem ser utilizadas para avaliar individualmente os membros da equipe, pois o medo das pessoas de que as métricas sejam usadas para prejudicá-las é muito grande, sendo essa uma das razões para os programas de métricas serem rejeitados.
- O uso das métricas de teste deve estar focado nas melhorias geradas no processo em função dos dados apresentados, e nunca na motivação da equipe. O uso das métricas para motivação pode dar uma noção errada à equipe de que estão sendo avaliados individualmente através das métricas.
- Uma comunicação adequada é fundamental para o sucesso do uso de métricas de teste. Por isso, é necessário explicar o porquê das métricas, assim como compartilhar os resultados das mesmas. Todas as pessoas envolvidas com os testes precisam entender a relevância dos dados que estão sendo coletados.
- Em algumas situações, pode haver interpretação equivocada das métricas, por erro no momento de distinguir entre possíveis tendências referidas. O ideal é não tomar decisões baseadas em tais variações.

Os exemplos expostos fazem parte de inúmeras questões que devem ser observadas pela equipe de testes no uso das métricas, e auxiliam na compreensão de alguns aspectos importantes das informações dos testes. As métricas de teste são ferramentas que acrescentam bastante valor ao processo, porém devem ser considerados os diferentes componentes e fatores que as afetam, de forma a garantir a confiança nas decisões tomadas a partir delas.

4.4 Proposta de Uso de Métricas de Teste de Software

A proposta apresentada foi elaborada junto a uma das empresas que respondeu à pesquisa, cujo diagnóstico encontra-se no capítulo 3, e informou não utilizar métricas de teste de software.

A empresa trabalha com diferentes projetos para clientes externos, e, de acordo com o que for definido em contrato, pode prestar manutenção dos sistemas. Também trabalha com projetos de uso interno, para os quais são desenvolvidas novas versões. A empresa selecionada forneceu as seguintes informações no questionário:

- Tempo de utilização dos testes na empresa
 - Entre um e três anos.
- Os testes são feitos
 - Por uma equipe da própria empresa.
- Quem faz os testes na empresa?
 - Uma equipe específica de teste, após os desenvolvedores terem feito os testes dos códigos.
- Formação da equipe de testes
 - 1 analista de teste e 2 testadores.
- Os profissionais têm especialização em testes?
 - Apenas cursos introdutórios sobre testes.
- O processo de testes é documentado?
 - Sim.
- Em que momento são efetuados os testes no processo de desenvolvimento do sistema?
 - Em todas as etapas de desenvolvimento do sistema.
- Usam ferramentas de automação dos testes? Quais?
 - Usam o a ferramenta Mantis para a gestão dos defeitos.
- Quais os testes adotados?
 - Nível de teste – Teste de integração (ou teste de interface), teste de sistema, teste de aceitação, teste de regressão.
 - Tipos de teste – Teste de interface, teste de desempenho (ou teste de performance), teste de usabilidade, teste de funcionalidade e teste de instalação.

- A empresa mantém um histórico dos erros?
 - Sim.
- Quais as barreiras para a não utilização das métricas?
 - Consome muito tempo.
 - Não é útil ou o custo/benefício não compensa.
- Estimativas usadas
 - Métricas de projetos anteriores para estimativa do esforço de teste.

Considerando todas as questões estudadas sobre o uso de métricas de teste e o perfil da empresa analisada, foram definidos alguns pontos básicos para a proposta de implementação das métricas. O primeiro ponto diz respeito às barreiras apresentadas para a utilização das métricas. De acordo com a empresa, as métricas consomem muito tempo e não são úteis ou a relação custo/benefício não compensa. Em função disso, a proposta foi formulada com foco nos objetivos prioritários para o uso de métricas de teste e no fato de que as métricas serão geradas com os dados já capturados pela equipe, evitando o aumento da carga de trabalho. Outras questões importantes analisadas junto à empresa são relacionadas à maneira como as métricas de teste serão reportadas e para quem.

O objetivo desta proposta é apresentar à empresa uma ideia inicial sobre as métricas de teste. Sendo assim, foram selecionadas as questões consideradas mais importantes, deixando para um segundo momento assuntos como, por exemplo, a necessidade de estabelecer um prazo para atingir os objetivos. A estratégia adotada pretende mostrar as métricas da forma mais simples possível, com o intuito de atrair o interesse da empresa para o assunto.

No primeiro momento, foram apresentados os diversos objetivos para o uso de métricas, estudados no capítulo 2, para que fossem estabelecidas as prioridades em relação aos mesmos. Os objetivos selecionados como prioritários pela empresa são:

- Objetivo 1: Analisar os defeitos: obter informações relacionadas às origens dos defeitos, à forma como foram detectados, quando foram detectados, etc.
 - Quantos defeitos podemos esperar?
 - Quais os tipos de defeitos encontrados?
 - Quantos defeitos já foram corrigidos?
 - Quais os defeitos prioritários?
 - Qual o testador que encontrou mais defeitos?
 - Quantos defeitos foram localizados por um determinado testador?
 - Quantos defeitos foram encontrados pelo usuário?
- Objetivo 2: Calcular o tempo e os recursos gastos com os testes.
 - Qual o custo dos testes?
 - Qual o custo para corrigir os defeitos?
- Objetivo 3: Visualizar se o produto está pronto para liberação ao cliente.

- Já foi testado o suficiente?
- Objetivo 4: Identificar o momento correto para concluir os testes.
- Objetivo 5: Melhorar a exatidão das estimativas.
- Objetivo 6: Auxiliar no gerenciamento do projeto e da execução dos testes.
- Objetivo 7: Avaliar o andamento do teste, em relação ao cronograma, através do status do teste.
 - Quanto tempo falta para terminar o ciclo de testes?
 - Quanto já foi testado?
 - Os testes serão concluídos dentro do prazo previsto?
 - Quanto teste ainda tem que ser feito em determinada área?
- Objetivo 8: Planejar adequadamente os recursos, prazos e benefícios do processo de testes.
- Objetivo 9: Identificar áreas que necessitam de melhorias.
 - Quais as áreas do software que têm mais ou menos defeitos?
- Objetivo 10: Identificar se o programa de testes é robusto, ou se a suíte de testes é fraca.
- Objetivo 11: Viabilizar a tomada de decisão de forma ágil (avaliação de escolhas, comparação de alternativas e monitoramento de melhorias).
- Objetivo 12: Detectar tendências nos dados que mostrem a necessidade de mais testes em determinadas áreas.

Após a definição dos objetivos prioritários, foram analisados os dados que a empresa já tem disponíveis para a geração das métricas. Vislumbrando a evolução do trabalho, para os dados não coletados atualmente pela empresa, foi questionado o grau de dificuldade para a captura dos mesmos. Os resultados são apresentados na Tabela 4.14, em que são relacionados diversos dados necessários para a geração das métricas, com a respectiva informação da empresa quanto à coleta atualmente efetuada.

Na empresa, as áreas de teste e desenvolvimento são separadas, com diferentes gestores. Em função disso, ao especificar os objetivos, a pessoa responsável mencionou diversas vezes que a utilização das métricas seria útil apenas para controlar os testes, e não o processo como um todo. Contudo, é importante salientar que as métricas de testes podem auxiliar de diversas formas na melhoria do projeto de desenvolvimento do sistema como um todo, e não somente para o processo de testes, como, por exemplo, a métrica relacionada aos tipos de defeitos encontrados no sistema, que permitem identificar áreas do desenvolvimento com maiores problemas, para melhor gerência do trabalho efetuado.

Tabela 4.14: Análise dos dados disponíveis para a geração das métricas de teste

Dado	Já é coletado?	SIM Como é coletado?	NÃO Qual a dificuldade em coletar?
Quantidade de casos de teste	NÃO		Os casos de teste são informais, criados a partir da experiência do analista de testes. Bem difícil. É viável.
Status dos casos de teste (passou, falhou, sob investigação, bloqueado, reexecutado)	NÃO		Os casos de teste são informais. Bem difícil, mas é viável.
Quantidade de ocorrências	SIM	Através do Mantis.	
Status das ocorrências	SIM	Através do Mantis.	
Ocorrências por período	SIM	Através do Mantis.	
Quantidade de defeitos	SIM	Através do Mantis.	
Status dos defeitos (novo, aberto, fechado, corrigido e testado novamente, corrigido e pendente de teste, entre outros)	SIM		
Defeitos por período	SIM		
Defeitos por caso de teste	NÃO		Bem difícil, mas é viável.
Severidade dos defeitos	NÃO		Bem difícil, mas é viável.
Tempo para arrumar um defeito (durante o período de testes e depois que já está em produção)	SIM	Apenas o tempo para arrumar um defeito durante o período de testes consta no Mantis. Quando o produto já foi liberado para o cliente, eventuais defeitos detectados não passam pela equipe de testes, o desenvolvedor é que arruma.	
Tempo do teste	SIM	Através do Mantis.	
Quantidade de defeitos em produção	NÃO		Difícil.
Tipos de defeitos (falhas de sistema, falhas de comunicação,	SIM	Através do Mantis.	

entre outros)			
Cobertura dos testes (todos os requisitos têm caso de teste correspondente?)	NÃO		Muito difícil.
Custo em testar	SIM		
Custo de desenvolvimento	SIM	A empresa tem. Poderiam conseguir para comparar.	
Defeitos corrigidos que resultaram em novas ocorrências	SIM	Através do Mantis.	
Defeitos encontrados por fase de teste	NÃO		Impossível. Não divide por fases. São versões geradas.
Defeito por módulo	NÃO		Poderia ser feito sem problemas.
Defeito por fase em que foi injetado (projeto, codificação, documentação, entre outros)	NÃO		Impossível, no momento.

Considerando as informações relacionadas na Tabela 4.14, foram selecionados os dados atualmente capturados pela empresa, que devem ser utilizados para a geração das métricas no início da implementação do processo. Para cada um dos dados, foram definidas as métricas de teste associadas, conforme Tabela 4.15. A estratégia de trabalhar inicialmente com os dados já capturados durante os testes foi definida com o intuito de minimizar as mudanças no trabalho da equipe no primeiro momento. Após a utilização das métricas, e a verificação na prática de que as mesmas são importantes, dados adicionais poderão ser coletados, a partir da necessidade da empresa e da própria equipe.

Tabela 4.15: Definição das métricas associadas aos dados capturados no processo de testes

Dados capturados atualmente pela empresa	Métricas de teste associadas
Quantidade de ocorrências	Número de ocorrências
Status das ocorrências	Status das ocorrências Taxa de ocorrências válidas
Quantidade de defeitos	Índice de densidade dos defeitos Efetividade e eficiência dos testes
Status dos defeitos (novo, aberto, fechado, corrigido e testado novamente, corrigido e pendente de teste, entre outros)	Situação ou tendência dos defeitos em função do tempo Providências adotadas em relação aos defeitos <i>Zero Bug Bounce</i> Densidade dos defeitos residuais

	Defeitos encontrados x Defeitos corrigidos Ocorrências pendentes de correção Ocorrências resolvidas que ainda não foram retestadas
Defeitos por período	Tempo médio para encontrar um defeito
Tempo para arrumar um defeito (durante o período de testes e depois que já está em produção)	Tempo para arrumar um defeito (apenas durante o período de testes).
Tempo do teste	Tempo médio para encontrar um defeito Tempo necessário para executar um teste Tempo disponível para o esforço de teste
Tipos de defeitos (falhas de sistema, falhas de comunicação, entre outros)	Tipos de defeitos
Custo em testar	Custo do teste
Custo de desenvolvimento	Custo do teste x custo do desenvolvimento
Defeitos corrigidos que resultaram em novas ocorrências	Taxa de problemas encontrados na Correção de Defeitos

A partir das informações apresentadas, é possível concluir que os dados já capturados pela equipe de testes possibilitam a geração de um bom número de métricas de teste que, por sua vez, atendem de várias formas aos objetivos prioritários da empresa, conforme relacionado na Tabela 4.16. Cada uma das métricas de teste é associada a vários objetivos que, na tabela, são representados pelos números de cada um deles, conforme apresentado anteriormente.

Tabela 4.16: Relação entre os objetivos da empresa e as métricas de teste

Métrica \ Objetivo	1	2	3	4	5	6	7	8	9	10	11	12
Número de ocorrências	X					X			X			
Status das ocorrências	X					X			X			
Taxa de ocorrências válidas	X					X			X			
Índice de densidade dos defeitos	X		X	X		X			X	X	X	
Efetividade e eficiência dos testes	X		X	X		X			X	X	X	
Situação ou tendência dos defeitos em função do tempo	X		X	X		X	X		X		X	
Providências adotadas em relação aos defeitos	X		X	X		X	X		X		X	
<i>Zero Bug Bounce</i>	X		X	X		X	X		X		X	
Densidade dos defeitos residuais	X		X	X		X	X		X		X	
Defeitos encontrados x Defeitos corrigidos	X		X	X		X	X		X		X	
Ocorrências pendentes de correção	X		X	X		X	X		X		X	
Ocorrências resolvidas que ainda não foram retestadas	X		X	X		X	X		X		X	
Tempo para arrumar um defeito (apenas durante o									X		X	

período de testes).												
Tempo médio para encontrar um defeito	X	X	X	X		X	X		X		X	
Tempo necessário para executar um teste		X	X	X		X	X		X		X	
Tempo disponível para o esforço de teste		X	X	X		X	X		X		X	
Tipos de defeitos	X					X						X
Custo do teste		X				X					X	
Custo do teste x custo do desenvolvimento		X				X					X	
Taxa de problemas encontrados na Correção de Defeitos	X					X			X		X	

A proposta foi elaborada a partir das informações fornecidas pela pessoa responsável pela área de testes e qualidade, que, por sua vez, tem a noção da importância do uso de métricas de teste. Para que seja feita a implementação de um programa de métricas de teste, é necessário apresentar à gerência a importância das métricas e os possíveis resultados da utilização da mesma, visando à autorização para o uso.

De acordo com a informação recebida da empresa, as equipes de teste e de desenvolvimento normalmente são apenas comunicadas das decisões gerenciais, devendo seguir as determinações. De qualquer forma, sugere-se que, para uma adequada utilização das métricas, as equipes sejam bem orientadas a respeito, pois, conforme mencionado em capítulos anteriores, o envolvimento da equipe é fundamental, e, para que isso ocorra, os membros da equipe devem compreender exatamente o trabalho que estão executando.

Em reunião com a responsável pela área de testes, foi mencionado que as métricas devem ser reportadas à diretoria da empresa, preferencialmente através de gráficos, que é o método usual na empresa. Sugere-se que as métricas sejam utilizadas para acompanhar o andamento do projeto, tanto pela diretoria, como pela gerência ou equipe de testes, e não apenas no final do trabalho.

A conclusão, a partir da situação analisada, é que a maior parte dos dados necessários para a geração das métricas relacionadas já são capturados normalmente durante os testes. A partir da manipulação desses dados e do uso adequado das métricas, haverá um acréscimo de qualidade ao processo de testes. Em suma, as barreiras iniciais apresentadas pela empresa foram priorizadas, buscando esclarecer algumas dificuldades inerentes à utilização de métricas.

Detalhes sobre o uso das métricas de teste, assim como das possíveis melhorias no processo a partir das mesmas, não serão apresentados nessa proposta, pois foram devidamente tratados nos capítulos precedentes. O objetivo nesse momento é apresentar a uma empresa que não utiliza métricas de teste, por julgar que toma muito tempo ou que a relação custo/benefício não compensa, as informações importantes que podem ser obtidas através do uso de métricas, além de mostrar que muitos dos dados necessários para gerá-las já são usualmente capturados durante os testes.

5 EXEMPLO COMENTADO DE USO DE MÉTRICAS DE TESTE DE SOFTWARE

5.1 Contextualização

A empresa apresentada como exemplo trabalha com diversos produtos, possuindo paradigmas e particularidades diferenciadas e, em função disso, possui vários projetos para atender as demandas de software. Tais projetos são divididos entre diferentes equipes de teste de software.

Estamos analisando o trabalho de uma equipe específica, que, por confidencialidade, trataremos como equipe A.

Os testes são efetuados manualmente, com o auxílio de planilhas Excel, onde são informadas as entradas e os resultados esperados. No momento, não há interesse da empresa em adquirir nenhuma ferramenta para automação dos testes.

Os registros dos testes são feitos com a impressão da tela que mostra o defeito (*Print Screen*). Não utilizam simplesmente a informação PASSOU/FALHOU, pois não há garantia do teste efetuado.

As métricas não representam aumento significativo nos custos do projeto. No início, são necessárias mais horas de trabalho diárias, porém, com a evolução do uso das métricas, a tendência é ocorrer diminuição no número de horas, em função da otimização dos processos.

5.2 Escolha das Métricas de Teste

Há alguns anos não utilizavam métricas no setor da equipe A. A empresa usava métrica de unidade por demanda², porém só funcionava para o sistema legado, onde o paradigma de linguagem escrito é Clipper. O legado tem peculiaridade diferente, sendo que a maior parte das alterações afeta outras partes do sistema, diferentemente dos produtos desenvolvidos e testados especificamente pela equipe A.

As mudanças no uso de métricas de teste pela equipe A iniciaram com a contratação de uma analista de testes que possuía conhecimento sobre métricas de teste. Anteriormente, os empregados da empresa não eram receptivos a novas idéias, porém foram contratados empregados do mercado, com nova visão, objetivando a definição de

² O testador utiliza a metade do tempo que o desenvolvedor para testar um determinado produto.

estimativas e métricas para construção de indicadores de desempenho. Não houve problema de rejeição da inserção de métricas no processo de testes de software, pois a equipe é formada por pessoas que têm noção da importância das métricas.

A situação atual do uso de métricas de teste na equipe A demorou aproximadamente dois anos para ser ajustada e, atualmente, encontra-se em processo de ampliação, com a inclusão de novas métricas. Tal modelo está em estudo para expansão às demais equipes da empresa.

5.3 Métricas Utilizadas

5.3.1 Primeira Etapa

A primeira métrica implantada pela equipe A foi a estimativa de esforço, baseada nos padrões de complexidade apresentados na Figura 5.1. Índices padrão foram obtidos a partir de pesquisa na Internet. A equipe demorou aproximadamente um ano e meio para ajustá-los de acordo com a realidade da empresa. Foram efetuados três ciclos de comparação de estimativas, até o cenário atual, sendo que agora as estimativas têm uma diferença de no máximo 30 minutos entre o tempo previsto e o efetivo.

Um dos problemas que são vistos como barreira para as organizações inserirem métricas em suas atividades rotineiras é a necessidade de realização de ajustes nos padrões, quantas vezes forem necessárias para obter uma maior assertividade.

5.3.1.1 Exemplo Um

O exemplo apresentado na Figura 5.1 mostra a estimativa de teste para duas atividades específicas. O primeiro passo consiste em definir o tipo de atividade, se é criação, alteração ou validação, e a complexidade da mesma. No caso, temos uma alteração e uma validação de complexidade média, cujos valores padrão são 0,9 e 0,5, respectivamente, que somam 1,4. Multiplicando 1,4 pelo valor padrão referente à complexidade média, que é 1,5, temos o resultado de 2,1, que é multiplicado por 2, o fator médio de produtividade. Desta forma, a estimativa total de tempo para a execução dos testes das duas atividades é de aproximadamente 4h.

Planilha de Estimativa de Teste

TOTAL					
Categoria					
Tempo (horas)					
4					

FUNCIONALIDADES					
Descrição da Atividade					
	1	2	3	4	5
Busca por % LIKE% - Mostra Nome do Cliente, CPF	A				
Mensagem "Registro já incluído"	V				
C	0,0	0,0	0,0	0,0	0,0
A	0,9	0,0	0,0	0,0	0,0
V	0,5	0,0	0,0	0,0	0,0
Soma (Categoria)	1,4	0,0	0,0	0,0	0,0
Complexidade (tamanho)	M				
	2,1	FALSO	FALSO	FALSO	FALSO

Complexidade	
1,2	B Baixa
1,5	M Média
1,8	A Alta
1,2	C Criação
0,9	A Alteração
0,5	V Validação

Figura 5.1: Exemplo de planilha Excel utilizada para estimativa do tempo de teste

Na Figura 5.2 temos uma outra situação em que é feita a estimativa do tempo de teste de duas atividades de validação, identificadas com a letra V na planilha. Seguindo o mesmo raciocínio do cálculo apresentado na Figura 5.1, temos a estimativa de tempo total de 3h para os testes. O campo referente à validação é preenchido com 1,0, que é a soma do valor padrão das duas atividades, em que cada uma vale 0,5. Os valores referentes à criação e alteração foram mantidos em zero, pois não há nenhuma atividade correspondente. A complexidade foi definida como média, e marcada com a letra M na planilha. O valor padrão 1,5 foi atribuído e multiplicado por 1,0, que é a soma das categorias. Para finalizar, o valor 1,5 foi multiplicado por 2,0, que é o fator de produtividade padrão.

Transações		
Descrição da Atividade		
Transação Completa e Comprovante		V
Tela (Mensagem Informativa, Impeditiva e Navegação)		V
Criação		0,0
Alteração		0,0
Validação		1,0
Soma (Categoria)		1,0
Complexidade (tamanho)		M
		1,5
Tempo (2,0 h)	2,00	3,0

Figura 5.2: Tempo de execução dos testes

A lógica utilizada para o cálculo do valor referente à complexidade está relacionada na Figura 5.3. Quando a tarefa for de complexidade alta (“A”), a soma das categorias é multiplicada por 1,8, quando for baixa (“B”), é multiplicada por 1,2, e quando for média (“M”), que é o caso do exemplo, a multiplicação é por 1,5. O tempo de teste é o produto do valor da complexidade da atividade e o fator de produtividade, ou fator médio, que é padronizado em 2,0 na empresa.

<p>Se o valor da complexidade = “A” Então Soma (Categoria) * 1,8</p> <p>Se o valor da complexidade = “M” Então Soma (Categoria) * 1,5</p> <p>Se o valor da complexidade = “B” Então Soma (Categoria) * 1,2</p> <p>Tempo de Teste = Complexidade da Atividade * Fator de Produtividade (Fator Médio) TT = CA * FP TT = 1,5 * 2,00 TT = 3 horas</p>
--

Figura 5.3: Lógica utilizada para o cálculo

5.3.1.2 Exemplo Dois

Em um determinado sistema constantemente são realizadas novas implementações e muitas manutenções com a finalidade de obter uma qualidade de software e proporcionar aos seus usuários um maior desempenho, facilidade em sua navegação e segurança.

O tempo de planejamento dos testes para cada funcionalidade é estimado conforme apresentado a seguir para as novas implementações ou manutenções a serem realizadas. Inicialmente, as tarefas a serem estimadas são relacionadas em uma tabela, especificando o que foi solicitado e a análise do caso, conforme Tabela 5.1.

Tabela 5.1: Tarefas a serem estimadas

Item	Solicitação	Análise
1	Atualmente no cadastro de clientes, a “busca” é realizada pela matrícula do cliente no sistema interno, o que ocasiona ineficiência no atendimento como morosidade e duplicidade ou mais de registros.	Será alterada a busca por %LIKE% para que sejam exibidos os registros conforme o nome do cliente com chave no CPF. Não irá permitir duplicidade. Na tentativa de inserção do mesmo CPF, exibir uma mensagem “Registro já incluído”.
2	O sistema atualmente não possui integração com convênios, atendendo somente particular.	Será alterado o campo referente às formas de atendimento, inserindo as opções de atendimento através dos convênios

		existentes.
3	O sistema não permite pagamento por cartão de crédito, impedindo o aumento de atendimentos. Ver a viabilidade para crédito e débito.	Será alterado o campo referente às formas de pagamento, inserindo a opção de uso de cartão de crédito, tanto débito como crédito.
4	Os atendentes realizam anotações em fichas para manter um histórico dos clientes. Ver a viabilidade para manter um histórico no sistema.	Será criado um módulo para manutenção do histórico dos clientes. Tal módulo permitirá inclusão, alteração e exclusão de histórico. Os históricos dos clientes podem ser impressos.
5	O sistema em inúmeras vezes “caiu” tornando inviável a sua operação.	Exige esforço de outra área, da infraestrutura.

A estimativa do tempo de planejamento de testes é feita a partir da combinação do tipo de tarefa e sua complexidade, cujos padrões definidos pela equipe estão relacionados na Tabela 5.2.

Tabela 5.2: Planilha utilizada para estimativa do tempo de planejamento dos testes

Variável	Complexidade	Número em horas
Criação (C)	Alta (A)	8
Criação (C)	Média (M)	6
Criação (C)	Baixa (B)	4
Alteração (A)	Alta (A)	3
Alteração (A)	Média (M)	2
Alteração (A)	Baixa (B)	1

As informações na Figura 5.4 explicam os dados da Tabela 5.2. Por exemplo, se tivermos uma tarefa de criação de caso de teste (“C”), cuja complexidade é alta (“A”), o tempo de desenvolvimento do caso de teste é estimado em 8 horas.

<p>Se a complexidade = “A” e “C” Então Tempo de Desenvolvimento CT = 8</p> <p>Se a complexidade = “M” e “C” Então Tempo de Desenvolvimento CT = 6</p> <p>Se a complexidade = “B” e “C” Então Tempo de Desenvolvimento CT = 4</p> <p>Se a complexidade = “A” e “A” Então Tempo de Desenvolvimento CT = 6</p> <p>Se a complexidade = “M” e “A” Então</p>
--

Tempo de Desenvolvimento CT = 4
 Se a complexidade = “B” e “A”
 Então
 Tempo de Desenvolvimento CT = 2

Figura 5.4: Lógica para estimativa do tempo de planejamento dos testes

Considerando as tarefas da Tabela 5.1, temos uma estimativa de 32 horas para criação dos casos de teste, conforme especificado na Tabela 5.3. O primeiro item, por exemplo, diz respeito a uma tarefa de alteração com complexidade média, cujo tempo estimado é de 4 horas, conforme padrões apresentados anteriormente na Tabela 5.2 e na Figura 5.4.

Tabela 5.3: Tempos estimados para a criação dos casos de teste

Item	Descrição	Tempo Estimado
1	Alteração com complexidade média	4h
2	Criação com complexidade alta	8h
3	Criação com complexidade alta	8h
4	Alteração com complexidade média	4h
5	Criação com complexidade alta	8h
Total		32h

As métricas para estimar o tempo de geração dos casos de teste não estão sendo usadas atualmente pela equipe A. Os padrões alteram com a evolução de cada colaborador, que passa a ter mais conhecimento do negócio do produto e dos processos internos da organização em que trabalha. Por esta razão, a área de metodologia de processos está analisando uma nova opção juntamente com a equipe, onde no primeiro momento será realizado um Projeto Piloto que posteriormente será estendido às demais equipes.

O tempo estimado para a execução dos testes das tarefas relacionadas na Tabela 5.1 foi calculado em aproximadamente 38h, conforme descrito na Figura 5.5. A lógica do cálculo é semelhante aos exemplos apresentados anteriormente.

Planilha de Estimativa de Teste

TOTAL					
Categoria					
Tempo (horas)	38				

FUNCIONALIDADES					
Descrição da Atividade	1	2	3	4	5
Busca por %LIKE% - Mostra Nome do Cliente, CPF	A				
Mensagem "Registro já incluído"	V				
Verificar a opção de débito via cartão de crédito			A		
Verificar a opção de crédito via cartão de crédito			A		
Validar a transação de débito via cartão de crédito			V		
Validar a transação de crédito via cartão de crédito			V		
Validar a impressão do comprovante de débito			V		
Validar a impressão do comprovante de crédito			V		
Verificar o módulo de inclusão de histórico de cliente				C	
Verificar o módulo de alteração de histórico de cliente				C	
Verificar o módulo de exclusão de histórico de cliente				C	
Validar o módulo de inclusão de histórico de cliente				V	
Validar o módulo de alteração de histórico de cliente				V	
Validar o módulo de exclusão de histórico de cliente				V	
Impressão dos históricos de clientes				C	
Verificar relatório com histórico de um determinado cliente				V	
C	0,0	0,0	0,0	4,8	0,0
A	0,9	0,0	1,8	0,0	0,0
V	0,5	0,0	2,0	2,0	0,0
Soma (Categoria)	1,4	0,0	3,8	6,8	0,0
Complexidade (tamanho)	M	A	A	M	
	2,1	0,0	6,8	10,2	FALSO
Tempo (2,0 h)	2,00	4	0	14	20
					38

Complexidade	
1,2	B Baixa
1,5	M Média
1,8	A Alta
1,2	C Criação
0,9	A Alteração
0,5	V Validação

Figura 5.5: Estimativa de tempo de teste para os requisitos apresentados na Tabela 5.1

5.3.2 Segunda Etapa

O segundo passo na implantação de métricas foi o gerenciamento das estimativas.

É importante observar que as métricas estão intimamente ligadas com as estimativas, ou seja, ao realizar uma estimativa deve-se analisar qual a maneira de medi-la corretamente. O resultado da medição é uma métrica, que passa a compor o quadro de indicadores.

5.3.2.1 Exemplo Três

O gerenciamento das estimativas é efetuado a partir dos dados no gráfico da Figura 5.6 e na Tabela 5.4. No eixo X do gráfico constam todos os itens cujo tempo de teste foi estimado, e, no eixo Y, os tempos estimados e efetivamente realizados. A barra amarela mostra os tempos estimados e a barra verde, os realizados.

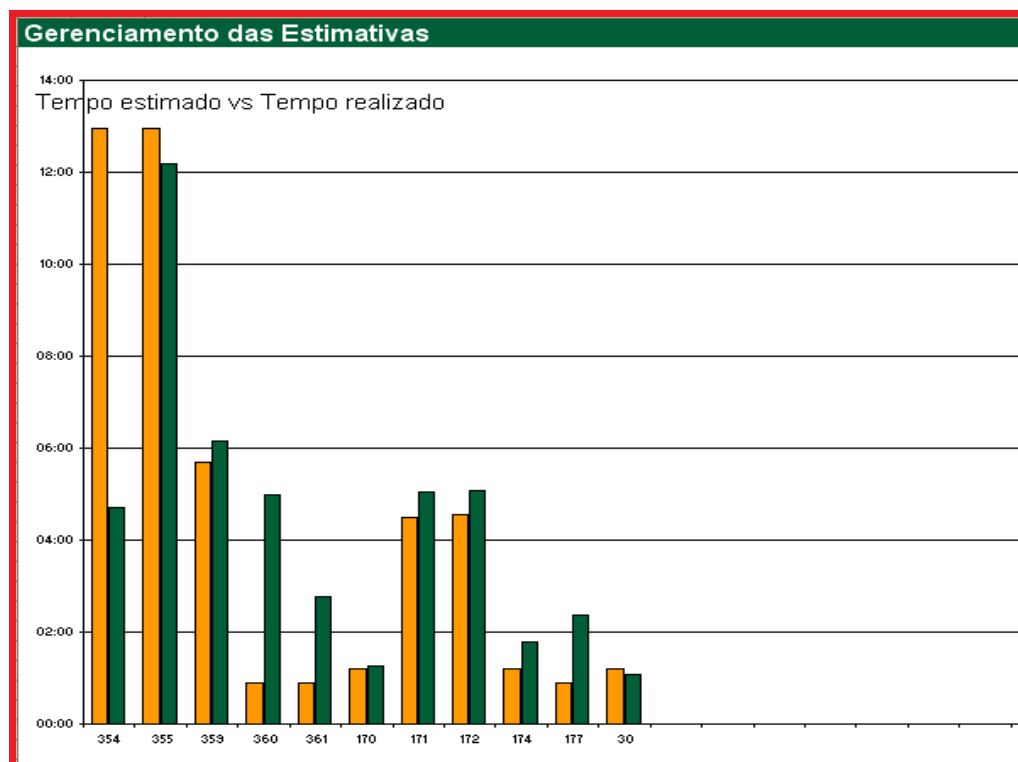


Figura 5.6: Gráfico utilizado para gerenciamento da estimativa do tempo de execução dos testes

Na Tabela 5.4, as informações do gráfico são detalhadas e divididas por sistema. Na tabela constam também o status do teste e a quantidade de erros encontrados. No item 354, por exemplo, foi estimado um tempo de 12h57min36seg para o teste, porém o tempo real foi de 4h43min5seg. Neste caso, há uma grande diferença entre o tempo estimado e o realizado. A diferença total entre o estimado e o realizado foi de 32min, sendo que o tempo total estimado foi 46h51min e o realizado, 47h23min. No total, a diferença de 32min pode ser considerada pequena, porém se a análise for feita por testador ou por tarefa a discrepância aumenta, mas não significativamente.

Tabela 5.4: Análise das estimativas da fase de testes

Sist	Item	Profissional TE	Estimado	Realizado	Status	Erros
195	354	Testador 1	12:57:36	04:43:05	Encerrado	1
195	355	Testador 2	12:57:36	12:11:12	Encerrado	0
195	359	Testador 3	05:42:00	06:08:41	Encerrado	0
195	360	Testador 4	00:52:48	04:58:26	Encerrado	2
195	361	Testador 2	00:52:48	02:46:58	Encerrado	1
196	170	Testador 1	01:12:00	01:16:02	Encerrado	0
196	171	Testador 1	04:30:00	05:03:20	Encerrado	1
196	172	Testador 3	04:33:36	05:04:46	Encerrado	0
196	174	Testador 4	01:12:00	01:47:41	Encerrado	0
196	177	Testador 3	00:52:48	02:22:16	Encerrado	1
197	30	Testador 2	01:12:00	01:05:30	Encerrado	3
Totais			46:51	47:23		

Na Tabela 5.5, é apresentada a diferença entre o tempo estimado e o realizado, por sistema, cujos valores foram calculados a partir dos dados da Tabela 5.4. Neste caso, a estimativa foi correta para o sistema 197, com diferença de apenas 7 minutos. No sistema 195 o tempo foi superestimado, e, no 196, subestimado.

Tabela 5.5: Análise do tempo por sistema

Sistema	Tempo Estimado	Tempo Realizado	Diferença	Observações
195	33h20min	30h46min	2h34min	Tempo superestimado
196	12h19min	15h32min	-3h13min	Tempo subestimado
197	1h12min	1h5min	7min	Tempo bem estimado

Uma das métricas de teste analisadas neste exemplo é a relação entre a quantidade de erros na fase de teste e a quantidade de erros na fase de homologação. Foram encontrados nove erros na fase de teste e 11 na homologação. Isso mostra que pode ter havido negligência na fase de testes, pois não detectaram os erros. Outra possibilidade é que a fase de testes tenha sido abreviada por problemas de atraso no cronograma e, em função disso, os erros foram detectados apenas na homologação. E por fim, há também a possibilidade de que alguns erros detectados na fase de testes não tenham sido corrigidos e, desta forma, tenham sido repassados à fase de homologação.

5.3.3 Terceira Etapa

Na sequência, começaram a ser tratadas as métricas abaixo relacionadas.

5.3.3.1 Índice de Densidade de Defeitos – Número de Defeitos Encontrados

A quantidade de defeitos na fase de teste deve ser maior do que a quantidade de defeitos na fase de homologação, que também deve ser maior do que a quantidade de defeitos na fase de aceitação. Tal regra é fundamental em função da diminuição do custo, pois, quanto mais tarde for encontrado o defeito, maior será o custo para consertá-lo.

Atualmente não existe um documento padrão demonstrando tais indicadores, porém são geradas consultas para a comparação dos resultados, através das informações gravadas no banco de dados. A empresa está estudando um formato adequado de documento.

5.3.3.2 Índice de Severidade de Defeitos

Essa métrica também é utilizada para definir a eficiência do testador. Para isso é utilizada a matriz GUT (Figura 5.7), onde são definidos os índices de gravidade e tendência dos defeitos. Por exemplo, se um testador acha um defeito e outro testador acha dez defeitos no mesmo período. A capacidade do primeiro testador não pode ser

questionada em função disso. Tal defeito pode ser bastante severo, enquanto que os dez defeitos encontrados pelo segundo testador podem ser apenas cosméticos.

A gravidade dos defeitos encontrados é definida pelo líder de teste, que é isento. Tal tarefa não pode ser delegada ao testador, pois haveria o risco de todos os defeitos serem considerados prioritários.

É preciso, nessas situações, analisar também os casos de teste, através da revisão dos mesmos, pois os problemas encontrados podem estar no planejamento dos testes e não no testador.

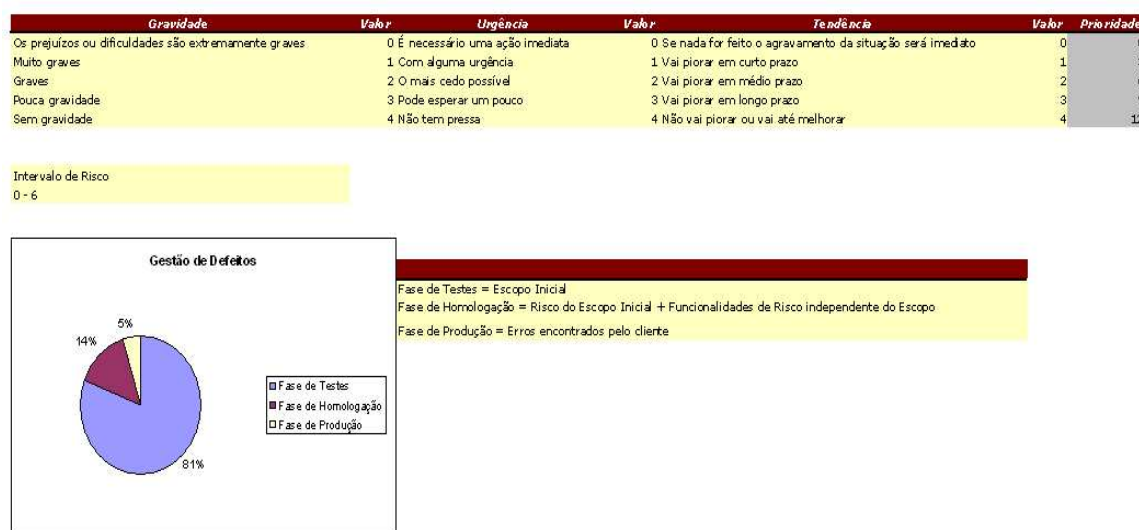


Figura 5.7: Matriz GUT

5.3.3.3 Distribuição de Defeitos por Funcionalidade

Conforme já estudado no capítulo 2, a métrica referente à distribuição dos defeitos por funcionalidade permite identificar a funcionalidade mais problemática. Na equipe A, a análise da distribuição dos defeitos por funcionalidade é feita utilizando uma planilha como a apresentada na Tabela 5.6, em que as tarefas equivalem às funcionalidades. Os defeitos encontrados são relacionados na última coluna. No exemplo, os testes ainda não foram concluídos, sendo assim, ainda há tarefas pendentes.

Tabela 5.6: Relação das tarefas executadas nos testes

Base de Dados							
Sist	Item	Descrição	Profissional TE	Estimado	Realizado	Status	Repiques
195	373	TAREFA 71029	Testador 1	12:00:00	13:19:26	Pendente	
195	398	TAREFA 74365	Testador 1	03:36:00	01:30:31	Encerrado	
195	399	TAREFA 74725	Testador 1	04:12:00	05:33:05	Pendente	
195	400	TAREFA 75114	Testador 1	04:30:00	01:15:08	Encerrado	
195	401	TAREFA 76204	Testador 1	03:00:00	01:31:34	Encerrado	
195	402	TAREFA 75821	Testador 1	10:26:24	09:16:46	Encerrado	1
195	404	TAREFA 73512	Testador 1	06:50:24	02:08:25	Encerrado	
195	405	TAREFA 73706	Testador 1	09:00:00	01:35:51	Encerrado	
195	406	TAREFA 73952	Testador 2	10:48:00	05:18:59	Encerrado	
195	409	TAREFA 74741	Testador 4	02:09:36	01:53:27	Pendente	
195	410	TAREFA 74841	Testador 2	06:00:00	04:51:54	Encerrado	
195	411	TAREFA 74860	Testador 4	03:36:00	02:02:01	Pendente	
195	413	TAREFA 75250	Testador 4	04:30:00	03:13:18	Pendente	
195	414	TAREFA 75251	Testador 2	04:30:00	00:00:00	Pendente	
195	415	TAREFA 75375	Testador 2	06:00:00	01:25:57	Pendente	
195	416	TAREFA 75447	Testador 4	02:24:00	02:02:40	Pendente	
195	417	TAREFA 75502	Testador 2	06:00:00	00:00:00	Pendente	
195	418	TAREFA 75515	Testador 4	02:24:00	01:44:02	Encerrado	
195	419	TAREFA 75689	Testador 4	02:09:36	01:14:48	Pendente	
195	420	TAREFA 75734	Testador 4	02:09:36	00:00:00	Pendente	
195	421	TAREFA 75804	Testador 2	05:42:00	00:00:00	Pendente	
195	422	TAREFA 75835	Testador 2	04:30:00	00:00:00	Pendente	
195	423	TAREFA 75841	Testador 5	01:12:00	00:00:26	Encerrado	
195	424	TAREFA 75907	Testador 5	01:30:00	00:00:53	Encerrado	
195	425	TAREFA 75909	Testador 5	01:12:00	00:13:52	Encerrado	
195	426	TAREFA 76092	Testador 2	03:00:00	03:45:51	Encerrado	
195	427	TAREFA 76508	Testador 2	04:12:00	07:28:48	Encerrado	
195	428	TAREFA 76525	Testador 2	03:00:00	03:59:43	Pendente	
195	429	TAREFA 76595	Testador 5	04:12:00	01:26:56	Encerrado	
195	430	TAREFA 76604	Testador 5	02:42:00	00:13:54	Encerrado	
195	431	TAREFA 76647	Testador 5	01:45:36	00:32:57	Encerrado	1
195	432	TAREFA 76718	Testador 2	01:45:36	00:00:00	Pendente	
195	433	TAREFA 76802	Testador 5	01:45:36	00:00:33	Encerrado	
195	434	TAREFA 76805	Testador 5	01:45:36	00:00:45	Encerrado	
195	435	TAREFA 76876	Testador 2	01:45:36	00:00:00	Pendente	
195	436	TAREFA 77239	Testador 5	01:45:36	00:00:00	Pendente	
195	437	TAREFA 77651	Testador 2	01:45:36	00:00:00	Pendente	
195	438	TAREFA 77774	Testador 5	01:45:36	00:00:54	Encerrado	
195	439	TAREFA 77570	Testador 5	01:45:36	01:06:29	Encerrado	
195	440	TAREFA 77797	Testador 2	01:45:36	00:00:00	Pendente	
195	441	TAREFA 77741	Testador 5	01:45:36	02:12:39	Encerrado	
196	190	TAREFA 75397	Testador 3	02:42:00	01:39:34	Encerrado	1
196	191	TAREFA 76315	Testador 3	03:00:00	00:17:27	Encerrado	
196	192	TAREFA 75681	Testador 3	02:42:00	01:05:47	Pendente	
196	194	TAREFA 73415	Testador 3	04:12:00	04:21:04	Pendente	1
196	195	TAREFA 75373	Testador 3	04:12:00	00:43:36	Encerrado	1
196	196	TAREFA 75479	Testador 3	04:12:00	01:13:56	Encerrado	1
196	197	TAREFA 76075	Testador 3	03:00:00	01:55:50	Encerrado	
196	198	TAREFA 77077	Testador 3	01:30:00	01:08:21	Encerrado	1
196	199	TAREFA 77722	Testador 3	02:42:00	01:29:37	Encerrado	
196	200	TAREFA 77748	Testador 3	03:00:00	01:02:26	Encerrado	

5.3.3.4 Defeitos Encontrados X Defeitos Corrigidos – Índice de Correção dos Defeitos – Eficácia na Remoção dos Defeitos.

A planilha da Tabela 5.6 permite também o acompanhamento dos defeitos encontrados e corrigidos. Essa métrica é utilizada para encaminhamento dos dados à gerência quando está vencendo o prazo para entrega do produto ao cliente e existem defeitos pendentes de correção. Além das pendências, são acrescentadas informações quanto ao impacto que tais defeitos podem causar, caso sejam repassados ao cliente,

conforme exemplo na Figura 5.8. Neste caso, o relatório foi gerado quando ainda restavam cinco tarefas pendentes de correção, das relacionadas na Tabela 5.6.

Matriz de Impacto

Período de homologação: de 18/09/2009 à 24/09/2009.

Data da instalação: 25/09/2009

Subsist.	Item	Descrição	Funcionamento nas Filiais ou na Matriz sem a correção	Impactante?	Não Impactante?	Observação
195	373 / 71029	Integração Usuários	A integração está funcionando corretamente, porém, ao realizar a impressão de movimentação retroativa de acompanhamento de metas, o usuário não está exibindo as vendas realizadas.		X	No orçamento está contemplada a implementação de visualização retroativa de acompanhamento de metas.
195	399 / 74725	Processo de Vendas no Cartão	O arquivo VC001.txt está sendo gerado sem o campo "VERSÃO".		X	-
195	409 / 74741	Lista de Preços	O filtro por valor parcelado ou à vista, independentemente da opção selecionada, exibe todos os registros.		X	-
195	411 / 74860	Exclusão de Venda	Não está gravando os valores das taxas.	X		Solicitada análise, pois não foram identificados problemas no descadastramento de parcelas para o setor de utensílios.
196	194 / 73415	Dados Bancários na Venda de Móveis	Não está reaplicando o nome da agência bancária.	X		Este ponto é impactante, pois estará armazenando uma informação em branco no campo agência. É importante que a correção seja enviada o quanto antes para entrada nesta versão.

Figura 5.8: Relatório de análise de impacto dos defeitos pendentes

5.3.3.5 Definição de Prioridade dos Testes

É necessário priorizar os testes a serem executados, do mais alto ao mais baixo. A definição é feita e no final do projeto é verificado se está adequada ou não. A equipe A está trabalhando com essa métrica e o resultado é positivo.

O analista de projetos define a priorização dos testes, que é divulgada por e-mail à equipe. Não existe atualmente um documento padrão. A empresa está estudando um formato adequado de documento.

5.3.3.6 Revisão dos Artefatos de Teste

Verifica se os testes que foram informados como ENCERRADOS realmente não apresentaram problema na utilização do produto pelo cliente.

Atualmente está sendo documentado um novo formato para revisão, pois as revisões são efetuadas informalmente, sem contabilizar o tempo para a atividade. A ferramenta utilizada atualmente não disponibiliza opção para contar a produtividade dos funcionários. Atualmente os testadores não estão sendo avaliados, mas no futuro será um dos itens, com a padronização. Importante salientar que os colaboradores não serão auditados e sim a verificação dos processos a serem cumpridos.

5.3.3.7 Quantidade de Vezes de Retorno do Defeito

Tal métrica pode identificar problemas no desenvolvimento, ou então no artefato de negócios, que pode não ter sido bem explicado. A informação da quantidade de vezes de retorno do defeito pode ser obtida através da coluna REPIQUES, da Tabela 5.6.

Quando o defeito retorna muitas vezes, é informado ao responsável pela equipe de desenvolvimento, que é terceirizada, para que sejam tomadas as devidas providências.

Esses dados também são apresentados à gerência nos relatórios específicos, conforme exemplo na Figura 5.9, onde consta um resumo dos defeitos que retornaram para correção, com a informação da quantidade de vezes da ocorrência. No relatório, os defeitos são relacionados de acordo com a fase da ocorrência, ou seja, fase de testes e fase de homologação.

1 - Versão 5.21 - Fase de Testes

1.1 - Abertura e Reabertura de Erros

Subsistema 195		
Item	Resumo	Quantidade
333	Mensagem Informativa - Cancelamento e Reativação de Contas	1
347	Alteração do Tipo de Produto	1
348	Classificação das Vendas	1
350	Emissão de Nota Fiscal	3
Total		6

Subsistema 196		
Item	Resumo	Quantidade
155	Emissão de orçamento	5
168	Impressão Regulamento	3
Total		8

2 - Versão 5.21 - Fase de Homologação

2.1 - Abertura e Reabertura de Erros

Subsistema 195		
Item	Resumo	Quantidade
Homologação	Proposta de Crédito + Emissão de orçamento	3
Total		3

2.2 - Erros fora de Escopo (Infraestrutura, Servidor ...)

Subsistema 195		
Item	Resumo	Quantidade
Homologação	Ambiente - Após a instalação do plano em ambiente de homologação, ao confirmar uma transferência de conta, está gerando tabelas temporárias no banco, sendo necessário copiar para outro servidor os registros. Foi realizada novamente a cópia de produção para o ambiente de testes, com a versão 5.20 o funcionamento da transferência foi realizado, porém, ao instalar a versão 5.1 o problema voltou a surgir. Foi aberto chamado para correção.	1
Total		1

Figura 5.9: Relatório de retorno dos defeitos

5.4 Resultados das Métricas

Os indicadores são orientados ao serviço, ou seja, se aumentar o serviço, aumenta o desenvolvimento. A evolução ou a involução dos resultados do produto reflete no trabalho.

Os resultados (métricas dos defeitos) são repassados à gerência através de relatórios, e são utilizados para acompanhamento da performance da empresa que presta o serviço de desenvolvimento demandando alterações, se necessário, ou até mesmo para geração de glosas. Além disso, a analista de projetos gera informações necessárias, baseadas em métricas existentes, para que a gerência possa definir se a release está pronta para entrega ao cliente. Nesse caso é encaminhado o relatório de análise de impacto. O relatório é gerado a partir das informações existentes na planilha apresentada na Tabela 5.6, analisando os casos de teste pendentes para definir o impacto dos defeitos.

A analista também utiliza as métricas para acompanhar o andamento do trabalho em relação ao cronograma. A evolução dos testes é registrada no banco de dados. Sempre

que houver necessidade de verificar o andamento do trabalho, é efetuada consulta no banco de dados para gerar uma planilha semelhante à apresentada na Tabela 5.6.

6 CONCLUSÃO

A título de encerramento do trabalho, é agora apresentada uma síntese do mesmo, contendo uma avaliação sobre as propostas e os resultados alcançados, assim como eventuais limitações. É feita ainda uma análise da importância do trabalho e são trazidas sugestões para a realização de estudos e trabalhos subsequentes.

Para a realização do trabalho, houve um grande empenho em estudar bastante sobre testes de software, de uma forma geral, para depois tratar especificamente das métricas de teste de software. Para uma melhor compreensão do assunto, foram feitas pesquisas em livros, e, principalmente, na Internet. Houve dificuldade em conseguir livros tratando especificamente de métricas de teste, pois a maior parte deles teriam que ser trazidos de fora do país, o que dificultaria bastante em termos financeiros. Em função disso, a Internet foi uma grande aliada, possibilitando o acesso às informações.

O ponto de partida foi a realização de pesquisa junto às empresas de TI do Rio Grande do Sul. Foi um processo difícil, pois, para conseguir uma amostra significativa de empresas, foi necessário ter muita persistência nesta fase do processo, efetuando muitos encaminhamentos de e-mails, com solicitações de preenchimento do questionário e explicações da importância do trabalho. É conhecida a dificuldade existente em obter respostas às pesquisas de uma forma geral. Apesar das dificuldades, o resultado da pesquisa foi fundamental para definir o escopo do trabalho.

A ideia inicial era trabalhar na melhoria do uso de métricas de teste de software, porém, com base no resultado da pesquisa, foi possível verificar que poucas empresas utilizavam as métricas de teste, e isso alterou o foco do trabalho. O objetivo passou a ser a motivação ao uso das métricas de teste, através da apresentação das informações relacionadas ao assunto da forma mais objetiva possível.

Para apresentar assuntos que motivassem o uso das métricas de teste, foram avaliadas diversas questões relacionadas ao tema. Inicialmente, foi necessário apresentar conceitos gerais sobre métricas, para depois abordar o assunto inserido no contexto dos testes de software. Foi feito um trabalho exaustivo de pesquisa, buscando apresentar as mais variadas métricas de teste existentes, com o intuito de atender o maior número de objetivos para o seu uso. Questões práticas também foram consideradas, como a forma de implementar um programa de métricas e exemplos de uso. Os problemas relacionados ao uso das métricas foram bem trabalhados, pois é fundamental que as métricas sejam utilizadas de forma correta, evitando que erros sejam cometidos, tanto do ponto de vista de relacionamento entre a equipe como de interpretações equivocadas dos dados.

Podemos considerar que, em meio a algumas limitações, este trabalho atingiu os objetivos previstos. Algumas questões adicionais poderiam ter sido tratadas, porém

ficam como sugestão para trabalhos futuros, como, por exemplo, colocar em prática a proposta apresentada no capítulo 4, acompanhando a empresa durante a implantação das métricas, analisando cada uma das etapas.

Uma outra ideia que surgiu a partir deste trabalho, e que será colocada em prática futuramente, é a realização de palestras sobre o uso de métricas de teste nas empresas de TI, resumindo o que foi estudado até o presente momento sobre o assunto, com o objetivo de estimular o interesse das empresas.

A motivação é a mola mestra deste trabalho. Motivação para a escolha do tema, por um grande interesse na área de testes de software. Motivação do orientador, por indicar as métricas de teste para o trabalho de conclusão. Motivação da aluna para aprofundar mais o assunto, durante a execução do trabalho e para o futuro profissional, que levará como base todo o conhecimento adquirido com o estudo efetuado durante os dois semestres do trabalho de conclusão. E, por fim, a motivação das empresas para a utilização de métricas de teste, que é o objetivo principal do trabalho.

REFERÊNCIAS

- AGAPITO, R. **Métricas de Testes de Software (EDD e ERD)**. 2008. Disponível em: <<http://www.testexpert.com.br/?q=node/1084>>. Acesso em: mai. 2009.
- AMLAND, S. **Risk Analysis Fundamentals and Metrics for software testing including a Financial Application case study**. 1999. 20 f. In: 5th International Conference EuroSTAR, Barcelona, Espanha.
- BASTOS et al. **Base de Conhecimento em Teste de Software**. 2. ed. São Paulo: Martins Editora, 2007.
- BRADSHAW, S. **Software Test Metrics – A Practical Approach**. 2007. Disponível em: <<http://www.ddj.com/development-tools/199201553;jsessionid=ZPBMO5XUEHANRQE1GHOSKH4ATMY32JVN>>. Acesso em: mai. 2009.
- BRADSHAW, S. **Effective Metrics for Managing a Test Effort**. 23 f. Questcon Technologies. Disponível em: <<http://www.questcon.com>>. Acesso em mai. 2009.
- BRADSHAW, S. **Test Metrics: A Practical Approach to Tracking & Interpretation**. 2004. 10 f. Questcon Technologies. Disponível em: <<http://www.stickyminds.com>>. Acesso em mai. 2009.
- BRAUN, L. **Modelo Para Medição do ROI em Processos de Teste de Software**. 2007. 40 f. Trabalho de Conclusão de Curso - Centro Universitário Feevale, Instituto de Ciências Exatas e Tecnológicas, Novo Hamburgo.
- BRITO, R. D. **Do Ambiente de Desenvolvimento ao Ambiente de Produção: Os Testes de Software como Garantia de Qualidade**. 2000. 57 f. Trabalho de aproveitamento do curso (MBA – Gestão de Tecnologia da Informação) – Universidade Cândido Mendes, Osasco.
- BORYSOWICH, C. **Evaluating Test Metrics**. 2006. Disponível em: <<http://it.toolbox.com/blogs/enterprise-solutions/evaluating-test-metrics-13438>>. Acesso em jul. 2009.
- BURNSTEIN, I.; SUWANNASART, T.; CARLSON, C. R. **Developing a Testing Maturity Model, Part II**. Illinois Institute of Technology. Disponível em: <<http://www.stsc.hill.af.mil/crosstalk/1996/09/Developi.asp>>. Acesso em mai. 2009.
- CAMACHO, C. R. **Uso de Métricas no Processo de Teste de Software**. 2008. 10f. Trabalho da Disciplina CMP102 – Engenharia de Software (Programa de Pós-

Graduação em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

COPSTEIN, B. **Teste de Software**. 2006. 53 f. Disponível em: <<http://www.inf.pucrs.br/~copstein/CursoTeste/Dia4/>> . Acesso em: jun. 2009.

DEMILLO et al. **Software Testing and Evaluation**. California: The Benjamin/Cummings Publishing Company, 1987.

ELIAS, G. S.; WILDT, D. **Métricas para Melhoria Contínua de Código – Um Estudo de Caso com Java**. 2008. 8 f. In: Seminário de Informática – RS (SEMINFO RS), Faculdade Cenecista Nossa Senhora dos Anjos, Gravataí.

GARCIA, L. F. **Qualidade de Software**. 11 f. Disponível em: <<http://www.pdf-search-engine.com/aulas-preesman-ppt-pdf.html>>. Acesso em jul. 2009.

HERBERT, J. **Teste de Software: Idéias, Percepções e Práticas Locais e Globais**. 2009. 23 f. In: II Testing Day, PUCRS, Porto Alegre.

HETZEL, W. **Guia Completo ao Teste de Software**. Rio de Janeiro: Campus, 1987.

HUTCHESON, M. L. **Software Testing Fundamentals: Methods and Metrics**. Indianapolis: Wiley Publishing, Inc. 2003.

Software Engineering – Software Measurement Process. **ISO/IEC 15939**. 2002.

KANER, C.; BOND, W. P. **Software Engineering Metrics: What do They Measure and How Do We Know?** 2004. 12 f. In: 10º Simpósio Internacional de Métricas de Software.

KONDA, K. R. **Measuring Defect Removal Accurately – Test Metrics Sidebar**. 2005. Disponível em: <<http://stpcollaborative.com/knowledge/276-measuring-defect-removal-accurately-test-metrics-sidebar>>. Acesso em: mai. 2009.

MAGAZINE, A. **Testing Metrics**. 6 f. Disponível em: <<http://ucanyahoo.googlepages.com/2532853.doc>>. Acesso em: nov. 2009.

MAGAZINE, A. **Effective Test Metrics Management**. 2003. 4 f. Bangalore, India. Disponível em: <<http://www.stickyminds.com/sitewide.asp?Function=edetail&ObjectType=ART&ObjectID=6262>>. Acesso em: nov. 2009.

MAGAZINE, A. **Designing Effective Test Status Report**. 2008. 3 f. Disponível em: <<http://anujmagazine.blogspot.com/2008/06/designing-effective-test-status-reports.html>>. Acesso em: jul. 2009.

MAIA, J. R. C. **Use Métricas Adequadas. Garanta a Qualidade de Projeto Orientado a Objeto**. 7 f. Euax – Gestão de Processos. Disponível em: <<http://www.euax.com.br/art.00.index.shtml>> . Acesso em: out. 2009.

MANCORIDIS, S. **Topics in Metrics for Software Testing**. 2008. 39 f. Disponível em: <<http://www.cs.drexel.edu/~spiros/teaching/SE320/>>. Acesso em: mai. 2009.

- MARINHO, E. H. **Medição e Métricas de Software**. 40 f. Disponível em: <<http://eulerhm.googlepages.com/2-MedioeMtricas.pdf>>. Acesso em mai. 2009.
- McGREGOR, J. D.; SYKES, D. A. **A Practical Guide to Testing Object-Oriented Software**. [S.l.]: Addison-Wesley, 2001.
- MOCHAL, T. **Stay on Track With Testing Metrics**. 2001. Disponível em: <<http://www.zdnet.com.au/news/business/soa/Stay-on-track-with-testing-metrics/0,139023166,120261868,00.htm?omnRef=1337>>. Acesso em: mai. 2009.
- MOREIRA, T.; RIOS, E. **Teste de Software**. 2. ed. Rio de Janeiro: Altabooks, 2006.
- MÜLLER, T. et al. **Base de Conhecimento Para Certificação em Teste – Foundation Level Syllabus**. 2007. 77 f. Comissão Internacional para Qualificação de Teste de Software ISQTB.
- MYERS, G. **The Art of Software Testing**. New York: John Wiley & Sons, 1979.
- PEZZÈ, M.; YOUNG, M. **Teste e Análise de Software - processos, princípios e técnicas**. Porto Alegre: Bookman, 2008.
- PIROZZI, R. **Understanding Metrics in Software Testing**. Disponível em: <http://www.logigear.com/newsletter/understanding_metrics_in_software_testing.asp>. Acesso em: mai. 2009.
- PUSALA, R. **Operational Excellence through Efficient Software Testing Metrics**. 2006. 15 f. [S.l].
- Rational Unified Process. **RUP**. 2001. Disponível em: <<http://www.wthreex.com/rup/portugues/index.htm>>. Acesso em: mai. 2009.
- SOARES, J.; MARTINHO, L. **Métricas e Gráficos para Gestão de Testes**. 2006. 27 f. Trabalho Individual - Universidade do Porto, Faculdade de Engenharia, Portugal.
- Guide to the Software Engineering Body of Knowledge. **SWEBOK**. IEEE Computer Society. Disponível em: <<http://www.computer.org/portal/web/swebok/html/contentsch5>>. Acesso em: mai. 2009.

APÊNDICE A

Questionário Sobre Testes de Software

Obrigado por sua participação nessa pesquisa, parte de um projeto sendo desenvolvido na UFRGS, coordenado pelo Prof. Marcelo Pimenta.

O objetivo desse questionário é coletar informações sobre a inserção das atividades de verificação, validação e teste de software nas empresas brasileiras.

A comunidade brasileira - principalmente as empresas relacionadas com desenvolvimento de software - não tem hábito - e às vezes nem oportunidade - de se autoconhecer. É nossa convicção que a divulgação dos resultados desta pesquisa pode ser importante para esclarecer algumas questões sobre como o teste é entendido e praticado.

Suas informações serão confidenciais e serão utilizadas somente para fins acadêmicos.

Sua empresa e você não serão identificados em qualquer momento da pesquisa e seus dados pessoais também não serão revelados.

Se tiver qualquer dúvida em alguma das perguntas, sinta-se à vontade para pedir esclarecimentos ou fazer quaisquer comentários via nosso email de contato: ltrodo@inf.ufrgs.br.

* *Required*

1 Nome da empresa *

Nome e função do responsável pelo preenchimento do questionário *

2 Utilizam testes de software?

Sim

Não

3 Caso a empresa utilize testes de software, favor responder as questões nos subitens abaixo. Há quanto tempo a empresa utiliza teste de software?

Menos de um ano

Entre um e três anos

Mais de três anos

Os testes são feitos

Por equipe da própria empresa

Por equipe terceirizada, dentro da empresa

Por equipe terceirizada, fora da empresa

Other:

No caso da existência de uma equipe de testes, ela é formada por


- Gerente de teste
- Líder do projeto de teste
- Arquiteto de teste
- Analista de teste
- Testador
- Other:

Caso os testes sejam feitos na própria empresa, eles são feitos por quem?

- Pelos desenvolvedores
- Por uma equipe específica de teste, após os desenvolvedores terem feito os testes de códigos
- Pelos usuários
- Other:

Informar a quantidade de pessoas que formam a equipe de testes De preferência, especificar a quantidade por função

exercida



Os profissionais possuem especialização em testes de software?

- Sim. Todos que pertencem à equipe de testes
- Apenas alguns profissionais
- Não

No caso de profissionais com especialização em testes de software, se possível, indicar onde obtiveram tal especialização

- UFRGS
- PUCRS
- Unisinos
- Other:

O processo de testes é documentado?

- Sim
- Não

Em que momento são efetuados os testes no processo de desenvolvimento do sistema?

- Na definição dos requisitos
- No início da implementação
- No meio da implementação
- Na etapa final, antes de entregar o produto ao cliente
- Quando o sistema já foi entregue ao cliente
- Na inclusão de novas funcionalidades
- Em todas as etapas de desenvolvimento do sistema
- Other:

Usam ferramentas de automação de testes?

- Sim
- Não

No caso de uso de ferramentas de automação de testes, informar quais

Testes adotados (Nível de teste)

- Teste de unidade (ou teste unitário ou teste de componentes)
- Teste de integração (ou teste de interfaces)
- Teste de sistema
- Teste de aceitação
- Teste de regressão

Testes adotados (Tipos de teste, com foco nos atributos de qualidade)

- Teste de interface
- Teste de desempenho (ou teste de performance)
- Teste de carga (stress)
- Teste de usabilidade
- Teste de segurança
- Teste de funcionalidade
- Teste de tolerância a falha
- Teste de instalação

A empresa mantém um catálogo de erros (uma memória de erros)?

- Sim
- Não

Em caso positivo, o catálogo de erros é usado para auxiliar no projeto de (novos) testes?

- Sim
 Não

Usam métricas de testes?

- Sim
 Não

Em caso positivo, quais as métricas usadas?

- Índice de densidade de defeitos
 Índice de severidade de defeitos
 Efetividade de caso de teste
 Distribuição de defeitos por funcionalidade
 Defeitos encontrados X defeitos corrigidos
 Cobertura de teste
 Other:

Em caso negativo, quais as barreiras para o uso de métricas?

- Não há barreiras
 Custo
 Dificuldade no uso
 Consome muito tempo
 Não é útil ou o custo/benefício não compensa
 Não tem informações sobre recursos disponíveis
 Não tem conhecimento sobre métricas de teste de software
 Other:

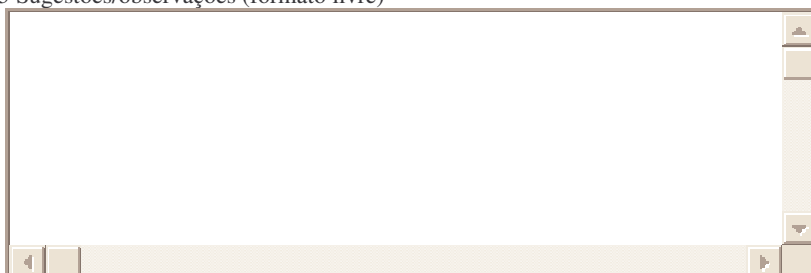
Estimativas usadas

- Métricas para medir o progresso do teste
 Métricas de projetos anteriores para estimativa do esforço do teste
 Other:

4 Caso a empresa não utilize testes de software, favor responder a questão abaixo Quais as barreiras para a implantação do processo de testes de software?

- Não há barreira
- Falta de conhecimento
- Falta de ferramentas de suporte
- Custo
- Dificuldade no uso
- Consome muito tempo
- Não é útil ou o custo/benefício não compensa
- Other:

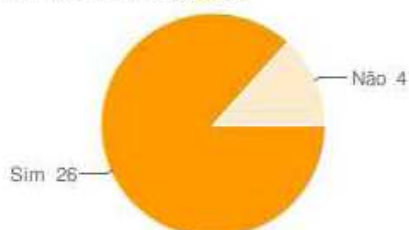
5 Sugestões/observações (formato livre)



APÊNDICE B

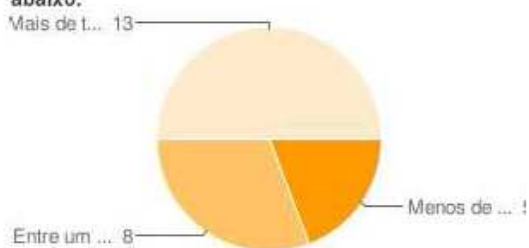
Respostas ao questionário apresentado no APÊNDICE A.

2 Utilizam testes de software?



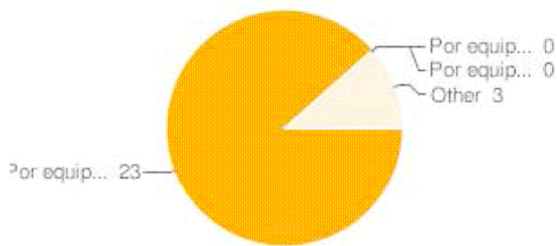
Sim	26	87%
Não	4	13%

3 Caso a empresa utilize testes de software, favor responder as questões nos subitens abaixo.



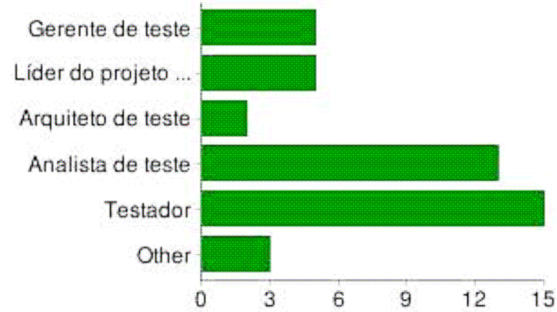
Menos de um ano	5	19%
Entre um e três anos	8	31%
Mais de três anos	13	50%

Os testes são feitos



Por equipe da própria empresa	23	88%
Por equipe terceirizada, dentro da empresa	0	0%
Por equipe terceirizada, fora da empresa	0	0%
Other	3	12%

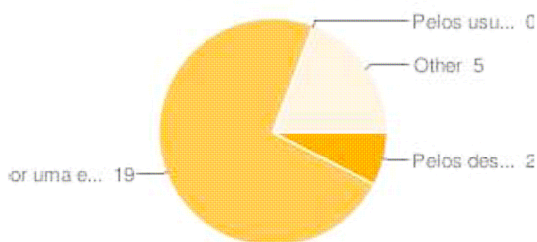
No caso da existência de uma equipe de testes, ela é formada por



Gerente de teste	5	25%
Líder do projeto de teste	5	25%
Arquiteto de teste	2	10%
Analista de teste	13	65%
Testador	15	75%
Other	3	15%

People may select more than one checkbox, so percentages may add up to more than 100%.

Caso os testes sejam feitos na própria empresa, eles são feitos por quem?



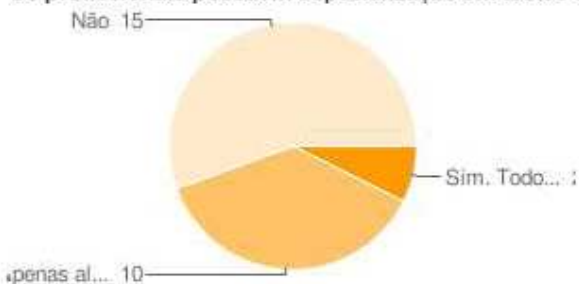
Pelos desenvolvedores	2	8%
Por uma equipe específica de teste, após os desenvolvedores terem feito os testes de códigos	19	73%
Pelos usuários	0	0%
Other	5	19%

Informar a quantidade de pessoas que formam a equipe de testes

4 testadores.2 analistas de teste. 1 gerente3 analistas6 testadores Gerente (responsável pelos testadores e desenvolvedores) - 1 Líderes de teste (QA leads) - 4 Testadores - 5 Arquitetos - 1 (que eu saiba)

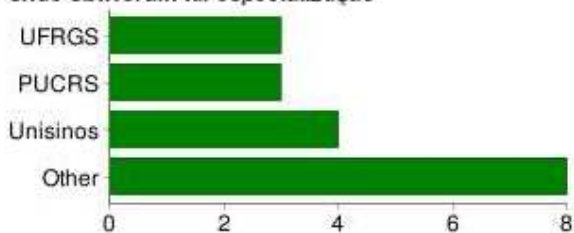
Desenvolvedores/testadores - em torno de 9 (não sei ao certo) 01 - Analista Líder 03 - Analistas/Testadores 3 pessoas um coordenador (que também faz as funções de analista e tester) um analista de testes (que também testa) um testador (que também analisa e testa) 2 Analista de testes / Testador 6 Os analistas de testes são os mesmos que executam testes Um Analista de Testes e do ...

Os profissionais possuem especialização em testes de software?



Sim. Todos que pertencem à equipe de testes	2	7%
Apenas alguns profissionais	10	37%
Não	15	56%

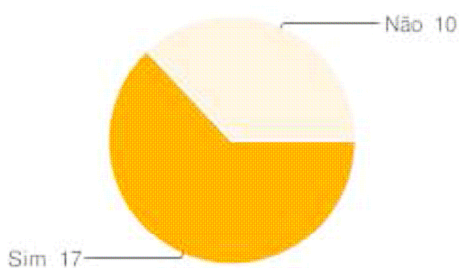
No caso de profissionais com especialização em testes de software, se possível, indicar onde obtiveram tal especialização



UFRGS	3	25%
PUCRS	3	25%
Unisinos	4	33%
Other	8	67%

People may select more than one checkbox, so percentages may add up to more than 100%.

O processo de testes é documentado?



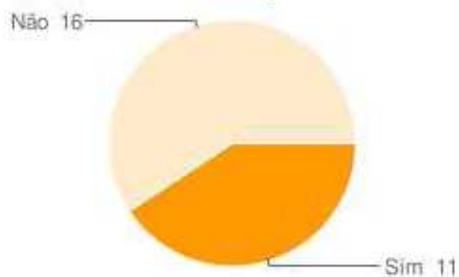
Sim	17	63%
Não	10	37%

Em que momento são efetuados os testes no processo de desenvolvimento do sistema?



People may select more than one checkbox, so percentages may add up to more than 100%.

Usam ferramentas de automação de testes?



Sim	11	41%
Não	16	59%

Testes adotados (Nível de teste)

Teste de unidade (ou teste unitário ou teste de componentes)	16	62%
Teste de integração (ou teste de	19	73%



interfaces)		
Teste de sistema	22	85%
Teste de aceitação	14	54%
Teste de regressão	10	38%

People may select more than one checkbox, so percentages may add up to more than 100%.

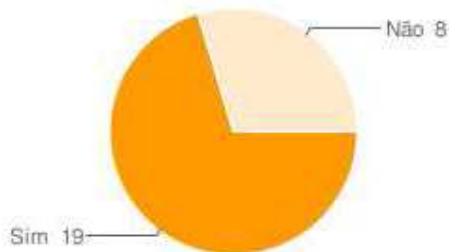
Testes adotados (Tipos de teste, com foco nos atributos de qualidade)



Teste de interface	20	77%
Teste de desempenho (ou teste de performance)	14	54%
Teste de carga (stress)	8	31%
Teste de usabilidade	17	65%
Teste de segurança	9	35%
Teste de funcionalidade	26	100%
Teste de tolerância a falha	5	19%
Teste de instalação	12	46%

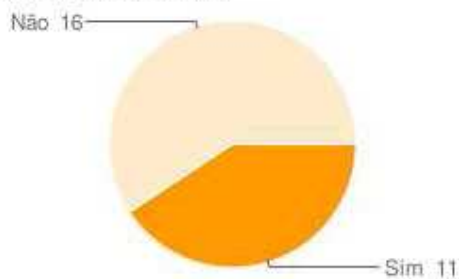
People may select more than one checkbox, so percentages may add up to more than 100%.

A empresa mantém um catálogo de erros (uma memória de erros)?



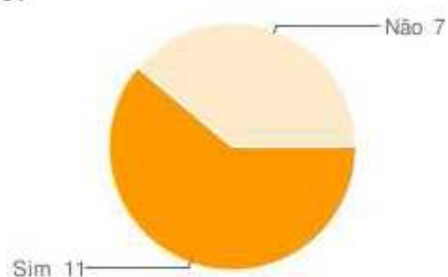
Sim	19	70%
Não	8	30%

Usam métricas de testes?



Sim	11	41%
Não	16	59%

Em caso positivo, o catálogo de erros é usado para auxiliar no projeto de (novos) testes?



Sim	11	61%
Não	7	39%

Em caso positivo, quais as métricas usadas?



Índice de densidade de defeitos	8	67%
Índice de severidade de defeitos	5	42%
Efetividade de caso de teste	6	50%
Distribuição de defeitos por funcionalidade	5	42%
Defeitos encontrados X defeitos corrigidos	9	75%
Cobertura de teste	5	42%
Other	1	8%

People may select more than one checkbox, so percentages may add up to more than 100%.

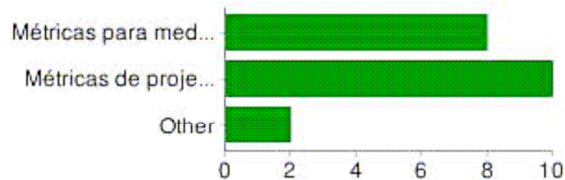
Em caso negativo, quais as barreiras para o uso de métricas?



Não há barreiras	5	29%
Custo	1	6%
Dificuldade no uso	2	12%
Consome muito tempo	5	29%
Não é útil ou o custo/benefício não compensa	1	6%
Não tem informações sobre recursos disponíveis	2	12%
Não tem conhecimento sobre métricas de teste de software	4	24%
Other	2	12%

People may select more than one checkbox, so percentages may add up to more than 100%.

Estimativas usadas



Métricas para medir o progresso do teste	8	57%
Métricas de projetos anteriores para estimativa do esforço do teste	10	71%
Other	2	14%

People may select more than one checkbox, so percentages may add up to more than 100%.

4 Caso a empresa não utilize testes de software, favor responder a questão abaixo

Não há barreira	2	33%
Falta de conhecimento	2	33%
Falta de	2	33%



ferramentas de suporte

Custo	3	50%
Dificuldade no uso	0	0%
Consome muito tempo	0	0%
Não é útil ou o custo/benefício não compensa	0	0%
Other	2	33%

People may select more than one checkbox, so percentages may add up to more than 100%.

APÊNDICE C

Métrica de teste	Referências
Número de ocorrências	(KONDA) (MOCHAL)
Status das ocorrências	(KONDA) (RUP)
Índice de Densidade de Defeitos	(RUP) (KONDA) (MAGAZINE)
Índice de severidade de defeitos	(KONDA) (RUP) (CAMACHO) (HUTCHESON) (MAGAZINE)
Tempo para arrumar um defeito	(RUP) (KONDA) (MOCHAL) (HERBERT)
Tempo para encontrar um defeito	(KONDA) (HUTCHESON)
Quantidade de falhas encontradas no produto	(HUTCHESON)
<i>Mean Time to Find a Defect</i>	(MAGAZINE)
Tipos de defeitos encontrados	(HUTCHESON) [SWEBOK] (HERBERT)
Cobertura de testes	(KONDA) (RUP) (HUTCHESON) (McGREGOR & SYKES) (SWEBOK) (MAGAZINE)
Efetividade de caso de teste	(KONDA)
Efetividade/Eficiência do teste	(HUTCHESON)
Defeitos por quantidade de linhas de código (kloc)	(KONDA)
Situação/tendência dos defeitos, em função do tempo	(RUP) (MOCHAL) (HERBERT)
Providências adotadas em relação aos defeitos	(SOARES & MARTINHO) (AMLAND)
Métricas adicionais	(SOARES & MARTINHO)
Número de casos de teste	(BRADSHAW) (MOCHAL) (BRADSHAW) (HUTCHESON) (HERBERT)
Taxa de falhas na primeira execução dos casos de teste	(BRADSHAW)
Custo dos testes	(HUTCHESON) (MOCHAL) (McGREGOR & SYKES) (HERBERT)
Curva S (<i>S-curve</i>)	(SOARES & MARTINHO) (BRADSHAW) (HUTCHESON)
The Zero Bug Bounce	(SOARES & MARTINHO)
Densidade dos defeitos residuais	(KONDA)
Relação entre defeitos e ocorrências	(KONDA)

Taxa de ocorrências válidas	(KONDA)
Taxa de problemas encontrados na correção de defeitos	(KONDA)
Defeitos Encontrados X Erros Corrigidos	(CAMACHO) (HUTCHESON)
Ocorrências pendentes de correção	(KONDA)
Defeitos encontrados X Defeitos estimados	(KONDA)
Probabilidade de defeito	(AMLAND)
Ocorrências resolvidas que ainda não foram retestadas	(KONDA)
Mudanças no escopo	(KONDA)
Fase em que o defeito foi encontrado	(KONDA)
Densidade de defeitos por unidade	(HUTCHESON) (CAMACHO)
Defeitos por fase em que foram injetados	
Tempo de teste estimado X Tempo de teste efetivamente utilizado	(CAMACHO) (KONDA) (MOCHAL)
Fator de segurança	(HUTCHESON)
Tempo necessário para executar um teste	(HUTCHESON)
Tempo disponível para o esforço de teste	(HUTCHESON)
Taxa de esforço de teste	(KONDA)
Categoria dos defeitos	(KONDA) (MOCHAL) (SWEBOK)