Qualidade de Software

A demanda por qualidade de software tem motivado a comunidade de software para o desenvolvimento de modelos para a qualidade.

Um software de qualidade é fácil de usar, funciona corretamente, é de fácil manutenção e mantém a integridade dos dados para evitar possíveis falhas, fora ou não, do seu controle. Para o desespero de seus usuários as falhas se apresentam sem avisos prévios, gerando um impacto econômico e social muitas vezes irremediável.

Os custos resultantes de defeitos ou erros provocados por falha de softwares, tanto para as empresas de softwares como para usuários, poderiam ser catastróficos, bancos poderiam perder milhões de dólares e clientes veriam seus dinheiros sumirem.

Em 1991 aconteceu uma pane no sistema telefônico da Califórnia e de toda a costa, motivo? A modificação de apenas três das milhões de linhas de código introduziu um bug. Este é um exemplo que demonstra de quanto dependemos das maquinas e de seus softwares. A tecnologia da Informação e comunicação transversa lizam cada vez mais todos os níveis das atividades humanas, tornando assim a importância da qualidade de software.

A qualidade é hoje o grande motivador em todas as áreas de atividade humana, todos querem oferecer e receber produtos e serviços com qualidades.

Um software de qualidade oferece segurança de pessoas, disponibiliza serviços essenciais (home banking, telefonia, etc), gera competitividade das empresas, etc.

A qualidade de um software deve estar em conformidade com especificações e padrões de desenvolvimento, há necessidades explícitas e objetivos propostos por aquelas pessoas que produzem software, garantindo que, tanto o produto do software quanto um bom processo de desenvolvimento, atinjam níveis de qualidade mantendo sempre as necessidades dos usuários.

O Que e Qualidade?

Segundo Herman G. Weinberg “A qualidade é relativa. O que é qualidade para uma pessoa pode ser falta de qualidade para outra.

A definição de qualidade pode parecer simples e até trivial. Contudo se observada mais detalhadamente pode se perceber sua complexidade.

Um dos principais desafios enfrentados pelos profissionais de qualidade e teste de software está em definir o que é qualidade no contexto do produto atual. Analisando então as variáveis disponíveis para o caso em concreto, tais como: orçamento, tempo e prazo.

Visando estabelecer uma visão comum entre todos os membros da equipe e seus usuários é necessário que sejam vencidas as dificuldades de comunicação entre os transmissores e os receptores da mensagem.

O Que É Qualidade de Software?

Para ajudar nessa questão, a International Organization Standardization - ISO e a International Electrotechnical Comission-IEC, que são organismos normalizadores com importância internacional reconhecida no setor de software, se uniram para editar normas internacionais conjuntas.

A norma internacional ISO/IEC, define qualidade de software como A totalidade de características de um produto de software que lhe confere a capacidade de satisfazer necessidades explícitas e implícitas.

Necessidades explícitas são aquelas definidas no requisito proposto. Esses requisitos devem definir as condições em que o produto deve ser utilizado e dizer seus objetivos, funções e desempenho esperado. São, portanto, fatores relativos à qualidade do processo do desenvolvimento do produto que são percebidos somente pelas pessoas que trabalharam no seu desenvolvimento.

Necessidades implícitas são aquelas que, embora não expressas no documento do produtor, são necessárias para o usuário. Estão englobados em esta classe os requisitos que não precisam ser declarados por serem óbvios, mas que pela gravidade de suas consequências devem ser levados em consideração (ex: mesmo em condições não previstas de erro ou má operação, um sistema de administração hospitalar não pode provocar a morte de pacientes).

As necessidades implícitas também são chamadas de qualidade em uso e devem permitir aos usuários atingir metas com efetividade, produtividade, segurança e satisfação em seu uso diário.

Qualidade

Está claro que produzir software de qualidade é uma meta essencial e básica da Engenharia de Software, que oferece métodos, técnicas e ferramentas. O necessário mesmo é que o software seja confiável, eficaz e siga os padrões exigidos pelo contexto. Freeman [Freeman 87] apresenta uma distinção entre qualidade básica e qualidade extra.

Em qualidade básica ele lista: funcionalidade, confiabilidade, facilidade de uso, economia e segurança de uso. Em qualidade extra ele lista: flexibilidade facilidade de reparo, adaptabilidade, facilidade de entendimento, boa documentação e facilidade de adicionar melhorias. Essas prioridades vão depender muito de cada caso e do custo de cada uma dessas qualidades.

A sociedade pressiona o setor de software para que a característica qualidade seja preponderante.

Qualidade e Requisitos

Segundo Crosby [1992] “A qualidade é a conformidade aos requisitos”. Esta definição nos é bastante interessante, pois evidencia qual o caminho a seguir para julgar a qualidade de um software.

Partindo deste pressuposto é necessário considerar três fatores para a correta verificação da conformidade aos requisitos proposta por Crosby.

Definição de Conformidade

Para a correta verificação da qualidade torna-se necessário uma prévia definição das margens de precisão dos resultados esperados. Tornando possível a medição em graus de qualidade do produto final.

Metodologia de Observação dos Resultados

É necessário ter em mente que a medida observada pode conter margens de erros. Existem diversos fatores que podem corromper os dados utilizados na observação.

Conciliar os interesses dos diversos Stakeholders

A qualidade está intimamente ligada aos requisitos e estes requisitos são definidos por alguém, logo a qualidade depende das escolhas que este alguém fez.

Muitos desenvolvedores de software devem conciliar, resolver ou minimizar os problemas organizacionais do cliente que contratou o desenvolvimento do produto.

Requisitos de software

Os requisitos do software são as descrições sobre o comportamento, as funcionalidades e especificações do software.

Todos os requisitos do software devem ser registrados no documento de requisitos que deve ser conciso e consistente para fornecer informações claras ao leitor.

Para a correta especificação de requisitos é necessário que os mesmos sejam precisos, completos, consistentes e tenham controle de rastreabilidade e Modificabilidade.

Seguindo a divisão tradicional os requisitos podem ser classificados em requisitos funcionais e não-Funcionais.

Qualidade de Software na Visão do Usuário

Os desenvolvedores de software não podem menosprezar o papel do usuário/cliente, não podendo se esquecer das necessidades implícitas do uso.

Cada cliente pode ter desejos e necessidades diferentes em relação ao mesmo tipo de produto.

A Pergunta e Qual o Interesse dos Usuários de Software?

Os usuários estão mais interessados no uso do software, na sua funcionalidade, no desempenho e nos efeitos que o uso possa produzir na sua empresa, organização, etc.

O cliente valoriza que o software responda às suas necessidades. A qualidade interna tem que responder às questões externas que o cliente venha a questionar. Por exemplo:

As Funções X Estão Disponíveis e São Executadas Eficientemente?

Funciona corretamente em imprevistos, como, por exemplo, efetuar débito em uma conta com saldo negativo?

O software é seguro, ou seja, evita que pessoas ou sistemas não autorizados tenham acesso aos dados para modificar?

É fácil de usar ou requer muito treinamento?

É fácil de integrar com outros sistemas existentes?

Aceita trabalhar com arquivos de outros sistemas ou enviar dados para outros sistemas?

É importante considerar que o cliente é quem está à frente. Ele tem poder de participação no processo externo do seu produto.

Hoje o mercado é mais competitivo, aumentando a oferta de produtos, e o cliente está mais consciente do seu poder. Esta mudança no mercado, maior globalização, leva a exigir melhor qualidade de produtos e processos para atender a esse novo cliente.

Qualidade de processo e produto: é possível medir a qualidade de software?

O controle de qualidade surge então como uma necessidade: inicialmente são feitas verificações, logo em seguida adotam-se técnicas e critérios bem definidos, em alguns casos chega-se a verificação de 100% dos produtos para eliminação daqueles produzidos com defeitos; certifica-se a qualidade do produto com um custo bem alto.

Como alternativa busca-se melhorar o processo de produção para se adquirir maior confiança na qualidade do produto final.

A engenharia de software tem como objetivo a melhoria da qualidade do seu produto com propostas e modelos de desenvolvimento, métodos e técnicas para aplicação nas diversas fases de desenvolvimento.

É importante a avaliação da qualidade de software nas duas visões, processo e produto, é aqui que se direciona o esforço.

O principal problema com que se defronta a engenharia de software é a dificuldade de se medir a qualidade. Há de se considerar que o software não se desgasta, por tanto, tal método de medição de qualidade não pode ser aproveitado.

A ISO/IEC fornece um modelo de propósito geral (Tabela 1) o qual define seis amplas categorias de características de qualidade de software que são subdivididas em sub características.

O modelo proposto pela ISO/IEC 9126(NBR13596) tem como objetivo servir de referência básica na avaliação de produto de software.

Características	Sub-Características	Significado
Funcionalidade: O conjunto de funções satisfaz as necessidades explícitas e implícitas para a finalidade a que se destina o produto?	Adequação	Propõe-se a fazer o que é apropriado?
	Acurácia	Gera resultados corretos ou conforme acordados?
	Interoperabilidade	É capaz de interagir com os sistemas especificados?
	Segurança de acesso	Evita acesso não autorizado, acidental ou deliberado a programas de dados?
	Conformidade	Está de acordo com normas e convenções previstas em leis e descrições similares?
Usabilidade: É fácil usar o software?	Inteligibilidade	É fácil entender os conceitos utilizados?
	Apreensibilidade	É fácil apreender a usar?
	Operacionalidade	É fácil de operar e controlar a operação?
Eficiência: Os recursos e os tempos utilizados são compatíveis com o nível de desempenho requerido para o produto?	Comportamento em relação aos recursos	Quanto recurso utiliza?
	Comportamento em relação ao tempo	Qual é o tempo de resposta e de processamento?
Manutenibilidade: Há facilidade para correções, atualizações e alterações?	Analisibilidade	É fácil encontrar uma falha quando ocorre?
	Modificabilidade	É fácil modificar e remover defeitos?
	Estabilidade	Há grandes riscos de bugs quando se faz alterações?
	Estabilidade	É fácil adaptar a outros ambientes sem aplicar outras ações ou meios além dos fornecidos para esta finalidade no software considerado?
Portabilidade: É possível utilizar o produto em diversas plataformas com pequeno esforço de adaptação?	Adaptabilidade	É fácil adaptar a outros ambientes sem aplicar outras ações ou meios além dos fornecidos para esta finalidade no software considerado?
	Capacidade para ser instalado	É fácil instalar em outros ambientes?
	Capacidade para substituir	É fácil substituir por outro software?
	Conformidade	Está de acordo com padrões ou convenções de portabilidade?

Considerações Práticas Para Garantir a Qualidade Do Produto

Para se garantir a qualidade de um produto durante seu desenvolvimento existe um conjunto de métodos e técnicas que devem ser implementadas durante todo o processo. Nesta seção, consta uma breve definição de algumas dessas atividades.

Identificação de Defeitos

Conhecer os conceitos de erro, defeito e falha é o requisito básico para se construir um processo de garantia de qualidade.

É preciso saber identificar o problema, antes de investigar sua proveniência e, desta forma, saber corrigi-lo.

Erro: Trata-se de um engano de um indivíduo.

Defeito: Deficiência mecânica ou algorítmica que se ativada pode levar a uma falha. Na tabela 2 há a definição dos tipos de defeitos encontrados.

Falha: Evento notável onde o sistema viola suas especificações.

Quanto antes a presença do defeito for revelada, menor o custo de correção do defeito e maior a probabilidade de corrigi-lo corretamente. A solução é introduzir atividades de VV&T ao longo de todo o ciclo de desenvolvimento.

Informação Estranha	Informação desnecessária.
Fato Incorreto	Informação que não é verdadeira para as condições especificadas.
Inconsistência	Informações conflitantes
Ambiguidade	Informação passível de ter múltiplas interpretações
Omissão	Informação necessária não incluída

Validação, Verificação e Testes (VV&T)

As atividades de VV&T atuam na garantia de qualidade, assegurando que o software cumpra as especificações definidas e atenda às necessidades de seus usuários. Definições:

Validação: Avalia um sistema ou componente para determinar se ele satisfaz os requisitos para ele especificados. “Estamos construindo o produto certo?”. O software deve atender às necessidades dos usuários.

Verificação: Avalia um sistema ou componente para determinar se os produtos de uma dada atividade de desenvolvimento satisfazem as condições impostas no início desta atividade. “Estamos construindo certo o produto?” Os artefatos construídos devem estar de acordo com a especificação do software.

Os principais métodos de Validação e Verificação são:

Estática: Revisões de Software (Revisões aos Pares, Inspeções de Software, entre outros). Denomina-se estática, pois não envolve a execução do produto, mas apenas leitura de artefatos, como código.

Dinâmica: Testes de Software (Testes de unidade, de integração, do software, entre outros).

Revisões de Software

São processos ou atividades para leitura de um artefato de software visando assegurar que ele cumpre sua especificação e atende às necessidades de seus usuários. Tem por objetivo realizar validação e verificação estática de artefatos de software. Pode ser aplicada a qualquer artefato produzido ao longo do processo de desenvolvimento de software.

Tipos de Revisões

Inspeções de Software: mais focadas em encontrar defeitos.

Walkthroughs: são mais apropriados para atividades de brainstorming, para explorar alternativas de projeto e resolução de problemas.

Testes de Software

Processo de executar um programa ou sistema com o objetivo de revelar a presença de erros; ou, falhando nesse objetivo, aumentar a confiança sobre o programa. Envolve todo um processo de planejamento, fases de teste, etc., que necessitaria de um outro artigo para aprofundar este assunto.

Os desenvolvedores fazem os softwares, mas os clientes são os que vão usá-los. Por isso, há a necessidade urgente de sistematizar formas de evitar os custos elevados consequente dos defeitos dos softwares e dos erros não intencionais dos usuários.

Isto só será possível se forem priorizadas algumas características de qualidade de software: usabilidade, confiabilidade, funcionalidade e manutenibilidade. Sendo esses, os requisitos essenciais do produto de software exigidos pelos compradores e atendidos pelos vendedores. Assim veremos resultados positivos tendo softwares de qualidade e mais úteis.

No Brasil ainda existem empresas de software que não utilizam técnicas para melhorar a qualidade de seus produtos, por isso aquelas empresas que desenvolvem software de qualidade são mais competitivas, o que favorece a abertura de mercados externos para os softwares brasileiros e beneficiando o mercado interno que acabará por exigir software melhores e a menor preço.

Não Existem Requisitos Ou Documentação

Geralmente desenvolvedores não fazem a tarefa de casa levantando os requisitos de software, já começam a escrever o código conforme é pedido. Ou, pior, não fazem as documentações necessárias de análise antes do desenvolvimento de um software. Isso causa diversos problemas:

- Quando o desenvolvedor desconhece os requisitos, ele provavelmente voltará a reescrever um código que já foi escrito, pela falta de análise, desordem, descaso por não levantar os requisitos;

- Às vezes pode até haver requisitos, mas numa equipe cada desenvolvedor pode interpretar os requisitos de uma forma diferente;

- Requisitos mal levantados ou mal escritos por falta de conhecimento ou experiência;

- Falta de uma documentação do software;

- Documentação mal elaborada.

Quando há a falta de documentação o serviço de software fica complicado em diversos sentidos: há uma grande dificuldade de encontrar mão de obra especializada, e quando encontram, devido à falta de documentação, os contratados levam meses para conhecer o software da empresa, afinal, o conhecimento está na cabeça de alguns; é difícil fazer teste de software para garantir um menor risco de erros; o desenvolvimento de software é comprometido aumentando as chances de erros do software. Estes são apenas alguns exemplos.

Não Existe a Fase de Projeto de Software

Simplesmente levantam-se rapidamente os requisitos e inicia o desenvolvimento, sem uma análise e projeto. Isso leva a falta de documentação e início de vários problemas de, no futuro, ter que reescrever códigos com erro, além de dificultar para a equipe de testes a análise do software.

Controle de Mudanças e de Versões Inadequadas ou Inexistentes

Não há um controle do que foi mudado, um muda o banco de dados sem comunicar e interferem outros módulos do software. Não há uso de controle de versões. O que é um erro. Há diversas ferramentas que fazem este controle, como o CVS.

Foco na Entrega

Muitos pensam em entregar o mais rápido o software, sem se preocupar com a qualidade. Isso faz o software voltar com diversos erros encontrados, muitas vezes pelos clientes, o que é péssimo. É melhor colocar um prazo real e entregar com qualidade, que correr para entregar rápido. Há empresas que eliminam a análise de documentação do projeto em nome de "produtividade", mas está suposta "produtividade" é apenas ilusória.

Inexistência de Um Time de Testes

Geralmente quem faz os testes são os desenvolvedores. Isso é um erro. Eles seguem uma lógica que é diferente da lógica de usuários finais. Ou quando tem uma equipe de testes, é a secretária, o office-boy, ou outros funcionários da empresa que não fazem um teste correto. Ou a falta de ferramentas de automação de testes. Ou o teste é realizado pelos próprios clientes, o que é pior.

Time de Testes Focado em Testes Superficiais

Quando há, como foi citado no item anterior, não utilizam ferramentas de automação de testes, de ferramentas de gestão de testes, de gestão de defeitos, etc. Fazem aquele teste de usuário, mesmo assim superficial, não fazem teste do banco de dados, do desempenho do sistema, de análise do código, não catalogam os defeitos em níveis de defeitos, não criam regras de mensagens de erros para ajudar ao usuário a contornar os possíveis erros, etc. Hoje montar uma equipe de testes é importante, se estuda muito teste de software, são raros os profissionais especializados e são muito bem remunerados.

Desenvolvimento Reativo

O desenvolvimento é focado na correção de erros, ao invés da evolução do software ou novas soluções. Isso devido aos fatores já mencionados.

O teste e a qualidade de software hoje em dia são tão importantes e é um assunto muito discutido no mundo todo e já há alguns anos no Brasil, porque hoje tudo depende de software: o piloto de avião para navegar, sistemas de pagamentos de contas, sistemas bancários, enfim, uma afinidade de softwares que todos nós dependemos.

Um erro pode colocar a vida de pessoas em risco, perdas de negócio e produtividade, prejuízos financeiros, comprometimento da reputação da empresa, etc. Vemos quase que diariamente notícias de empresas prejudicadas devido a erros em softwares.

Hoje softwares se comunicam com tudo: WebService de nota fiscal eletrônica, PAD, celulares, internet, rede, etc. Devem se portar para os mais diversos sistemas operacionais, os mais diversos browsers, e tudo isso leva a uma maior complexidade, não apenas de código, mas de análise, e uma boa documentação para manutenção e continuidade dos softwares.

Sem um bom levantamento de requisitos, análise, documentação e projeto a chance de erros é enorme e pior: sem todos estes passos antes do desenvolvimento será um grande pesadelo difícil de sair.

Se a engenharia em geral gasta seu tempo em análise, documentação, projeto e uma série de bateria de testes, porque a engenharia de software deve ser diferente? É preciso mudar a mentalidade e cultura dos desenvolvedores e empresas de software a investirem mais nestes itens, não apenas para a melhora na qualidade do software, mas para melhorar o rendimento da equipe interna, de novos contratados, criação de novos produtos ao invés do foco reativo, diminuição da equipe de suporte técnico, consecutivamente dos custos da empresa, maior competitividade no mercado, consecutivamente um lucro maior.

Trocam tudo isso em nome de “maior produtividade”, o que é um erro, não aumenta a produtividade coisa nenhuma, só piora, é comprovado através de pesquisas que a grande maioria das empresas brasileiras gastam 70% do tempo de desenvolvimento corrigindo erros, por falta dos itens já mencionados. Onde está a produtividade? É uma ilusão pensar que economizando nestes itens haverá maior produtividade. Além do mais, basta utilizar ferramentas CASE que geram o código a partir da documentação. E vamos falar um pouco de ferramentas CASE.

Sobre Ferramentas CASE

Ferramentas CASE (do inglês Computer-Aided Software Engineering) é uma classificação que abrange todas as ferramentas baseada em computadores que auxiliam atividades de engenharia de software, desde análise de requisitos e modelagem até programação e testes. Podem ser consideradas como ferramentas automatizadas que tem como objetivo auxiliar o desenvolvedor de sistemas em uma ou várias etapas do ciclo de desenvolvimento de software.

Objetivos

- Melhoria da qualidade de software;

- Aumento da produtividade no processo de software.

Vantagens do uso de ferramentas CASE

- Qualidade no produto final;

- Produtividade;

- Agilizar o tempo para tomada de decisão;

- Menor quantidade de códigos de programação;

- Melhoria e redução de custos na manutenção;

- Agilidade no retrabalho do software.

Desvantagem do uso de ferramentas CASE

- Incompatibilidade de ferramentas;

- Treinamento para utilização.

- Uma ferramenta CASE que gosto bastante é o Enterprise Architect, que permite tudo isso e ainda gerar códigos em:

- ActionScript

- Ada

- C e C++

- C#

- Java

- Delphi

- Verilog

- PHP

- VHDL

- Python

- System C

- VB.Net

- Visual Basic

E mais…

Esta é apenas uma solução para documentação e análise de sistema, que permite aumentar a qualidade melhorando ainda mais a produtividade no desenvolvimento de software.

Muito foi falado de testes neste artigo, então apresento a seguir algumas ferramentas de testes open-source.

Ferramentas de Testes

Para automação de testes podemos citar as seguintes ferramentas open-source:

Selenium–Testes - Automatizados para Web por meios Funcionais e de Aceitação.

Testes de performance em aplicações de diferentes tipos de servidores (HTTP/HTTPS, SOAP, JMS, etc.).

Watir – Testes Automatizados para Web escritos na linguagem Ruby. Existem derivações em .Net (WatN) e Java (WatJ).