CAPÍTULO

TÉCNICAS DE REVISÃO

15

CONCEITOS-CHAVE

As revisões de software são como um "filtro" para a gestão de qualidade. Isso significa que as revisões são aplicadas em várias etapas durante o processo de engenharia de software e servem para revelar erros e defeitos que podem ser eliminados. As revisões de software "purificam" o resultado do trabalho da engenharia de software, até mesmo os modelos de requisitos e de projeto, dados de teste e código. Freedman e Weinberg [Fre90] discutem a necessidade de revisões da seguinte maneira:

> O trabalho técnico precisa de revisão pela mesma razão que o lápis precisa de borracha: *Errar é humano*. A segunda razão para precisarmos de revisões técnicas é que embora as pessoas sejam boas para "descobrir" alguns de seus próprios erros, vários tipos de erros escapam mais facilmente daquele que os cometeu do que de outras pessoas externas. O processo de revisão é, portanto, a resposta para a oração de Robert Burns:
>
> Oh! que uma força conceda poder
> para nos vermos como os outros nos veem
>
> Uma revisão — qualquer revisão — é uma forma de usar a diversidade de um grupo de pessoas para:
>
> 1. Apontar aperfeiçoamentos necessários no produto de uma única pessoa ou de uma equipe.
> 2. Confirmar aquelas partes de um produto em que aperfeiçoamentos são indesejáveis ou desnecessários.
> 3. Obter trabalho técnico de qualidade mais uniforme, ou pelo menos mais previsível; qualidade que possa ser alcançada sem revisões, de modo a tornar o trabalho técnico mais gerenciável.

Diversos tipos de revisão podem ser realizados como parte do processo de engenharia de software. Cada um deles tem sua função. Um encontro informal na máquina de café é

PANORAMA

O que é? À medida que desenvolvemos o trabalho de engenharia de software, cometemos erros. Não há nenhum motivo para se envergonhar disso — desde que tentemos, com muita dedicação, encontrar e corrigir os erros antes que sejam passados para os usuários finais. As revisões técnicas são o mecanismo mais efetivo para descobrir erros logo no início da gestão de qualidade.

Quem realiza? Os engenheiros de software realizam as revisões técnicas, também chamadas revisões paritárias, juntamente com seus colegas.

Por que é importante? Ao se descobrir um erro logo no início do processo, fica menos caro corrigi-lo. Além disso, os erros podem aumentar à medida que o processo continua. Portanto, um erro relativamente insignificante, sem tratamento no início do processo, pode ser amplificado e se transformar em um conjunto de erros graves para a sequência do projeto. Finalmente, as revisões minimizam o tempo devido à redução do número de reformulações que serão necessárias ao longo do projeto.

Quais são as etapas envolvidas? A abordagem em relação às revisões irá variar dependendo do grau de formalidade escolhido. Em geral, são empregadas seis etapas, embora nem todas sejam usadas para todo tipo de revisão: planejamento, preparação, estruturação da reunião, anotação de erros, realização das correções (feita fora da revisão) e verificação se as correções foram feitas apropriadamente.

Qual é o artefato? O artefato de uma revisão é uma lista de problemas e/ou erros que foram descobertos. Além disso, também é indicado o estado técnico do produto resultante.

Como garantir que o trabalho foi realizado corretamente? Primeiramente, escolha o tipo de revisão apropriado para o seu ambiente de desenvolvimento. Siga as diretrizes que levam a revisões bem-sucedidas. Se as revisões realizadas conduzirem a software de maior qualidade, elas foram feitas corretamente.

uma forma de revisão, caso sejam discutidos problemas técnicos. Uma apresentação formal da arquitetura de software para um público formado por clientes, pessoal técnico e gerencial também é uma forma de revisão. Neste livro, entretanto, focalizaremos as *revisões técnicas ou paritárias,* exemplificadas por *revisões informais, walkthroughs e inspeções.* Uma revisão técnica, RT, é o filtro mais efetivo do ponto de vista de controle da qualidade. Conduzida por engenheiros de software (e outros) para engenheiros de software, a RT é um meio eficaz para revelar erros e aumentar a qualidade do software.

15.1 Impacto de Defeitos de Software nos Custos

Revisões são como um filtro no fluxo de trabalho da gestão de qualidade. Um número muito pequeno de revisões e o fluxo será "sujo". Um número muito grande de revisões e o fluxo diminui muito e vira um gotejamento. Use métricas para determinar quais revisões funcionam e as enfatize. Elimine do fluxo as revisões ineficazes para acelerar o processo.

No contexto da gestão de qualidade, os termos *defeito* e *falha* são sinônimos. Ambos implicam um problema de qualidade que é descoberto *depois* de o software ter sido liberado para os usuários finais (ou para uma outra atividade estrutural dentro da gestão de qualidade). Em capítulos anteriores, usamos o termo *erro* para indicar um problema de qualidade que é descoberto por engenheiros de software (ou outros) *antes* de o software ser liberado ao usuário final (ou para outra atividade estrutural dentro da gestão de qualidade).

O principal objetivo das revisões técnicas é encontrar erros durante o processo, de modo a não se tornarem defeitos depois da liberação do software. O benefício evidente das revisões técnicas é a descoberta precoce de erros, de modo que eles não sejam propagados para a próxima etapa na gestão de qualidade.

Diversos estudos e análise sobre o tema indicam que as atividades de projeto introduzem de 50 a 65% de todos os erros (e, em última instância, todos os defeitos), durante a gestão de qualidade. Entretanto, técnicas de revisão demonstraram ser até 75% eficazes [Jon86] na descoberta de falhas de projeto. Detectando e eliminando um grande percentual desses erros, o processo de revisão reduz substancialmente o custo das atividades subsequentes na gestão de qualidade.

INFORMAÇÕES

Bugs, erros e defeitos

O objetivo do controle da qualidade de software e da gestão da qualidade em geral é, em sentido mais amplo, eliminar problemas de qualidade no software. Tais problemas são conhecidos por diversos nomes — *bugs*, falhas, erros ou defeitos, apenas para citar alguns. Esses termos são sinônimos ou existem diferenças sutis entre eles?

Neste livro é feita uma distinção clara entre erro (um problema de qualidade encontrado antes de o software ser liberado aos usuários finais) e *defeito* (um problema de qualidade encontrado apenas *depois* de o software ter sido liberado aos usuários finais[1]). Essa distinção é feita porque os erros e os defeitos podem acarretar impactos econômicos, comerciais, psicológicos e humanos muito diferentes. Os engenheiros de software têm a missão de encontrar e corrigir o maior número possível de erros antes dos clientes e/ou usuários finais. Devem-se evitar defeitos — pois (de modo justificável) criam uma imagem negativa do pessoal de software.

É importante notar, entretanto, que a distinção temporal entre erros e defeitos feita neste livro não é um pensamento dominante. O consenso geral na comunidade de engenharia de software é que defeitos e erros, falhas e *bugs* são sinônimos. Ou seja, o momento em que o problema foi encontrado não tem nenhuma influência no termo usado para descrevê-lo. Parte do argumento a favor desta visão é que, muitas vezes, fica difícil fazer uma distinção clara entre pré e pós-entrega (consideremos, por exemplo, um processo incremental usado no desenvolvimento ágil).

Independentemente da maneira escolhida para interpretar esses termos ou do momento em que um problema é descoberto, o que importa efetivamente é que os engenheiros de software devem se esforçar — *muitíssimo* — para encontrar problemas antes que seus clientes e usuários finais os façam. Caso tenha maior interesse nessa questão, uma discussão razoavelmente completa sobre a terminologia envolvendo *bugs* pode ser encontrada em **www.softwaredevelopment.ca/bugs.shtml.**

1 Se considerarmos o aperfeiçoamento na gestão de qualidade, um problema de qualidade que se propaga de uma atividade estrutural do processo (por exemplo, **modelagem**) para outra (por exemplo, **construção**) também pode ser chamado de "defeito", pois o problema deveria ter sido descoberto antes de um produto resultante (por exemplo, um modelo de projeto) ter sido "liberado" para a atividade seguinte.

15.2 AMPLIFICAÇÃO E ELIMINAÇÃO DE DEFEITOS

"Algumas enfermidades, como dizem os médicos, são fáceis de curar logo no início, mas difíceis de diagnosticar... Com o tempo, quando elas não foram diagnosticadas e tratadas no início, tornam-se de fácil diagnóstico, mas difícil cura."

Nicolau Maquiavel

Um modelo de amplificação de *defeitos* [IBM81] pode ser usado para ilustrar a geração e a detecção de erros durante o projeto e nas ações para geração de código de uma gestão de qualidade. O modelo é ilustrado esquematicamente na Figura 15.1. Um retângulo representa uma ação de engenharia de software. Durante a ação, os erros podem ser gerados inadvertidamente. Pode ser que a revisão falhe na descoberta de erros recentes e erros de etapas anteriores, resultando em uma série de erros que são passados para a etapa seguinte. Em alguns casos, esses erros que são passados e provenientes de etapas anteriores são amplificados (fator de amplificação *x*) pelo trabalho atual. As subdivisões dos retângulos representam cada uma dessas características e o percentual de eficiência para a detecção de erros, uma função da meticulosidade da revisão.

A Figura 15.2 ilustra um exemplo hipotético da amplificação de defeitos para uma gestão de qualidade em que não foi realizada nenhuma revisão. De acordo com a figura, supõe-se que cada etapa de teste revele e corrija 50% de todos os erros de entrada sem introduzir nenhum erro novo (uma hipótese otimista). Dez defeitos preliminares de projeto são amplificados para 94 erros antes do início do teste. Vinte erros (defeitos) latentes são liberados para o campo. A Figura 15.3 considera as mesmas condições, exceto que as revisões de projeto e código são realizadas como parte de cada ação de engenharia de software. Nesse caso, dez erros de projeto preliminares (de arquitetura) iniciais são amplificados para 24 erros antes do início dos testes. Existem apenas três erros latentes. Podem ser estabelecidos os custos relativos associados à descoberta e à correção dos erros, bem como o custo total (com e sem revisão, para nosso exemplo hipotético). O número de erros revelados durante cada uma das etapas indicadas nas Figuras 15.2 e 15.3 é multiplicado pelo custo para eliminar um erro (1,5 unidades de custo para projeto, 6,5 unidades de custo antes do teste, 15 unidades de custo durante o teste e 67 uni-

FIGURA 15.1
Modelo de amplificação de defeitos

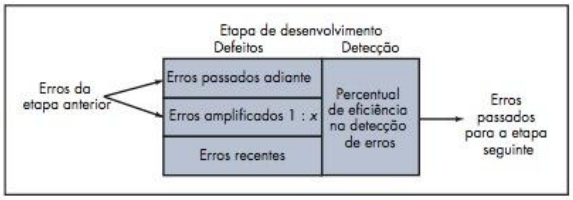


FIGURA 15.2
Amplificação de defeitos – sem revisão

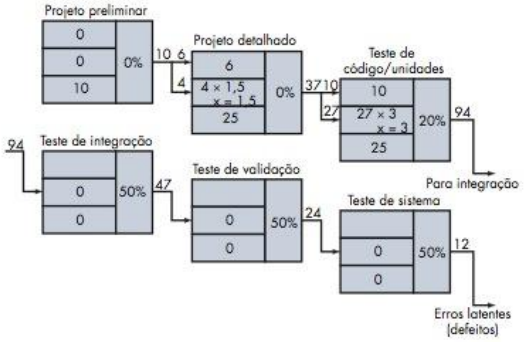
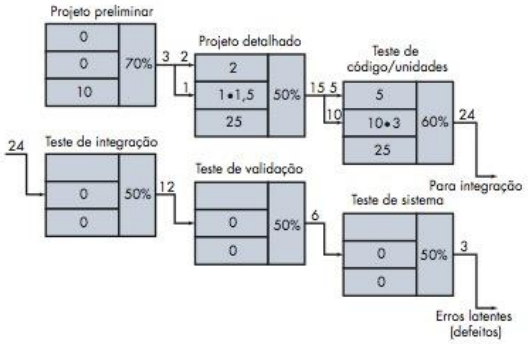


FIGURA 15.3
Amplificação de defeitos – revisões realizadas



dades de custo depois da entrega).[2] Usando esses dados, o custo total para desenvolvimento e manutenção quando são realizadas revisões é de 783 unidades de custo. Quando não é feita nenhuma revisão, o custo total é de 2.177 unidades — aproximadamente três vezes mais caro.

Para realizarmos revisões, temos de despender tempo e esforço, e a empresa de desenvolvimento tem de disponibilizar recursos financeiros. Entretanto, os resultados do exemplo anterior não deixam nenhuma dúvida de que podemos pagar agora ou então pagar muito mais no futuro.

15.3 MÉTRICAS DE REVISÃO E SEU EMPREGO

As revisões técnicas são algumas das muitas ações exigidas como parte de uma prática de engenharia de software adequada. Cada ação requer dedicação de nossa parte. Já que o esforço disponível para o projeto é finito, é importante para uma organização de engenharia de software compreender a eficácia de cada ação definindo um conjunto de métricas (Capítulo 23) que podem ser usadas para avaliar sua eficácia.

Embora possam ser definidas muitas métricas para as revisões técnicas, um subconjunto relativamente pequeno pode nos dar uma boa ideia da situação. As métricas a seguir podem ser reunidas para cada revisão a ser realizada:

- *Esforço de preparação, Ep* — o esforço (em homens/hora) exigido para revisar um produto resultante antes da reunião de revisão.
- *Esforço de avaliação, Ea* — o esforço (em homens/hora) que é despendido durante a revisão em si.
- *Reformulação esforço, Re* — o esforço (em homens/hora) dedicado à correção dos erros revelados durante a revisão.
- *Tamanho do artefato de software, TPS* — uma medida do tamanho do artefato de software que foi revisto (por exemplo, o número de modelos UML ou o número de páginas de documento ou então o número de linhas de código).
- *Erros secundários encontrados, Errsec* — o número de erros encontrados que podem ser classificados como secundários (exigindo menos para ser corrigidos do que algum esforço pré-especificado).

2 Esses multiplicadores são ligeiramente diferentes dos dados apresentados na Figura 14.2, que são mais comuns. Entretanto, servem para ilustrar bem a amplificação dos custos de defeitos.

- *Erros graves encontrados, Errgraves* — o número de erros encontrados que podem ser classificados como graves (exigindo mais para ser corrigidos do que algum esforço pré-especificado).

Essas métricas podem ser ainda mais refinadas associando-se o tipo de artefato de software que foi revisto para as métricas.

15.3.1 Análise de métricas

Antes de a análise poder ser iniciada, devem ser realizados alguns cálculos simples. O esforço total de revisão e o número total de erros descobertos são definidos como:

$$E_{\text{revisão}} = E_p + E_a + R_e$$
$$\text{Err}_{\text{tot}} = \text{Err}_{\text{sec}} + \text{Err}_{\text{graves}}$$

A densidade de erros representa os erros encontrados por unidade do artefato de software revisto.

$$\text{Densidade de erros} = \frac{\text{Err}_{\text{tot}}}{\text{TPS}}$$

Por exemplo, se um modelo de requisitos for revisado para revelar erros, inconsistências e omissões, seria possível calcular a densidade de erros de várias formas diferentes. O modelo de requisitos contém 18 diagramas UML como parte de um total de 32 páginas de material descritivo. A revisão revela 18 erros secundários e 4 erros graves. Consequentemente, $\text{Err}_{\text{tot}} = 22$. A densidade de erros é de 1,2 erros por diagrama UML ou 0,68 erros por página de modelo de requisitos.

Se as revisões forem realizadas para uma série de tipos de artefatos resultantes diferentes (por exemplo, modelo de necessidades, modelo de projeto, código e casos de teste), a porcentagem de erros revelados para cada revisão pode ser calculada em relação ao número total de erros encontrados para todas as revisões. Além disso, podemos calcular também a densidade de erros para cada artefato.

Assim que forem coletados dados de várias revisões realizadas durante vários projetos, os valores médios da densidade de erros permitem que estimemos o número de erros a ser encontrado em um novo documento, mesmo que não seja revisado. Se, por exemplo, a densidade média de erros para um modelo de requisitos for 0,6 erros por página e um novo modelo de requisitos tiver 32 páginas, uma estimativa grosseira sugeriria que a equipe de software irá encontrar aproximadamente 19 ou 20 erros durante a revisão do documento. Se encontrarmos apenas 6 erros, fizemos um trabalho extremamente bom no desenvolvimento do modelo de requisitos ou então a abordagem de revisão não foi suficientemente completa.

Assim que forem realizados os testes (Capítulos 17 a 20), é possível reunir dados de erros adicionais, incluindo o esforço necessário para encontrar e corrigir erros revelados durante os testes e a densidade de erros do software. Os custos associados à descoberta e à correção de um erro durante um teste podem ser comparados com aqueles obtidos nas revisões. Isso é discutido na Seção 15.3.2.

15.3.2 Eficácia dos custos de revisões

É difícil medir, em tempo real, a eficácia dos custos de qualquer revisão técnica. Uma organização de engenharia de software pode avaliar a eficácia das revisões e seu custo-benefício apenas após terem sido completadas as revisões, reunidas as métricas de revisão e calculados os dados médios. E finalmente, a partir desse ponto, medir a qualidade do software (por meio de testes).

Voltando ao exemplo apresentado na Seção 15.3.1, determinou-se que a densidade média de erros para os modelos de requisitos foi de 0,6 erros por página. Verificou-se que o esforço necessário para corrigir um erro secundário do modelo (imediatamente após a revisão) exige 4 homens/hora. Verificou-se que o esforço necessário para corrigir um erro grave era de 18 homens/ hora.

Examinando-se os dados coletados na revisão, determina-se que os erros secundários ocorrem com uma frequência aproximada 6 vezes maior que a de erros graves. Consequentemente, podemos estimar que o esforço médio para encontrar e corrigir um erro de requisitos durante uma revisão é de cerca de 6 homens/hora.

Os erros relacionados aos requisitos, revelados durante os testes, exigem uma média de 45 homens/hora para ser encontrados e corrigidos (não há dados disponíveis sobre a gravidade relativa do erro). Usando as médias observadas, obtemos:

$$\text{Esforço poupado por erro} = \text{Err}_{\text{testes}} - \text{Err}_{\text{revisões}}$$
$$45 - 6 = 30 \text{ homens/hora/erro}$$

Como foram encontrados 22 erros durante a revisão do modelo de requisitos, obteríamos uma economia aproximada de 660 homens/hora de esforço de testes. E essa economia refere-se apenas aos erros relacionados ao modelo de requisitos. Os erros associados a projeto e codificação aumentariam ainda mais o benefício geral. Em suma — o esforço poupado nos leva a ciclos de entrega mais curtos e a menor tempo de colocação do produto no mercado.

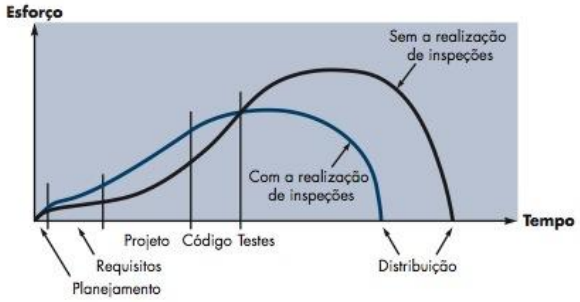
Em seu livro sobre revisões paritárias, Karl Wiegers [Wie02] discute dados incidentais obtidos de grandes empresas que usaram *inspeções* (um tipo de revisão técnica relativamente formal) como parte de suas atividades de controle da qualidade de software. A Hewlett Packard relatou um retorno sobre o investimento de 10:1 para inspeções e citou que a entrega real do produto foi acelerada, em média, 1,8 mês. A AT&T indicou que as inspeções reduziram o custo total de erros de software por um fator de 10, a qualidade foi incrementada em um grau de magnitude e a produtividade aumentou 14%. Outros relatos informam benefícios semelhantes. As revisões técnicas (para projeto e outras atividades técnicas) fornecem uma relação custo-benefício demonstrável e efetivamente economizam tempo.

Porém, para muitas pessoas da área de software, essa afirmação é contraintuitiva. "As revisões tomam tempo", argumentam os profissionais de software "não temos muito tempo!". Eles defendem que tempo é um bem precioso em qualquer projeto de software e a habilidade de revisar "detalhadamente todos os artefatos resultantes" absorve muito tempo.

Os exemplos apresentados anteriormente nesta seção indicam o contrário. Mais importante ainda, foram coletados dados do setor para revisões de software por mais de duas décadas e estão sintetizados de forma qualitativa nos gráficos da Figura 15.4.

Em relação à figura, o esforço despendido quando revisões são empregadas aumenta efetivamente no início do desenvolvimento de um módulo de software, mas esse investimento inicial para revisões rende dividendos, pois o esforço de testes e correções é reduzido. Igualmente importante, a data de entrega/distribuição para o desenvolvimento com revisões é anterior àquela sem o uso de revisões. As revisões não gastam tempo, elas poupam!

FIGURA 15.4
Esforço despendido com e sem o emprego de revisões
***Fonte:* Adaptado de [Fag86]**



15.4 REVISÕES: UM ESPECTRO DE FORMALIDADE

As revisões técnicas devem ser aplicadas com um nível de formalidade apropriado para o produto a ser construído, a cronologia do projeto e as pessoas que estão realizando o trabalho. A Figura 15.5 representa um modelo de referência para revisões técnicas [Lai02] que identifica quatro características que contribuem para a formalidade na qual uma revisão é conduzida.

Cada uma das características do modelo de referência ajuda a definir o nível da formalidade da revisão. A formalidade de uma revisão aumenta quando: (1) são definidos explicitamente os papéis distintos para os revisores, (2) há um nível suficiente de planejamento e preparação para a revisão, (3) é definida uma estrutura distinta para a revisão (incluindo tarefas e artefatos internos) e (4) ocorre o follow-up pelos revisores para qualquer correção realizada.

Para compreender o modelo de referência, suponhamos que tenhamos decidido revisar o projeto da interface para o **CasaSeguraGarantida.com**. Isso pode ser feito de várias formas diferentes, que vão desde relativamente informal a extremamente rigorosa. Caso ache que a abordagem informal é a mais apropriada, solicite a alguns colegas (pares) para examinarem o protótipo da interface em uma tentativa de descobrir problemas potenciais. Todos os participantes decidem que não haverá nenhuma preparação prévia, mas que você irá avaliar o protótipo de uma forma razoavelmente estruturada — verificando primeiramente o layout, em seguida a estética, depois as opções de navegação e assim por diante. Na qualidade de projetista, você decide fazer algumas anotações, mas nada formal.

Mas o que acontece se a interface for fundamental para o sucesso do projeto inteiro? E se vidas humanas dependerem de uma interface ergonomicamente consistente? Será necessária uma abordagem mais rigorosa. Deverá ser formada uma equipe de revisão. Cada membro da equipe deve ter um papel específico a ser desempenhado — liderar a equipe, registrar as descobertas, apresentar o material e assim por diante. Cada revisor deve ter acesso ao artefato de software (neste caso, o protótipo da interface) antes da revisão e gastar tempo procurando erros, inconsistências e omissões. Um conjunto de tarefas específicas necessita ser conduzido, tomando como base uma agenda desenvolvida antes de a revisão ocorrer. Os resultados da revisão precisam ser registrados formalmente, e a equipe deve decidir sobre a situação do artefato de software com base na revisão. Os membros da equipe de revisão também podem verificar se as correções foram feitas de forma apropriada.

Neste livro, consideramos duas grandes categorias de revisões técnicas: revisões informais e revisões técnicas mais formais. Dentro de cada categoria, podem ser escolhidas várias abordagens distintas que são apresentadas nas seções seguintes.

FIGURA 15.5
Modelo de referência para revisões técnicas

15.5 REVISÕES INFORMAIS

Entre as revisões informais temos um simples teste de mesa de um artefato de engenharia de software (com um colega), uma reunião informal (envolvendo mais de duas pessoas) com a finalidade de revisar um artefato, ou os aspectos orientados a revisões da programação em pares (Capítulo 3).

Um *simples teste de mesa* ou uma *reunião informal* realizada com um colega é uma revisão. Entretanto, pelo fato de não haver nenhum planejamento ou preparação antecipados, nenhuma agenda ou estrutura de reuniões e nenhum follow-up sobre os erros encontrados, a eficácia de tais revisões é consideravelmente menor que as abordagens mais formais. Mas um simples teste de mesa pode realmente revelar erros que, de outra forma, poderiam se propagar ainda mais na gestão de qualidade.

Uma forma de aumentar a eficácia de uma revisão do tipo teste de mesa é desenvolver um conjunto de listas de verificação simples para cada artefato produzido pela equipe de software. As questões levantadas na lista de verificação são genéricas, mas servirão como guia para os revisores verificarem o produto resultante. Por exemplo, vamos reexaminar um teste de mesa do protótipo da interface para **CasaSeguraGarantida.com**. Em vez de simplesmente ficar testando o protótipo na estação de trabalho do projetista, o projetista e um colega examinam o protótipo usando uma lista de verificação para interfaces:

O layout é projetado usando convenções padronizadas? Da esquerda para a direita? De cima para baixo?

- A apresentação precisa de barra de rolagem?
- A cor e o posicionamento, o tipo e o tamanho dos elementos são usados efetivamente?
- Todas as opções de navegação ou funções representadas estão no mesmo nível de abstração?
- Todas as opções de navegação são claramente identificadas?

e assim por diante. Quaisquer erros ou problemas verificados pelos revisores são registrados pelo projetista para resolução mais tarde. Poderiam ser programados testes de mesa de forma *ad hoc* ou eles seriam compulsórios, como parte da boa prática de engenharia de software. Em geral, a quantidade de material a ser revisada é relativamente pequena e o tempo total gasto em um teste de mesa vai um pouco além de uma ou duas horas.

No Capítulo 3, descrevemos a *programação em pares* da seguinte maneira: "XP (eXtreme Programming) recomenda que duas pessoas trabalhem juntas em uma mesma estação de trabalho para criar código. Isso disponibiliza um mecanismo para a resolução de problemas em tempo real (duas cabeças normalmente funcionam melhor do que uma) e a garantia da qualidade em tempo real".

A programação em pares pode ser caracterizada como um teste de mesa contínuo. Em vez de programar uma revisão em algum momento, a programação em pares encoraja a revisão contínua enquanto um artefato de software (projeto ou código) é criado. O benefício é a descoberta imediata de erros e, consequentemente, maior qualidade do artefato final.

Em sua discussão sobre a eficácia da programação em pares, Williams e Kessler [Wil00] afirmam:

> Tanto evidências incidentais quanto evidências estatísticas iniciais indicam que a programação em pares é uma técnica poderosa para gerar, de forma produtiva, produtos de software de alta qualidade. O par trabalha e compartilha ideias juntos para abordar as complexidades do desenvolvimento de software. Eles realizam inspeções continuamente em cada um dos artefatos do outro, levando à forma mais precoce e eficiente possível de eliminação de defeitos. Além disso, cada um faz o outro se manter atentamente focado na tarefa em questão.

Alguns engenheiros de software sustentam que a redundância inerente da programação em pares é um desperdício de recursos. Afinal de contas, por que alocar duas pessoas para um

trabalho que uma única é capaz de realizar? A resposta a essa pergunta pode ser encontrada na Seção 15.3.2. Se a qualidade do produto resultante da programação em pares for significativamente melhor que o trabalho individual, as economias relacionadas com qualidade são plenamente capazes de justificar a "redundância" implícita na programação em pares.

INFORMAÇÕES

Listas de verificação para revisões

Mesmo quando as revisões são bem organizadas e apropriadamente conduzidas, não é uma má ideia munir os revisores de um recurso a mais. Vale a pena ter uma lista de verificação que forneça a cada revisor as perguntas que deveriam ser feitas sobre o artefato específico que está passando por revisão.

Um dos conjuntos mais completos de listas de verificação para revisões foi desenvolvido pela NASA no Goddard Space Flight Center e se encontra disponível em **http://sw-assurance.gsfc.nasa.gov/disciplines/quality/index.php**

Outras listas de verificação para revisões técnicas muito úteis também foram propostas por:

Process Impact (**www.processimpact.com/pr_goodies.shtml**)

Software Dioxide (**www.softwaredioxide.com/channels/con.View.asp?id=6309**)

Macadamian (**www.macadamian.com**)

The Open Group Architecture Review Checklist (**www.opengroup.org/architecture/togaf7-doc/arch/p4/comp/clists/syseng.htm**)

DFAS [downloadable] (**www.dfas.mil/ technology/pal/ssps/docstds/spm036.doc**)

15.6 REVISÕES TÉCNICAS FORMAIS

"Não há impulso maior do que aquele de um homem revisar o trabalho de outro homem."

Mark Twain

Revisão técnica formal ou *RTF* é uma atividade de controle da qualidade de software realizada por engenheiros de software (e outros profissionais). Os objetivos de uma RTF são: (1) descobrir erros na função, lógica ou implementação para qualquer representação do software; (2) verificar se o software que está sendo revisado atende aos requisitos; (3) garantir que o software foi representado de acordo com padrões predefinidos; (4) obter software que seja desenvolvido de maneira uniforme; e (5) tornar os projetos mais gerenciáveis. Além disso, a RTF serve como base de treinamento, possibilitando que engenheiros mais novos observem diferentes abordagens para análise, projeto e implementação de software.

A RTF também serve para promover backup e continuidade, pois muitas pessoas se familiarizam com partes do software que de outra maneira jamais teriam visto.

A RTF é, atualmente, uma classe de revisões que inclui *walkthroughs* e *inspeções*. Cada RTFé realizada como uma reunião e apenas será bem-sucedida se for apropriadamente planejada, controlada e tiver a participação de todos os envolvidos. Nas seções a seguir são apresentadas orientações similares àquelas para um walkthrough na forma de uma revisão técnica formal representativa. Caso tenha interesse em inspeções de software, bem como queira informações adicionais sobre walkthroughs, ver [Rad02], [Wie02] ou [Fre90].

15.6.1 Uma reunião de revisão

WebRef

O NASA SATC *Formal Inspection Guidebook* pode ser copiado de **satc.gsfc.nasa.gov/Documents/fi/gdb/fi.pdf**.

Independentemente do formato de RTF escolhido, cada reunião de revisão deve observar as seguintes restrições:

- Devem estar envolvidas de três a cinco pessoas em uma revisão (tipicamente).
- Deve ocorrer uma preparação antecipada, porém não deve tomar mais do que duas horas de trabalho de cada pessoa.
- A duração da reunião de revisão deve ser de menos de duas horas.

Dadas essas restrições, deve ser óbvio que uma RTF se concentre em uma parte específica (e pequena) do software. Por exemplo, em vez de tentar revisar um projeto inteiro, os walkthroughs são realizados para cada componente ou para um pequeno grupo de componentes. Afunilando-se o foco, a RTF terá maior probabilidade de revelar erros.

PONTO-CHAVE

Uma RTF concentra-se em uma parte relativamente pequena de um artefato.

O foco da RTF é um artefato resultante (por exemplo, parte de um modelo de requisitos, o projeto detalhado de um componente, o código-fonte de um componente). O indivíduo que desenvolveu o artefato — o *produtor* — informa ao líder de projeto do artefato está completo e que é necessário fazer uma revisão. O líder de projeto contata um *líder de revisão,* que avalia o artefato em termos de completude, gera cópias dos materiais resultantes e as distribui para dois ou três *revisores* para preparação prévia. Espera-se que cada revisor gaste de uma a duas horas revisando o artefato, tomando notas e, de alguma forma, se familiarizando com o trabalho realizado. Ao mesmo tempo, o líder da reunião de revisão também revisa o artefato e estabelece uma agenda para a reunião de revisão, que normalmente é marcada para o dia seguinte.

Em algumas situações, é uma boa ideia fazer com que outra pessoa que não seja o produtor analise o produto que está sendo submetido a uma revisão. Isso leva a uma interpretação literal do artefato e a um melhor diagnóstico dos erros.

Uma reunião de revisão tem a participação de um líder de revisão, todos os revisores e o produtor. Um dos revisores assume o papel de *registrador,* isto é, o indivíduo que registra (por escrito) todas as questões importantes surgidas durante a revisão. A RTF começa com uma introdução da agenda e uma breve introdução por parte do produtor, que continua na descrição do artefato resultante, explicando o material, enquanto os revisores levantam questões com base em sua preparação prévia. Quando são descobertos problemas ou erros válidos, o registrador toma nota de cada um deles.

No final da revisão, todos os participantes da RTF devem decidir se: (1) aceitam o artefato sem as modificações adicionais, (2) rejeitam o artefato devido a erros graves (uma vez corrigidos, deve ser realizada outra revisão) ou (3) aceitam o artefato provisoriamente (foram encontrados erros secundários que devem ser corrigidos, mas não haverá nenhuma outra revisão). Após uma tomada de decisão, todos os participantes da RTF assinam um documento de aprovação, indicando sua participação na revisão e sua concordância com as descobertas da equipe de revisão.

15.6.2 Relatório de revisão e manutenção de registros

Durante a RTF, um revisor (o registrador) registra ativamente todos os problemas levantados. Estes são sintetizados no final da reunião de revisão e é produzida uma *lista de problemas* de revisão. Além disso, um *relatório sintetizado da revisão técnica formal* é completado. O relatório sintetizado de revisão deve responder a três questões:

1. O que foi revisado?
2. Quem o revisou?
3. Quais foram as descobertas e as conclusões?

Um relatório sintetizado da revisão é um formulário de uma página (com possíveis anexos). Ele torna-se um registro histórico do projeto e pode ser distribuído ao líder do projeto e a outras partes interessadas.

A lista de problemas de revisão atende a dois propósitos: (1) identificar áreas problemáticas no artefato e (2) servir como uma lista de verificação de itens de ação que orienta o produtor à medida que são feitas as correções. Normalmente é anexada uma lista de problemas ao relatório sintetizado.

Devemos estabelecer um procedimento de acompanhamento para garantir que itens contidos na lista de problemas tenham sido corrigidos apropriadamente. Caso isso não seja feito, é possível que problemas levantados possam "ficar para trás". Uma das abordagens é atribuir a responsabilidade pelo acompanhamento (follow-up) ao líder da revisão.

Não aponte erros de forma áspera. Uma maneira de ser gentil é fazer uma pergunta que possibilite ao produtor descobrir o próprio erro.

15.6.3 Diretrizes de revisão

Devem-se estabelecer previamente diretrizes para a realização de revisões técnicas formais, distribuídas a todos os revisores, ter a anuência de todos e, então, segui-las à risca. Uma revisão não controlada muitas vezes pode ser pior do que não fazer nenhuma revisão. A seguir, apresentamos um conjunto mínimo de diretrizes para revisões técnicas formais:

1. *Revisar o produto, não o produtor.* Uma RTF envolve pessoas e egos. Conduzida de forma apropriada, a RTF deve deixar todos os seus participantes com uma agradável sensação de dever cumprido. Conduzido de forma imprópria, a RTF pode assumir a aura de uma inquisição. Os erros devem ser apontados gentilmente; o clima da reunião deve ser descontraído e construtivo; o intuito não deve ser o de causar embaraços ou menosprezo. O líder da revisão deve conduzir a reunião de revisão de tal forma a garantir que o clima seja mantido e as atitudes sejam apropriadas; além disso, deve interromper imediatamente uma revisão que começou a sair do controle.

"Uma reunião é, muitas vezes, um evento em que minutos são perdidos e horas são desperdiçadas."

Autor desconhecido

2. *Estabelecer uma agenda e mantê-la.* Um dos principais males de reuniões de todos os tipos é desviar do foco. Uma RTF deve ser mantida em seu rumo e prazo estabelecidos. O líder da revisão tem a responsabilidade de manter o cronograma da reunião e não deverá ficar receoso em alertar as pessoas quando ela estiver saindo do foco.
3. *Limitar debates e refutação.* Quando uma questão é levantada por um revisor, talvez não haja um acordo universal sobre seu impacto. Em vez de perder tempo debatendo a questão, o problema deve ser registrado para posterior discussão, fora da reunião. 4. *Enunciar as áreas do problema, mas não tentar resolver todo problema registrado.* Uma revisão não é uma sessão para resolução de problemas. A solução de um problema pode, muitas vezes, ser realizada pelo próprio produtor ou com a ajuda de apenas outro colega. A resolução de problemas deve ser adiada para depois da reunião de revisão.
4. *Enunciar as áreas do problema mas não tentar resolver todo o problema registrado.* Uma visão não é uma sessão para resolução de problemas. A solução de um problema pode, muitas vezes, ser realizada pelo próprio produtor ou com a ajuda de apenas outro colega. A resolução de problemas deve ser adiada para depois da reunião de revisão.
5. *Tomar notas.* Algumas vezes é uma boa ideia para o registrador fazer apontamentos em um quadro, de modo que os termos e as prioridades possam ser avaliados por outros revisores à medida que as informações são registradas. Alternativamente, as anotações podem ser introduzidas diretamente em um notebook.
6. *Limitar o número de participantes e insistir na preparação antecipada.* Duas cabeças funcionam melhor do que uma, mas catorze cabeças não funcionam, necessariamente, melhor do que quatro. Mantenha o número de pessoas envolvidas no mínimo necessário. Entretanto, todos os membros da equipe de revisão devem se preparar com antecedência. O líder da revisão deve solicitar comentários escritos (fornecendo uma indicação de que o revisor reviu o material).
7. *Desenvolver uma lista de verificação para cada artefato que provavelmente será revisado.* A lista de verificação ajuda o líder da revisão a estruturar a RTF e auxilia cada revisor a se concentrar nas questões importantes. As listas de verificação devem ser desenvolvidas para análise, projeto, código e até mesmo para o teste dos artefatos.

"É uma das mais belas recompensas da vida, que nenhum homem pode, sinceramente, tentar ajudar outro sem ajudar a si mesmo."

Ralph Waldo Emerson

8. *Alocar os recursos e programar o tempo para as RTFs.* Para as revisões serem eficazes, elas devem ser programadas como tarefas durante a gestão de qualidade. Além disso, deve-se programar o tempo para as inevitáveis modificações que ocorrerão como resultado de uma RTF.
9. *Realizar treinamento significativo para todos os revisores.* Para serem eficazes, todos os participantes de uma revisão deveriam receber algum treinamento formal. O treinamento deve enfatizar tanto questões relacionadas a processos, como o lado psicológico das revisões. Freedman e Weinberg [Fre90] estimam uma curva de aprendizado para cada vinte pessoas que participarão efetivamente de revisões.
10. *Revisar revisões iniciais.* Um interrogatório pode ser benéfico na descoberta de problemas com o próprio processo de revisão. Os primeiros artefatos a ser revisados devem ser as próprias diretrizes de revisão.

Como muitas variáveis (por exemplo, o número de participantes, o tipo de artefatos resultantes, o timing, ou tempo adequado e a duração, a abordagem de revisão específica) causam im-

pacto sobre uma revisão bem-sucedida, uma organização de software deve fazer experimentos para determinar qual abordagem funcionará melhor em um contexto local.

15.6.4 Revisões por amostragem

Em um ambiente ideal, todo artefato de engenharia de software deveria passar por uma revisão técnica formal. No mundo real dos projetos de software, os recursos são limitados e o tempo é escasso. Como consequência, as revisões são muitas vezes esquecidas, muito embora seu valor como um mecanismo de controle de qualidade seja reconhecido.

Thelin e seus colegas [The01] sugerem um processo de revisão por amostragem em que amostras de todos os artefatos da engenharia de software sejam inspecionadas para determinar quais são mais suscetíveis a erro. Recursos completos de RTF são, então, direcionados apenas para os artefatos com maior probabilidade de ser suscetíveis a erro (com base em dados coletados durante a amostragem).

As revisões tomam tempo, mas é um tempo bem empregado. Entretanto, se o tempo for reduzido e você não tiver nenhuma outra opção, não descarte as revisões. Em vez disso, use revisões por amostragem.

Para ser eficaz, o processo de revisão por amostragem deve tentar quantificar aqueles produtos de trabalho que são alvos primários para as RTFs completas. Para conseguir isso, são sugeridas as seguintes etapas [The01]:

1. Inspecionar uma fração a_i de cada artefato de software resultante i. Registrar o número de falhas f_i encontradas em a_i.
2. Desenvolver uma estimativa total do número de falhas contido no artefato i multiplicando f_i por $1/a_i$.
3. Classificar os artefatos em ordem decrescente, de acordo com a estimativa total do número de falhas contidas em cada um deles.
4. Concentrar recursos de revisão disponíveis naqueles artefatos que possuem o maior número estimado de falhas.

CASASEGURA

Problemas de qualidade

Cena: Sala de Doug Miller quando se inicia o projeto de software *CasaSegura*.

Atores: Doug Miller (gerente da equipe de engenharia de software do *CasaSegura* e outros membros da equipe de engenharia de software do produto.

Conversa:

Doug: Sei que não investimos tempo para desenvolver um plano de qualidade para este projeto, mas nós já estamos nele e temos de considerar a qualidade... Certo?

Jamie: Certamente. Já decidimos que enquanto desenvolvermos o modelo de requisitos [Capítulos 6 e 7], Ed se comprometeu a desenvolver um procedimento de testes para cada requisito.

Doug: Isso é realmente interessante, mas não iremos esperar até que os testes avaliem a qualidade, não é mesmo?

Vinod: Não! Obviamente, não. Temos revisões programadas no plano de projeto para este incremento de software. Começaremos o controle da qualidade com as revisões.

Jamie: Estou bastante preocupado, pois acredito que não teremos tempo suficiente para realizar todas as revisões. Na realidade, eu sei que não teremos.

Doug: Huumm. Então o que você sugere?

Jamie: Acho que devemos escolher aqueles elementos mais críticos dos modelos de requisitos e de projeto e os revisarmos.

Vinod: Mas e se deixarmos alguma coisa de lado em uma parte do modelo em que não fizemos uma revisão?

Shakira: Li alguma coisa sobre uma técnica de amostragem [Seção 15.6.4] que poderia nos ajudar a determinar candidatos a uma revisão. (Shakira explica a abordagem.)

Jamie: Talvez... Mas não estou certo se teremos tempo até mesmo para amostrar cada elemento dos modelos.

Vinod: O que você quer que façamos, Doug?

Doug: Usemos algo da *Extreme Programming* (*Programação Extrema*) [Capítulo 3]. Desenvolveremos os elementos de cada modelo em pares — duas pessoas — e faremos uma revisão informal de cada um deles à medida que prosseguimos. Em seguida, separaremos elementos "críticos" para uma revisão mais formal em equipe, mas manteremos essas revisões em um número mínimo. Dessa forma, tudo será examinado por mais de uma pessoa, mas ainda manteremos nossas datas de entrega.

Jamie: Isso significa que teremos de revisar o cronograma.

Doug: Que assim seja. A qualidade prevalece sobre o cronograma nesse projeto.

A fração do trabalho que é amostrada deve ser representativa do artefato como um todo e suficientemente grande para ser significativa para os revisores que fazem a amostragem. À medida que a_i aumenta, a probabilidade de que a amostra seja uma representação válida do artefato também aumenta. Entretanto, os recursos necessários para realizar a amostragem também aumentam. Uma equipe de engenharia de software deve estabelecer o melhor valor para a_i para os tipos de artefatos produzidos.[3]

15.7 Resumo

O intuito de toda revisão técnica é encontrar erros e revelar problemas que teriam um impacto negativo sobre o software a ser entregue. Quanto antes um erro for descoberto e corrigido, menor a probabilidade de que esse erro se propague a outros artefatos de engenharia de software, amplificando o problema e resultando em um esforço significativamente maior para corrigi-lo.

Para determinar se as atividades de controle da qualidade estão funcionando, deve-se reunir um conjunto de métricas. As métricas de revisão concentram-se no esforço exigido para conduzir uma revisão e nos tipos e gravidade dos erros revelados durante a revisão. Assim que os dados sobre métricas tiverem sido coletados, eles poderão ser usados para avaliar a eficácia das revisões realizadas. Os dados do setor indicam que as revisões geram um retorno significativo sobre o investimento.

Um modelo de referência da formalidade de uma revisão identifica os papéis desempenhados pelas pessoas, o planejamento e a preparação, a estrutura das reuniões, a abordagem e a verificação da correção como características que indicam o nível de formalidade em que uma revisão é conduzida. As revisões informais são superficiais por natureza, mas mesmo assim podem ser usadas efetivamente na descoberta de erros. As revisões formais são mais estruturadas e têm maior probabilidade de resultarem em software de alta qualidade.

As revisões informais caracterizam-se por um mínimo de planejamento e de preparação e um pouco de manutenção de registros. Os testes de mesa e a programação em pares caem na categoria de revisão informal.

Uma revisão técnica formal é uma reunião estilizada que se mostrou extremamente eficaz na revelação de erros. Os walkthroughs e as inspeções estabelecem papéis definidos para cada revisor, encorajam a antecipação do planejamento e da preparação, exigem a aplicação de diretrizes de revisão já definidas e tornam compulsórios a manutenção de registros e o relatório de estado das revisões. As revisões por amostragem podem ser usadas quando não é possível realizar revisões técnicas formais para todos os artefatos.

Problemas e Pontos a Ponderar

15.1. Explique a diferença entre *erro* e *defeito*.

15.2. Por que não podemos simplesmente aguardar até que os testes terminem para descobrir e corrigir todos os erros de software?

15.3. Suponha que tenham sido introduzidos 10 erros no modelo de requisitos e que cada erro será amplificado no projeto detalhado por um fator de 2:1 e que outros 20 erros de projeto sejam introduzidos e então amplificados na razão de 1,5:1 no código onde mais 30 erros são introduzidos. Suponha ainda que o teste de todas as unidades irá encontrar 30% de todos os erros, a integração descobrirá 30% dos erros remanescentes e os testes de validação encontrarão 50% dos erros restantes. Não é realizada nenhuma revisão. Quantos erros serão liberados para o campo?

3 Thelin e seus colegas realizaram uma simulação detalhada que pode ajudar a fazer essa determinação. Ver[The01] para mais detalhes.

15.4. Reconsidere a situação descrita no Problema 15.3, mas suponha agora que sejam efetuadas as revisões de requisitos, de projeto e de código e estas terão uma eficiência de 60% na descoberta de todos os erros nessa etapa. Quantos erros chegarão ao campo?

15.5. Reconsidere a situação descrita nos Problemas 15.3 e 15.4. Se cada um dos erros que chegam aos usuários custar US$ 4.800 para ser encontrado, e corrigido e cada erro encontrado na revisão custar US$ 240 para ser encontrado e corrigido, quanto é poupado em termos monetários com a realização das revisões?

15.6. Descreva com suas próprias palavras o significado da Figura 15.4.

15.7. Qual das características do modelo de referência você imagina que possua a maior influência sobre a formalidade da revisão? Justifique.

15.8. Você seria capaz de imaginar alguns casos em que um teste de mesa poderia criar problemas em vez de gerar benefícios?

15.9. A revisão técnica formal é eficaz apenas se todos estiverem preparados com antecedência. Como se reconhece um participante da revisão que não se preparou? O que você faria caso fosse o líder da revisão?

15.10. Considerando-se todas as diretrizes para a revisão apresentada na Seção 15.6.3, o que você acha que é mais importante e por quê?

Leituras e Fontes de Informação Complementares

Há um número relativamente pequeno de livros sobre revisões de software. Entre as edições recentes que dão uma orientação útil temos os de Wong (Modern Software Review, IRM Press, 2006), Radice (High Quality, Low Cost Software Inspections, Paradoxicon Publishers, 2002), Wiegers (Peer Reviews in Software: A Practical Guide, Addison-Wesley, 2001) e Gilb e Graham (Software Inspections, Addison-Wesley, 1993). Freedman e Weinberg (Handbook of Walkthroughs, Inspections and Technical Reviews, Dorset House, 1990) permanece um clássico e continua a fornecer informações úteis sobre esse tema importante.

Uma ampla gama de fontes de informação sobre revisões de software se encontra à disposição na Internet. Uma lista atualizada de referências que são relevantes sobre revisões de software pode ser encontrada no site **www.mhhe.com/engcs/compsci/pressman/professional/olc/ser.htm**.