SUMÁRIO

Visão Geral de Qualidade de Software

Definição de Qualidade de Software, a qualidade pode ter diferentes interpretações, dependendo de quem a está avaliando. As diferenças de qualidade de produto e de processo de software e fatores relacionados à Qualidade de Software.

OBJETIVOS:

- Compreender as diferentes visões da Qualidade de Software;
- Compreender os fatores da qualidade de um software;
- Entender o que são métricas de software e;
- A importância das revisões de software.

Conceito de Qualidade de Software

Com a constante demanda gerada pela vida moderna, cada vez mais os computadores passam a integrar a rotina diária e a produção de software vem tendo um aumento constante. A exigência por qualidade estende-se também à área de software e pode ser considerada o centro das atenções para o desenvolvimento de software. Por exemplo, do ponto de vista dos fornecedores de software, qualidade não é mais um fator de vantagem no mercado, mas uma condição necessária e pode-se dizer indispensável para que seja possível competir com sucesso.

A era da Qualidade de Software.

Desde os tempos remotos, muitos problemas no desenvolvimento dos sistemas computacionais já se faziam sentir. Em 1968 o Comitê de Ciências da OTAN reuniu 50 especialistas, cientistas e profissionais da indústria de software para discutir possíveis soluções para o que passou a ser conhecido como a Crise do Software.

Nesse encontro se firmou o termo Engenharia de Software, e foi definida formalmente a necessidade da aplicação de uma abordagem sistemática, disciplinada e quantificável para o desenvolvimento, operação e manutenção de produtos de software.

A engenharia de software através de uma perspectiva histórica:

- Década de 60 e os anos que a antecedem: podem ser chamados de Era Funcional – quando se aprendeu a usar a tecnologia da informação para suprir as necessidades institucionais e começar a integrar o software nas operações diárias das instituições.
- Década de 70: ficou conhecida como a Era do Método - nessa fase, como as organizações de software foram caracterizadas por maciços atrasos nos planos e constantes ultrapassagens dos custos planejados, a maior preocupação era planejar e controlar os projetos de software. Foi quando os modelos de ciclo-de-vida, baseados em várias fases, foram introduzidos e analisados.

- Década de 80: foi a era do Custo - O custo do hardware começou a cair e a tecnologia da informação se tornou acessível às pessoas, não mais apenas às instituições. A competição das indústrias tomou um rumo diferente pois aplicações de baixo custo puderam ser largamente implementadas. A importância da produtividade no desenvolvimento de software aumentou significativamente. Nessa fase, vários modelos de custo na Engenharia de Software foram implementados e usados. Foi também no final dessa década que se reconheceu a importância da Qualidade de Software.
- Década de 90: Era da Qualidade. A década de 90 e os anos que seguem podem, certamente, ser chamados de Era da Qualidade. Com a tecnologia do estado da arte, espera-se atender a demanda dos clientes com a crescente exigência de alta qualidade.

Qualidade de Software: Visão Geral de Qualidade de Software

Qualidade é um termo que pode ter diferentes interpretações e para se estudar a Qualidade de Software de maneira efetiva é necessário, inicialmente, obter um consenso em relação à definição de Qualidade de Software que está sendo abordada.

Existem muitas definições de Qualidade de Software propostas na literatura, sob diferentes pontos de vistas, vejamos alguns:

- *Qualidade de Software é a conformidade a requisitos funcionais e de desempenho que foram explicitamente declarados, a padrões de desenvolvimento claramente documentados, e a características implícitas que são esperadas de todo software desenvolvido por profissionais" (Pressman,1994).*
- *"Um produto de software apresenta qualidade dependendo do grau de satisfação das necessidades dos clientes sob todos os aspectos do produto" (Sanders, 1994).*

- *"Qualidade é a totalidade de características e critérios de um produto ou serviço que exercem suas habilidades para satisfazer as necessidades declaradas ou envolvidas" (ISO9126 1994).*
- *"Qualidade é a totalidade das características de uma entidade, que lhe confere a capacidade de satisfazer necessidades explícitas e implícitas" (NBR ISO 8402, 1994).*

De um modo geral, Qualidade de Software pode ser definida como: Um conjunto de atributos de software que devem ser satisfeitos de modo que o software atenda às necessidades do usuário (seja ele um usuário final, um desenvolvedor ou uma organização), onde a determinação dos atributos relevantes para cada software varia em função:

- do domínio da aplicação;
- das tecnologias utilizadas;

- das características específicas do projeto;
- das necessidades do usuário e da organização.

Podemos dizer ainda que a qualidade depende também do ponto de vista de quem a avalia, onde usuários, desenvolvedores e organizações podem ter pontos de necessidades diferentes:

Usuário: avalia o software sem conhecer seus aspectos internos, está apenas interessado na facilidade do uso, no desempenho, na confiabilidade dos resultados e no preço;

Desenvolvedores: avaliam aspectos de conformidade em relação aos requisitos dos clientes e também aspectos internos do software;

Organização: avalia aspectos de conformidade em relação aos requisitos dos clientes e desenvolvedores e também aspectos de custo e cronograma.

Qualidade de Produto X Qualidade de Processo de Software

- *Qualidade de Software pode ser vista como um conjunto de características que devem ser alcançadas em um determinado grau para que o produto atenda às necessidades de seus usuários (Rocha et al.,2001).*

A qualidade do produto final é profundamente afetada pela qualidade do processo de desenvolvimento, portanto a qualidade deve ser uma meta a ser alcançada e aprimorada ao longo do processo de software.

Aqui se faz importante definirmos o que vem a ser um produto de software, assim como esclarecermos alguns pontos de um processo de software, vamos lá?

Produto de Software

> - *Um produto de software compreende os programas e procedimentos de computador e a documentação e dados associados, que foram projetados para serem liberados para o usuário (ISO/IEC 12207-1, 1995).*

Da mesma forma como existem diversas interpretações para qualidade de um modo geral, também existem diversas interpretações para qualidade de um produto de software, tais como:

- Boa fabricação;
- Deve durar muito;
- Bom desempenho;
- Adaptável às minhas necessidades específicas;
- Fácil de usar;
- Sem defeitos, entre outros.

A especificação de Qualidade de Produto de Software deve ser mais precisa e detalhada. A formalização de Qualidade de Produto de Software pode ser feita usando-se um Modelo de Qualidade de Produto de Software.

Como mesmo as proposições bem-sucedidas trazem dificuldades de aplicação, por causa dos muitos aspectos de qualidade oferecidos, surgiu a necessidade de um modelo padronizado.

Por essa razão o comitê técnico da ISO/IEC começou a trabalhar para desenvolver o consenso requerido e encorajar a padronização em nível mundial. As primeiras tentativas de padronização surgiram em 1978.

Em 1985 foi iniciado o desenvolvimento da Norma Internacional ISO/IEC 9126: “Information Technology – Software product evaluation – Quality characteristics and guidelines for thei use” - publicada em 1991.

A Norma NBR 13596 é uma tradução da Norma ISO/IEC 9126. Foi elaborada pela comissão de Estudos de Qualidade de Software do Subcomitê de Software do Comitê de Informática da ABNT – Associação Brasileira de Normas Técnicas e publicada em agosto de 1996. A norma avalia a qualidade de um produto de software através de um conjunto de características.

Processo de Software

Algumas definições da palavra processo:

- Um processo é “a maneira pela qual se realiza uma operação, segundo determinadas normas” (dicionário Aurélio)
- Um processo é uma sequência de passos realizados para um dado propósito” (IEEE)
- O processo integra pessoas, ferramentas e procedimentos.

- Processo é o que as pessoas fazem, usando procedimentos, métodos, ferramentas e equipamentos, para transformar matéria prima (entrada) em um produto (saída) que tem valor para o cliente.

Um processo de software pode ser definido como um conjunto de atividades, métodos, práticas e transformações que as pessoas usam para desenvolver e manter o software e os produtos associados (por exemplo: planos de projeto, documentos, código, casos de teste e manuais de usuário) (IEEE-STD-610).

Um processo de software envolve um grande conjunto de elementos, tais como objetivos organizacionais, políticas, pessoas, comprometimentos, ferramentas, métodos, atividades de apoio e as tarefas da engenharia de software.

Para que o processo de software seja eficiente ele precisa ser constantemente avaliado, medido e controlado. Quando as empresas não possuem conhecimento suficiente de todos os elementos do processo, essas atividades de avaliar, medir e controlar ficam

difíceis de serem realizadas, nesse caso o processo de software passa a ser descontrolado, sem gerência e até mesmo caótico. Algumas características de processos de softwares caóticos são (Pressman, 2002).

- O processo é improvisado (profissionais e gerentes);
- O processo não é rigorosamente seguido e o cumprimento do mesmo não é controlado;
- O processo é altamente dependente dos profissionais atuais;
- A visão do progresso e da qualidade do processo é baixa;
- A qualidade do produto decorrente do processo é comprometida em função de prazos;
- Quando são impostas datas urgentes para entrega dos produtos, frequentemente, a funcionalidade e a qualidade dos mesmos são comprometidas para atender o cronograma;

- Não existe nenhuma base objetiva para julgar a qualidade do produto ou para resolver problemas de processo ou de produto, portanto, a qualidade do produto é difícil de ser prevista;
- As atividades ligadas à melhoria da qualidade, tais como revisões e testes, frequentemente são encurtadas ou eliminadas quando os projetos ultrapassam o cronograma previsto.

Já quando as coisas caminham bem e o processo é controlado e gerenciado com eficiência o processo passa a ser bom, passa a ser maduro e eficiente, gerando resultados positivos, tais como:

- O processo continua a despeito de problemas inesperados (Robustez).
- Rapidez na produção do sistema (Velocidade).
- O processo é aceito por todos os envolvidos nele (Aceitabilidade).

- Os erros do processo são descobertos antes que resultem em erros no produto (Confiabilidade).
- O processo evolui para atender alterações de necessidades organizacionais (Manutenibilidade).
- O processo é compreendido (usualmente através de documentação e de treinamento), utilizado, vivo e ativo.
- O processo é bem controlado – a fidelidade ao processo é objeto de auditoria e de controle.
- Medidas do produto e do processo são utilizadas.
- Os papéis e responsabilidades no processo estão claros ao longo de todo o projeto e por toda a organização.
- A produtividade e a qualidade dos produtos de software resultantes podem ser melhoradas com o tempo através de ganhos consistentes na disciplina obtida pelo uso do processo de software.

- Os gerentes monitoram a qualidade dos produtos de software e a satisfação do cliente.
- Existe uma base quantitativa, objetiva para julgar a qualidade dos produtos e analisar problemas com o produto e o processo.
- A organização com processo de software de alta qualidade tem esse processo institucionalizado através de políticas, padrões e estruturas organizacionais.

Fica claro que para se alcançar os objetivos de produtividade e qualidade é necessário que o processo de software seja eficiente, definido, gerenciado, medido e controlado.

Fatores de Qualidade de Software

Existem dois tipos de Qualidade de Software: um tipo de qualidade com a qual o usuário do programa interage- essa é a qualidade externa. E um tipo de qualidade com a qual outros desenvolvedores interagem - essa é a qualidade interna, sendo assim podemos dizer te temos os fatores de qualidade interno e os fatores de qualidade externo (Pressman 2002).

Fatores externos – ponto de vista do usuário:

- Correção: característica do software que realiza as tarefas como foram definidas em sua especificação de requisitos.
- Robustez: um software é robusto se realiza as suas tarefas de forma correta mesmo quando submetido a condições anormais.

- Extensibilidade: característica de um software poder ser facilmente adaptado a inclusões e alterações de requisitos.
- Reusabilidade: característica de um software que pode ser reutilizado ao todo ou em parte por outros softwares.
- Compatibilidade: facilidade de se combinar o software com outros softwares. Essa característica é importante porque raramente um software é construído sem interação com outros softwares.
- Eficiência: refere-se ao bom uso que o software faz dos recursos de hardware, tais como memória e processadores.
- Portabilidade: é a facilidade de se utilizar o software em diferentes ambientes de hardware e software.
- Verificabilidade: é a facilidade de se preparar rotinas para se verificar a conformidade do software com os seus requisitos.

- Integridade: é uma característica relacionada à segurança de dados, programas e documentos. Integridade é a habilidade de proteger tais componentes contra acessos não autorizados.
- Facilidade de uso: também denominada usabilidade, é a facilidade com que o software pode ser aprendido e utilizado.

Fatores internos: ponto de vista estrutural do software. Permitem atingir os fatores externos:

- Modularidade: característica de um software que é constituído por unidades denominadas módulos.
- Legibilidade:
- Manutenibilidade: facilidade de realizar manutenção em um software.

A forma com um software é construído permite atingir os fatores internos de qualidade.

Os fatores internos de qualidade permitem atingir os fatores externos de qualidade.

Métricas de Software

Segundo Tom De Marco,

- *"Não se pode gerenciar o que não se pode medir".*

Roger Pressman, 2002, considera que,

- *"Se você não sabe para onde você quer ir, qualquer caminho você pode seguir. Se você não sabe onde você está, um mapa, não vai ajudar!".*

Ainda, segundo Pressman, uma métrica é a medição de um atributo (propriedades ou características) de uma determinada entidade (produto, processo ou recursos):

- Tamanho do produto de software (ex: Número de Linhas de código)
- Número de pessoas necessárias para implementar um caso de uso
- Número de defeitos encontrados por fase de desenvolvimento
- Esforço para a realização de uma tarefa

- Tempo para a realização de uma tarefa
- Custo para a realização de uma tarefa
- Grau de satisfação do cliente (ex: adequação do produto ao propósito, conformidade do produto com a especificação)

Por que devemos medir o software? É realmente importante ou seria uma perda de tempo?

Algumas vantagens de realizarmos essas medições no software:

- Entender e aperfeiçoar o processo de desenvolvimento
- Melhorar a gerência de projetos e o relacionamento com clientes
- Reduzir frustrações e pressões de cronograma
- Gerenciar contratos de software
- Indicar a qualidade de um produto de software

- Avaliar a produtividade do processo
- Avaliar os benefícios (em termos de produtividade e qualidade) de novos métodos e ferramentas de engenharia de software
- Avaliar retorno de investimento
- Identificar as melhores práticas de desenvolvimento de software
- Embasar solicitações de novas ferramentas e treinamento
- Avaliar o impacto da variação de um ou mais atributos do produto ou do processo na qualidade e/ou produtividade
- Formar uma baseline para estimativas
- Melhorar a exatidão das estimativas
- Oferecer dados qualitativos e quantitativos ao gerenciamento de desenvolvimento de software, de forma a realizar melhorias em todo o processo de desenvolvimento de software

Para realizar as medições precisa-se fazer uso de alguma ou algumas métricas, a qual deve ser válida (quantifica o que quer se medir), confiável (produz os mesmos resultados dadas as mesmas condições) e prática (barata, fácil de computar e fácil de interpretar).

Categorização de métricas

- Métricas diretas (fundamentais ou básicas): medida realizada em termos de atributos observados (usualmente determinada pela contagem). Ex.: custo, esforço, no. linhas de código, capacidade de memória, números de páginas, números diagramas, etc.
- Métricas indiretas (derivadas): medidas obtidas a partir de outras métricas. Ex.: complexidade, eficiência, confiabilidade, facilidade de manutenção.
- Métricas orientadas a tamanho: são medidas diretas do tamanho dos artefatos de software associados ao processo por meio do qual o software é

desenvolvido. Ex.: esforço, custo, números de KLOC, números de páginas de documentação, números de Erros.

- Métricas orientadas por função: consiste em um método para medição de software do ponto de vista do usuário, determinando de forma consistente o tamanho e a complexidade de um software.
- Métricas de produtividade: concentram-se na saída do processo de engenharia de software. Ex.: números de casos de uso/iteração.
- Métricas de qualidade: Oferecem uma indicação de quanto o software se adequa às exigências implícitas e explícitas do cliente.Ex.: erros/fase .
- Métricas técnicas: concentram-se nas características do software e não no processo por meio do qual o software foi desenvolvido. Ex.: complexidade lógica e grau de manutenibilidade.

Revisões de Software

De acordo com SEI CMM (3-1.5, 1988),

"um processo de revisão pode ser definido como uma avaliação crítica de um objeto (...) assim walkthroughs, inspeções e auditorias podem ser visualizados como formas de processos de revisão."

As revisões são métodos de validação de qualidade de um processo ou produto amplamente usado pela equipe técnica do projeto. São consideradas como verdadeiros "filtros" de erros e inconsistências no processo de desenvolvimento de software.

A atividade de revisão começou como uma ferramenta de controle gerencial – Revisão de progresso, onde o progresso não pode ser avaliado simplesmente contando-se o número de tarefas finalizadas.

Era preciso estabelecer um meio de avaliar também a qualidade do trabalho executado, surgiram então as revisões que avaliam aspectos técnicos do produto. Qualquer produto pode ser submetido a uma revisão aplicada desde as primeiras fases do ciclo de vida. As revisões devem ser formais e também informais.

Deve sempre ser realizado um planejamento para a realização das revisões de software.

Cabe ao engenheiro de software planejar:

- o que deve ser revisado
- quais os resultados esperados
- quem deve fazer a revisão,
- determinar “checkpoints” dentro do ciclo de vida onde a revisão deve ser aplicada,
- determinar resultados esperados.

Algumas informações importantes no planejamento das revisões são:

- quem participa?
- qual informação é requerida antes da revisão?
- quais pré-condições que devem ser satisfeitas antes que a revisão possa ser conduzida?
- Como Organizar?
- Gerar checklists ou outra indicação do que deve ser coberto na revisão;
- Determinar as condições de término ou critérios que devem ser satisfeitos para que a revisão termine;
- Gerar registros e documentos que devem ser produzidos.

As revisões são o principal mecanismo para avaliar o progresso do desenvolvimento de maneira confiável, trazendo vários benefícios para o bom desenvolvimento do software, tais como:

- As revisões trazem à luz as capacidades de cada indivíduo envolvido no desenvolvimento,
- revisões são capazes de revelar lotes ou classes de erros de uma só vez;
- revisões proporcionam retorno já nas primeiras fases, prevenindo que erros mais sérios surjam;
- revisões treinam e educam os participantes e
- têm significante efeito positivo na competência dos desenvolvedores.

Quando se realiza revisões e correções desde o início do ciclo de desenvolvimento de software, um dos grandes ganhos é com relação aos custos. Vejamos o custo da

remoção de erros: Atividades de projeto são responsáveis por 50 a 65% dos erros, a revisão pode revelar até 75% desses erros e, revelar erros cedo diminui o custo de validação e correção.

- Fase de projeto: custo 1
- Fase anterior ao teste: custo 6.5
- Fase de teste: custo 15
- Fase de manutenção: custo 60 a 100

- *"Uma má revisão pode ser pior do que nenhuma revisão",*

por isso 'e importante considerar as atividades a seguir:

- Determine uma agenda (e mantenha-a);
- Limite os debates;

- Levante as áreas problemáticas – não tente resolver todos os problemas;
- Tome notas;
- Revise o produto, não o produtor;
- Limite o número de participantes e insista na preparação;
- Prepare um checklist, de acordo com o produto a ser revisado;
- Reserve recursos do projeto para revisões;
- Promova treinamento para os revisores;
- Revise suas antigas revisões.

REFLEXÃO:

Conhecer um pouco mais sobre o assunto Qualidade de Software, entendendo a sua importância. A Qualidade de Software pode ser subdividida em qualidade de produto e qualidade de processo, sendo as duas extremamente importantes e relacionadas.

Dizer que um software tem qualidade é necessário que o mesmo seja avaliado, o que pode ser realizado com o auxílio das métricas e constantes revisões.

ATIVIDADE (EM GRUPO)

- **Elaborar Apresentação por grupo**
- **Responder as questões**

1. Defina Qualidade de Software.
2. Defina processo de software e comente as características de um bom processo de software.
3. Comente sobre a categorização das métricas.
4. Quando falamos de revisões de software, o que é importante que o engenheiro considere no planejamento?

REFERÊNCIAS

Métricas e Estimativas de Software – O início de um rally de regularidade. Url: http:// www.apinfo.com/artigo44.htm

PRESSMAN, Roger S. **Engenharia de Software**. Rio de Janeiro: MacGraw Hill, 2002.

ROCHA, ANA REGINA CAVALCANTI; MALDONADO, JOSÉ CARLOS; WEBER, KIVAL CHAVES. **Qualidade de Software – Teoria e Prática**. 1ª edição. São Paulo: Prentice Hall, 2001.

SANCHES, ROSELY. **Material Didático:** Qualidade de Software. ICMC-USP, 2003.

SANCHES, ROSELY; FABBRI, SANDRA PINTO; MALDONADO, JOSÉ CARLOS. **Qualidade de Software:** da engenharia de software aos modelos de qualidade. VI Escola Regional de Informática, SBC, 2003.

SOMERVILLE, IAN. **Engenharia de Software**. 6ª edição. São Paulo: Addison Wesley, 2003.