A IMPORTÂNCIA DO PLANEJAMENTO DOS TESTES NO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

Vivaldo José Breternitz *
Edson Saraiva de Almeida **

RESUMO

O processo de desenvolvimento de software envolve uma série de atividades nas quais, apesar das técnicas, métodos e ferramentas empregados, falhas no produto podem ser inseridas já nas fases iniciais do ciclo de desenvolvimento. Testes de "software" têm como objetivo a descoberta de falhas e devem ser planejados de forma cuidadosa, permitindo encontrar e corrigir o maior número de defeitos, o mais cedo possível, para minimizar o custo de correção e a probabilidade de ocorrência de falhas quando o "software" já estiver em regime de produção. Esse trabalho tem como objetivo despertar a consciência dos profissionais de Tecnologia da Informação para a importância de testes planejados e executados de maneira sistemática, bem como fornecer subsídios que os auxiliem no processo de planejamento desses testes.

PALAVRAS-CHAVE: *testes, sistemas, programas, desenvolvimento, erros, falhas, planejamento, software.*

ABSTRACT

The process of software development involves a series of activities in which, regardless the techniques, methods and tools used, defects may be introduced in the product, even in the initial stages of the development cycle. Software tests have the end of discovering defects and must be planned carefully, making it possible to find and correct the highest number of errors, as early as possible, to minimize the cost of correction and the probability of the occurrence of defects when the software is already in the phase of production. This work has the objective of arousing the conscience of the professionals of Information Technology to the importance of planned and executed tests in a systematic way, as well as supplying them with supporting information, that may help them in the process of planning these tests.

KEY WORDS: t*esting, systems, programs, development, errors, failures, planning, software.*

* Mestre em Engenharia Elétrica pela Universidade Mackenzie; consultor de empresas e professor da Universidade Mackenzie e das Faculdades Padre Anchieta. Rua. Profa. Joceny V. Curado, 73 - CEP 13206-320, Jundiaí/SP. Fone 11 4607.3616, vjbreternitz@mackenzie.com.br.

** Mestre em Engenharia de Computação pelo IPT - Instituto de Pesquisas Tecnológicas, consultor de empresas e professor da Universidade São Judas Tadeu. Rua Taquari, 546 - São Paulo, Mooca CEP 03166-000, tel 0800-11-1677. prof.ealmeida@usjt.br

Introdução

Por muitos anos, as empresas têm aceitado como um custo do negócio o retrabalho e os erros gerados durante o processo desenvolvimento de software. A competição atualmente vigente no mundo globalizado está mudando este panorama e colocando a qualidade como principal fator para o sucesso de uma organização. Mesmo diante desse cenário, a maior parte das empresas ainda não consegue avaliar a ineficiência gerada pela falta de uma abordagem disciplinada no processo de desenvolvimento do software.

A fase de testes é, juntamente com a de documentação, uma das mais desprezadas do desenvolvimento de um "software". A ela são usualmente alocados os profissionais menos experientes, o prazo fixado para a mesma, freqüentemente é diminuído e destinado a outras atividades.

Essa postura sempre gera problemas, desde pequenos inconvenientes aos usuários do "software", até grandes tragédias. Dentre essas, sempre são lembradas:

• ¨ a pane generalizada nos serviços de telefonia 0800, ocorrida nos Estados Unidos em 1991, quando um novo sistema estava sendo colocado em produção;

• ¨ auto destruição em 1996 do foguete francês Arianne 5, que em função de um "overflow" desviou-se do rumo. Os franceses acreditavam estarem iniciando uma nova era no que se referia a foguetes lançadores de satélites (ARNOLD, 2002);

• ¨ o envenenamento, por overdose de radiação, de seis pessoas que vinham passando por radioterapia com o uso do equipamento Therac-25, controlado por computador (RAWLINSON, 1987);

• ¨ a falha de um míssil americano Patriot, ao deixar de interceptar um Scud iraquiano que viria a matar dezenas de americanos durante a Guerra do Golfo (1991). Essa falha foi causada por um arredondamento indevido (ARNOLD, 2002);

• ¨ mais recentemente, a virtual paralisação da Internet nos Estados Unidos em 3 de outubro de 2002, em função do "travamento" dos canais de telecomunicações, causado por uma falha em "upgrade" de "software" que vinha sendo executado pela WorldCom, empresa responsável por cerca de metade do tráfego de Internet naquele país.

A experiência dos autores registra inúmeros problemas semelhantes, embora não tão graves; dentre estes:

• ¨ o pagamento de auxílio paternidade a uma freira que ministrava aulas na rede pública de ensino;

• ¨ a emissão de talões de cheques com o nome do segundo titular (geralmente a esposa) trocado (imagine-se a esposa de um desses primeiros titulares sendo erroneamente levada a crer que seu marido mantinha uma conta conjunta com outra mulher…).

Sob o ponto de vista de custos, o assunto é também bastante sério: o Departamento do Comércio dos Estados Unidos estima que o problema custa àquele país cerca de 60 bilhões de dólares ao ano, e que melhorias no processo de valida-

ção de software poderiam reduzir esse custo em cerca 35%.

Além dos inúmeros problemas a que as empresas se sujeitam, em função da utilização cada vez maior de sistemas de computador, e dado o atual cenário, em que cada vez mais freqüentemente as organizações recorrem a sistemas integrados ou delegam seu desenvolvimento a terceiros, parece-nos claro que a tendência é de que sejam aperfeiçoados os processos de especificação de software e em especial os de teste.

Acreditamos que a situação possa evoluir para um ponto que o relacionamento entre clientes e fornecedores de software evolua para algo similar ao que existe entre clientes e fornecedores de outros bens e serviços, com uma entidade como o Procon intervindo no processo. Se isso se confirmar, os fornecedores certamente exigirão que os clientes deem um aceite mais formal nos sistemas entregues, de forma a eximi-los de responsabilidades; por sua vez, os clientes deverão testar os produtos recebidos, de forma a que possam certificar-se do bom funcionamento do produto adquirido e preservar seus direitos. Nos Estados Unidos, a National Academy of Sciences pede a criação de uma lei de garantia de qualidade de software.

No processo de desenvolvimento de "software" é comum a geração de erros não só por falhas de codificação, mas por interpretação incorreta de informações em função de falhas de comunicação. Estes erros podem ser gerados em qualquer fase do processo, desde sua concepção até a de codificação propriamente dita. Técnicas de garantia da qualidade se propõem a auxiliar na descoberta destes erros o mais cedo possível, para que os custos de correção sejam menores e os impactos em termos de perda de prazos ou sinistros como os acima mencionados sejam minimizados.

Isso posto, sendo os objetivos desse artigo despertar a consciência dos profissionais de Tecnologia da Informação para a importância de testes planejados e executados de maneira metodologicamente correta, bem como fornecer subsídios que os auxiliem nesse propósito, passamos a discutir pontos que julgamos relevantes acerca do assunto.

Planejamento dos testes

O Plano de Teste é um guia em alto nível que direciona a natureza, restrições, resultados esperados e orienta o desenvolvimento de todas as atividades relacionadas ao teste. Elementos determinantes para o planejamento são os resultados do impacto de possíveis falhas e o tipo de sistema que está sendo testado. Fatores críticos para o sucesso do esforço de teste, como o nível de risco e o nível apropriado de esforço na atividade de teste devem ser explicitados. O escopo do teste e quem o executará são determinados. O balanceamento entre cronograma, custo, escopo e abordagem tecnológica também é estabelecido pelo Plano.

Em muitos casos, pode-se concluir que não existem recursos ou tempo disponíveis para executar todos os testes de maneira abrangente e convencional. Nestes casos, deve-se priorizar os testes de funções do sistema críticas para o suporte ao negócio da empresa.

Estratégia de teste

Alguns desenvolvedores de sistemas continuam a acreditar que só devem se preocupar com a qualidade depois que o código estiver pronto. Não concordamos com essa afirmativa; segundo Pressman (PRESSMAN, 1997) a garantia da qualidade é uma atividade "guarda-chuva", abrangendo todo o processo de produção do software. Essa garantia é objeto de diversas normas (ISSO, IEEE), cuja observação durante todo o processo auxilia consideravelmente na produção de software de boa qualidade.

Segundo Pressman (PRESSMAN, 1997), uma estratégia de teste integra técnicas de projeto de casos de teste numa série definida de passos que leva à construção bem sucedida do software, Esses passos podem envolver uma combinação das técnicas e aplicação de diferentes critérios de teste.

Os testes iniciam-se no nível de módulos e prosseguem "para fora", na direção da integração de todo o sistema de computador. Diferentes técnicas de teste são apropriadas a diferentes etapas do projeto. Os testes usualmente são executados pela equipe que está desenvolvendo o "software"; já no caso de grandes projetos, por grupo de testes independente, freqüentemente exterior às organizações que desenvolvem ou serão usuárias do novo sistema; no jargão da área, esse grupo é chamado ITG (*Independent Test Group*).

A estratégia descrita por Pressman prevê as seguintes etapas:

• ¨ teste de unidade (unity test), no qual são focalizados cada um dos módulos de um programa, individualmente, garantindo que os mesmos funcionem adequadamente;

• ¨ teste de integração, visando a assegurar que o programa construído a partir dos módulos testados no nível de unidade funcione adequadamente, testando-se especialmente as "interfaces" entre os módulos;

• ¨ teste de validação, considerando o conjunto de programas de forma integrada, validando os requisitos funcionais e de desempenho definidos durante a fase de análise;

• ¨ teste de sistema, onde o "software" produzido é testando em conjunto com outros elementos, como hardware, usuários, gerenciador de banco de dados, etc.

A implementação dessa estratégia evidentemente varia em função de inúmeros fatores, dentre os quais tamanho, complexidade e criticidade do sistema, características dos desenvolvedores e usuários, etc. Como o objetivo deste trabalho é despertar a consciência dos profissionais envolvidos com o assunto, passaremos a expor algumas técnicas e critérios de teste.

Técnicas e critérios de teste

As técnicas de teste podem ser divididas em dois grandes grupos, os chamados testes "caixa-preta" e "caixa-branca". Os testes "caixa-preta", também chamados testes funcionais, testam o sistema do ponto de vista do usuário, não levando em conta a estrutura interna e a forma de implementação do sistema. Estes são os

testes usualmente praticados quando não se dispõe do código-fonte, como ocorre quando se testa pacotes de software. Os testes "caixa-branca", assim chamados por figurativamente podermos enxergar o interior do sistema, procuram atingir todo o código; evidentemente, para esses casos é necessário que o código fonte esteja disponível.

Para facilidade de compreensão, vamos nos concentrar nos testes "caixa-preta". Um dos critérios utilizados por esta técnica é o de "classes de equivalência" (MYERS, 1979). De acordo com esse conceito, já que não se pode testar todos os casos possíveis, deve-se dividi-los em classes, de modo que os casos dentro de cada classe sejam "equivalentes", permitindo que se teste apenas um subconjunto deles, mantendo a representatividade.

Um dos aspectos mais problemáticos desse tipo de teste é determinar as classes de equivalência. Estas devem ser definidas de forma a que agrupem casos de teste para os quais o comportamento do sistema deva ser o mesmo. Na prática, é necessário selecionar atributos das transações do sistema que sejam determinantes para o comportamento esperado. Em um sistema que processa as operações realizadas em bolsas de valores pelos clientes de uma corretora, um atributo determinante para uma transação de liquidação financeira dessas operações poderá ser a forma de liquidação: débito ou crédito em conta corrente, Transferência Eletrônica Disponível, etc. A combinação desse atributo com outros, como por exemplo, transação de compra ou venda, irá definir as classes de equivalência.

As classes de equivalência podem ser determinadas para as entradas e saídas do sistema, sendo chamadas classes de entrada ou de saída. Uma vez claramente definidas essas classes, pode-se construir a matriz de entradas/saídas do sistema, que conterá, para cada classe de entrada, a classe ou as classes de saída correspondentes. No caso de operações em bolsas, para cada classe de entrada acima mencionada (compra de ações liquidada com cheque, venda de ações liquidada via crédito em conta corrente, etc.), deverão ser gerados diferentes resultados na classe de saída "Relatório de Liquidações do Dia".

Determinadas as classes de entrada e de saída e a matriz de correspondência entre elas, pode-se construir os casos de teste. Recomenda-se que seja definido pelo menos um caso de teste para cada classe de equivalência, estabelecido de acordo com a natureza de cada classe. No caso anterior de valores a pagar ou a receber, poderiam ser estabelecidas as diversas classes, como por exemplo, valor zero, valor típico e um valor composto por noves do tamanho máximo do campo "valor a liquidar".

Definidos os casos de teste, esses devem ser reunidos em roteiros de teste. Um roteiro de teste é constituído por um ou mais casos de teste, de modo a reproduzir um conjunto de transações do mundo real. Um roteiro de teste poderia ser, ainda no caso de uma corretora, a abertura de uma ordem de compra ou venda, a execução dessa ordem, a apuração dos valores a liquidar e a passagem desses valores para o Sistema de Pagamentos Brasileiro e para outros sistemas que

operacionalizariam formas de liquidação diferentes. Para cada roteiro de teste, devem ser previamente determinados os resultados esperados. A elaboração de roteiros é um trabalho bastante extenso, exigindo quase sempre a participação dos usuários do sistema, daqueles que o especificaram e do pessoal responsável pelos testes.

Atenção especial deve ser prestada ao cronograma de desenvolvimento do sistema: as atividades até agora mencionadas devem ser planejadas de forma a que ocorram paralelamente às demais, de forma a que quando os roteiros estejam prontos, se possa iniciar imediatamente os testes. No outro sentido, é difícil elaborar-se os roteiros de forma muito antecipada, pois a prática mostra que os sistemas vão sofrendo alterações de especificações durante o processo de desenvolvimento.

A fase seguinte é a de execução dos testes, sendo os resultados obtidos comparados com os esperados. As divergências, chamadas usualmente ocorrências, podem ser resultados de erros do sistema ou de especificação. Erros de especificação devem ser remetidos aos usuários para correção e deverão gerar alterações no sistema. Nos casos de erros de sistema, a recomendação é a identificação dos componentes afetados, e a correção das falhas.

Corrigidos os erros referentes a uma dada ocorrência, os testes que permitiram sua identificação devem ser novamente executados, repetindo-se o ciclo até que se possa ter segurança quanto à eliminação das falhas detectadas. Antes que o sistema entre em produção, todos os roteiros de teste devem ser executados novamente, pois o processo de correção dos defeitos gera a possibilidade da introdução de novos erros nos programas.

A possibilidade de automação da execução dos testes deve sempre ser investigada e aplicada quando possível. Para isso estão disponíveis ferramentas automatizadas, capazes de simular os comandos gerados por usuários de ambientes CICS, Windows, etc. O controle da execução dos testes, o armazenamento dos casos e dos roteiros de teste, a comparação entre os resultados gerados e os esperados, tudo isso é feito de forma automática por essas ferramentas, facilitando em muito os testes, especialmente quando comparados àqueles executados com a ajuda apenas de geradores de arquivos como os utilizados em ambientes "mainframe" nas décadas de 70 e 80.

Ao processo de testes devem ser aplicadas ferramentas de gerenciamento de projetos (como as especificadas pelo PMI Standards Committee (PMBOK, 1996), por exemplo), de forma a garantir cumprimento de prazos, economicidade de recursos etc. Métricas, documentação dos testes, comunicação entre os participantes do processo e as instâncias superiores devem ser garantidas. Aliás, a ausência de patrocínio dessas instâncias é fator que certamente levará ao fracasso qualquer processo de teste.

A observação de métricas de qualidade é outro ponto importante e usualmente negligenciado. Pode-se falar em quatro aspectos a serem considerados:

• ¨ corretitude, ou grau em que o “software” executa as funções que lhe são exigidas, usualmente expressa pela relação Erros/Kloc (“Thousand Lines of Code” ou milhar de linhas de código);

• ¨ manutenibilidade, grau de facilidade com que o software pode ser corrigido, adaptado ou ampliado. É medida em termos de Tempo Médio para Mudança, que corresponde ao tempo gasto para analisar um pedido de mudança, projetar a modificação adequada implementá-la, testá-la e distribuí-la aos usuários;

• ¨ integridade, que é a capacidade que um “software” tem de suportar ataques (acidentais ou intencionais) à sua integridade.. O nível de integridade usualmente é expresso por £ = (1 - ameaça . (1 - segurança)), onde ameaça é a probabilidade de que um ataque de um tipo específico ocorra dentro de determinado tempo e segurança a probabilidade de que o ataque seja repelido;

• ¨ usabilidade ou “user friendliness” do “software”, que pode ser medida sob quatro aspectos: habilidade física/intelectual para se aprender a utilizar o sistema, tempo exigido para que um usuário se torne moderadamente eficiente no uso do software, aumento de produtividade de usuário moderadamente eficiente e avaliação subjetiva (através de questionário ou similar).

Reiteramos nossa posição de que o efetivo gerenciamento de projeto, entendido no sentido de aplicação de conhecimentos, habilidades, ferramentas e técnicas as atividades do projeto a fim de atender ou superar as necessidades e expectativas que os interessados (stakeholders) possuem no projeto (PMBOK 1996), é condição “sine qua non”, até mesmo mais importante que a aplicação estrita das técnicas de teste. Efeitos clássicos da aplicação de testes sem controle gerencial são as redundâncias, o retrabalho, o não teste de algumas funcionalidades, etc., tudo gerando aumento de custos, prazos e principalmente, riscos.

Uma questão vem à tona todas as vezes que testes de “software” são discutidos: quando se pode afirmar que o sistema foi suficientemente testado? O estabelecimento de critérios de cobertura permite responder a essa questão por garantir que um nível adequado de segurança será dado ao sistema. Critérios de cobertura podem, por exemplo, definir quais funcionalidades serão testadas, em função de sua criticidade ou da disponibilidade de recursos e prazos para os testes. Como já se disse, estes critérios devem ser definidos no planejamento, de forma a garantir os recursos necessários para conclusão dos testes.

Conclusão

O processo de teste pode vir a consumir grande parte dos recursos orçados para o desenvolvimento do sistema. Quando estes custos não são previstos, devido a limitações ou desconhecimento da organização, a atividade de teste acaba sendo relegada ao segundo plano. A conseqüência freqüentemente é a entrega de software com baixa qualidade. O planejamento das atividades contribui para melhorar a efetividade e eficiência do teste, minimizando os custos da correção de defeitos.

A maioria das empresas utiliza metodologias de desenvolvimento de “software” que detalham as fases o os marcos do ciclo de desenvolvimento especificando os produtos de cada fase. É importante planejar-se um processo de testes que acompanhe o ciclo de desenvolvimento de “software” para assegurar um aperfeiçoamento do processo melhorando a qualidade final do produto de “software”.

Referências bibliográficas

ARNOLD, D.N. *Two disasters Caused by Computer Arithmetic Errors*,em http://www.math.psu.edu/dna/455.f96/disasters. html, abril de 2002.

MYERS, G. *The art of software testing*. New York: J. Wiley, 1979.

PMBOK Guide. *Project Management Body Of Knowledg*e, Newtown Square, PA: Project Management Institute, 1996.

PRESSMAN, R. S. *Software engineering: a practitioner’s approach*. New York: McGraw-Hill,1997.

RAWLINSON, J. A. (maio de 1987) The Therac-25 Experience. em *OCTRF/OCI Physicists Meeting*, Kingston, Ontário.