

Requisitos de Métodos de Garantia da Qualidade no Desenvolvimento de Softwares

Damázio Pereira Teixeira

Departamento de Ciência da Computação – Universidade Federal de Minas Gerais
(UFMG)
Pampulha – 31.270-901 – Minas Gerais – MG – Brasil

damazio@dcc.ufmg.br

***Resumo.** Este documento proporciona um catálogo de requisitos para auxiliar e controlar a qualidade no desenvolvimento de software, onde métricas e padrões são recomendados com o intuito de estimar o nível de qualidade do desenvolvimento de software. Esta metodologia permite a Gerentes de Projetos, ou responsáveis envolvidos em processos de desenvolvimento de software, adotar critérios e personalizar atividades que englobam informações implícitas e explícitas contidas na análise de requisitos do projeto. Este documento irá produzir insumo que servirá de base para mensurar tanto o processo quanto o produto através das linhas de segmento no que se diz respeito à qualidade.*

1. Introdução

Segundo Bauer (1972), a engenharia de software é "O estabelecimento e uso de sólidos princípios de engenharia para que se possam obter softwares economicamente viáveis, que sejam confiáveis e que funcionem eficientemente em máquinas reais". Partindo deste conceito, vimos que muito de qualidade está implícito, pois confiabilidade e eficiência são preceitos deste fator. Temos a qualidade como grande motivador e diferencial em todas as áreas de atividade humana.

Pode-se também inferir que qualidade é um conjunto de atributos de software que devem ser satisfeitos de modo que o software atenda às necessidades dos usuários. Por outro lado, segundo Pressman (1994), "Qualidade de software é a conformidade de 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".

Os Métodos de Garantia da Qualidade no desenvolvimento de Softwares começam no momento das representações, sejam elas executáveis ou não. "A garantia de qualidade de software é uma atividade que deve ser aplicada e gerenciada ao longo de todo o processo de desenvolvimento; envolvendo revisões técnicas formais, múltiplas fases de teste, controle da documentação de software e das mudanças nos procedimentos para garantir a adequação aos padrões e mecanismos de medição e divulgação". Pressman (1995).

A qualidade no desenvolvimento de software está ligada diretamente nos conceitos de avaliação de processo sendo tratadas por alguns padrões e modelos consolidados e que são referência no que tange este assunto. Alguns deles, bem conceituados e de grande adesão são: o projeto SPICE (Software Process Improvement

and Capability Determination) através da ISO/IEC 15504 [RB013] [RB014], o modelo CMMI (Capability Maturity Model Integration) [RB008] [RB011] [RB015], Norma ISO/IEC 12207 [RB012] e uma série de normas ISO 9000 [RB016]. Dentre as linhas que garantem a qualidade no desenvolvimento de software, cita-se: funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade. Através destas linhas podem-se definir métricas para atividades que acompanharão o desenvolvimento ao longo do projeto.

Este documento tem o objetivo de reunir um conjunto de requisitos de qualidade no desenvolvimento de software, onde todos os elementos participantes tais como Hardware, Software, Redes, Bancos de Dados, Procedimentos, Pessoas e Documentos tem uma atenção especial em cada fase do desenvolvimento do projeto de software. A metodologia aqui descrita destina-se a Gerentes de Projetos, responsáveis envolvidos em processos de desenvolvimento de software e ou participantes de comunidades correlatas com este tipo de processo.

2. Requisitos de Qualidade

Quando um software satisfaz sua especificação e cumpre os objetivos esperados pelo cliente, temos um forte indício de qualidade do software, porém não podemos garantir a qualidade no desenvolvimento. O desenvolvimento de software é um conceito muito mais amplo do que o produto final, pois é neste momento que os requisitos implícitos são tratados.

Kano et al [RB017] classificam os requisitos de acordo com duas dimensões, conforme ilustrado no diagrama da Figura 1 em que "atendimento ao requisito" é colocado em um eixo e o "sentimento de satisfação" no outro. Segundo essa análise, os requisitos de qualidade se classificam como:

- **necessários**, se o não atendimento gera insatisfação e o atendimento gera indiferença. Por exemplo, a não ocorrência de problemas e erros na usabilidade do software durante o trabalho.
- **normais**, se há insatisfação pelo não atendimento e satisfação pelo atendimento. Por exemplo, uma ferramenta que permita uma redução de rotinas e passos para se concluir algo a ser executado.
- **atrativos**, se provocam um sentimento de satisfação quando atendidos mas indiferença se não forem implementados. Por exemplo, funcionalidades extras de comparativos com serviços anteriores, exportadores, designer agradável, entre outros.

Deve-se notar que a curva de satisfação é dinâmica e pessoal, variando-se por regiões, mercado e nível de conhecimento sobre a área do conhecimento que o software atua. A análise desses requisitos e sua relevância devem ser acompanhadas em cada projeto junto ao cliente, e deve ser avaliada durante todo o processo de desenvolvimento de software.

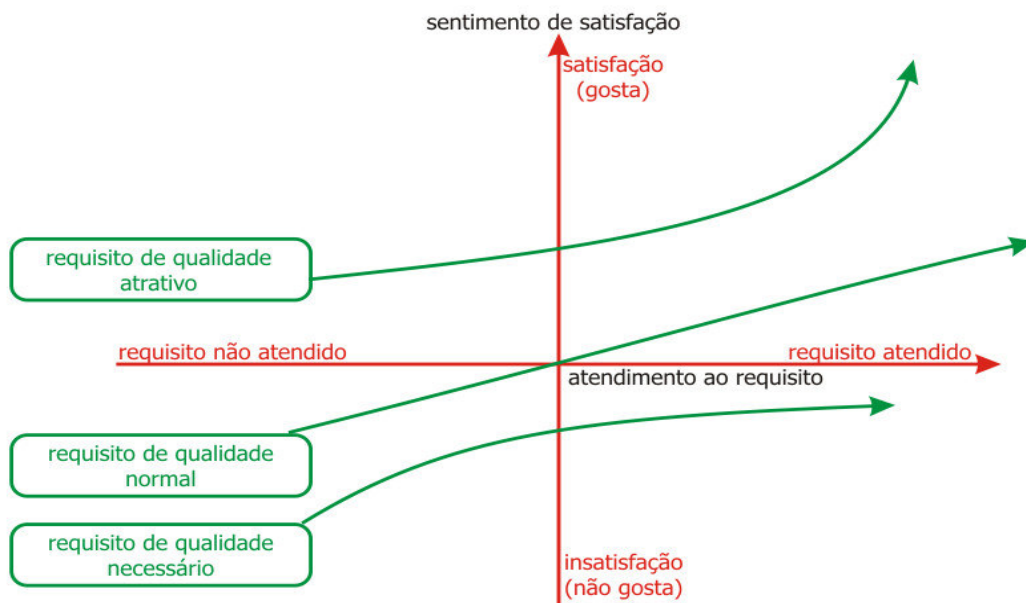


Figura 1 – Requisitos de Qualidade

Nesta seção estão listados alguns dos principais requisitos funcionais ou não, analisados diretamente com seus elementos ativos. Estes requisitos aqui descritos são conceitos e reflexões retirados a partir de estudos bibliográficos dos requisitos de Qualidade McCall e HP.

A Figura 2 mostra modelo de qualidade McCall, onde a qualidade está relacionada a três áreas distintas, sendo elas: operação, transição e revisão.



Figura 2 – modelo de qualidade McCall

Na área de operação, onde podemos dizer que o software será avaliado pelo cliente e também afirmar que a avaliação se dá através da curva de requisito normal de qualidade, esta se traduz através dos seguintes requisitos implícitos:

- **corretitude**, que se traduz em satisfazer a especificação e cumprir os objetivos solicitados pelo cliente. Ou seja, atender ao máximo a solicitação realizada pelo cliente onde ele tenta passar toda sua necessidade de software e facilidade que gostaria de ter quanto a sua utilização.

- **confiabilidade**, que se traduz em o programa executar a função esperada com a precisão exigida. Ou seja, o software deverá capaz de realizar as tarefas e atividades esperadas pelo usuário, mas retornando sempre uma informação condizente ao provável esperado, onde estas saídas serão de fundamental importância para tomadas de decisões.

- **eficiência**, que seria uma quantidade mínima de recursos necessários para se atingir um objetivo. Ou seja, envolve-se neste aspecto requisitos mínimos de hardware e grande facilidade de aprendizado e uso.

- **integridade**, que é a garantia que o software somente será utilizado por pessoas autorizadas e sua utilização pode ser controlada. Ou seja, em muitos casos onde a necessidade se dá através de softwares de pequenos portes é desprezível, mas no contrário onde o software será utilizado por muitas pessoas e realizar tarefas críticas é preciso inibir a utilização total ou em parte de pessoas não autorizadas e controlar o que foi feito e por quem para uma auditoria mais precisa da qualidade.

- **usabilidade**, o menor esforço por parte do cliente em aprender, utilizar e transformar trabalho em produção através do software produzido. Este requisito já comentado em eficiência, se torna independente e fundamental quando o software irá ser utilizado por muitas pessoas de níveis distintos de conhecimento, sendo preciso ter um alto grau de facilidade no aprendizado e utilização.

Na área de revisão, que foca mudanças no software, qualidade está definida através dos requisitos:

- **manutenibilidade**, significa o mínimo de esforço para localizar e corrigir um erro ou inconsistência. Através deste requisito fica claro a importância que se tem em trabalhar e planejar testes e inspeções de código periodicamente a fim de evitar re-trabalho e adaptações forçadas no software.

- **flexibilidade**, o menor esforço para alterar uma funcionalidade ou regra de negócio do sistema. Este requisito está muito ligado ao escopo do projeto que deve permitir mudanças “previstas” e “imprevistas” ao longo do desenvolvimento do software. Para atender este requisito, um trabalho de documentação deve ser muito forte e um controle de todo projeto é preciso em mãos para permitir de forma simples a mudança ou implementação de novas tarefas e ou atividades no software.

- **testabilidade**, o menor esforço para testar o programa para garantir que ele atende aos requisitos explícitos. Dentre muitas técnicas de testes, sendo algumas delas conhecidas como Caixa Branca, Caixa Preta, de Unidade, de Aceitação, de Operação entre outras são fundamentais para garantir a qualidade do software. Em especial, para a qualidade de software, este é um requisito imprescindível que deve ter uma correlação direta durante todo o desenvolvimento do mesmo, fazendo-se presente em todos os processos tentando evitar o mínimo de não coesão dos requisitos com o que foi proposto e o menor índice de erros do software.

Na área de transição, que trata a mudança de ambiente de um software sendo esta em questões de hardware / sistemas operacionais / bancos de dados / ou ambiente hospedado e segmentos correlatos, qualidade está definida como:

- **portabilidade**, a independência e autonomia de um software em relação à hardware, softwares de apoio como banco de dados e ou sistemas operacionais. O software deve visar sua independência de plataformas e bases para se atingir uma autonomia. Esta autonomia é conseguida utilizando-se os padrões de desenvolvimento de software hoje praticados no mercado que separam em camadas a inteligência do software com a comunicação física e lógica do software.

- **reusabilidade**, quando um programa possui de forma modular partes de suas funcionalidades que podem ser utilizadas ou copiadas para outros sistemas. Esta reusabilidade ganha importância pois não permite além de retrabalho a duplicação de códigos. Se este requisito for atendido com o apoio de uma boa documentação, é provável que um integrante novato em uma equipe consiga se adaptar de maneira muito mais rápida aos processos utilizados pelo software.

- **interoperabilidade**, seria o mínimo de esforço exigido para acoplar / integrar um sistema a outro. Ou seja, caso seja de necessidade a comunicação de ambos os lados para uma troca de informação, este requisito se faz presente, útil e imprescindível. Algumas situações como validação de cartão de crédito, envio de fatura ao banco, etc são exemplos de interoperabilidade que devem sempre ser avaliada e bem especificada além de possuir ao seu entorno um forte sistema de segurança.

Incorporado nestas 3 áreas, está o modelo FURPS (Functionality, Usability, Reliability, Performance, Supportability). Este modelo descreve a funcionalidade como um conjunto de características distintas do programa, generalidade e multiusabilidade das funções, segurança e controle do sistema em geral [RB019].

- **Usabilidade** seria atender a recursos visuais para facilitar o usuário, permitindo um maior grau de aprendizagem, assimilação de ferramentas, estética visual, consistência de informações e documentação simplificada, eficiente e coesa.

- **Confiabilidade** está em ter o mínimo de inconsistências, menor frequência de falhas e o não impacto destas falhas no serviço executado, entre outros.

- **Desempenho** que é um requisito na maioria das vezes implícito, está ligado ao desempenho de processamento e resposta, consumo de recursos e estabilidade.

- **Supportabilidade** está na capacidade de ampliar o programa, adaptabilidade, compatibilidade, e facilidade de instalação e treinamento do programa.

Avaliando detalhadamente alguns pontos-chaves, temos a:

- **Modularidade**: para a garantia de qualidade no desenvolvimento de software, uma das grandes armas é a modularidade de funções. É interessante que cada função seja bem definida e descrita, além de possuir documentação e comentários de código para que demais desenvolvedores possam utilizar e adaptar seu software de acordo com o padrão adotado. Quando se faz necessário a criação de uma função, esta deve ser pensada de forma mais genérica, visando assim possibilitar sua utilização em outras partes do software. Em último caso, esta funcionalidade modularizada poderá sofrer alguns tipos de especializações, e nestes casos, o empacotamento destas funções de escopo similar deverão estar sempre juntas evitando um retrabalho, ambiguidade e duplicidade de código.

- **Documentação:** outro fator que está impactando diretamente na qualidade do desenvolvimento de software, comentado anteriormente é a documentação. Uma documentação bem feita, relatando detalhadamente o que cada funcionalidade representa, o que faz, quais são seus estados e ações, o que ela recebe como parâmetro e se retorna algum tipo de informação. Esta documentação deve ser especificada a nível de desenvolvedores, possibilitando um auto aprendizado quando lido por outros desenvolvedores. Além da documentação produzida, comentários bem resolvidos em linhas de código são de grande valia e se traduzem em qualidade, pois possibilitam um melhor acompanhamento do que acontece diretamente no software. Um comentário bem descrito pode reduzir consideravelmente a manutenibilidade e a testabilidade do software.

- **Rastreabilidade:** este requisito descreve a capacidade de rastrear uma representação de projeto ou componente de programa até os requisitos. Ou seja, um requisito definido pelo cliente deve ser facilmente encontrado nos diagramas do software e ter uma correlação destes diagramas com o código. Além disso, recomenda-se que este requisito seja perceptível ao cliente durante a utilização do sistema.

- **Responsabilidade:** é vital definir um responsável por um projeto que irá liderar uma equipe sendo ela pequena, média ou grande. Este responsável deverá ter o controle do andamento do projeto e deverá garantir que o software está atendendo os prazos e requisitos especificados. Além destas atribuições, o responsável pelo projeto deverá delegar responsabilidades aos participantes do projeto especificando o que será desenvolvido e o prazo para conclusão. Este responsável irá gerir o projeto de acordo com as métricas sugeridas na próxima seção. É atribuição deste responsável, tomar medidas para prevenir ou remediar situações de atraso no desenvolvimento do software.

3. Métricas de Qualidade no Desenvolvimento

Nesta seção algumas métricas serão avaliadas, e algumas formas serão definidas em conformidade com sua utilização e controle.

Na visão de Fletcher Buckley [RB009], ainda em 1979, os padrões de qualidade são heranças de conceitos militares que são aplicadas também ao desenvolvimento de software. Dentre as seções sugeridas por Buckley, temos a proposta, documentação, gerência, padrões, práticas e convenções, revisões e auditorias, gerência de configuração, levantamento de problemas e tomada de ações correlatas, controle de código, ferramentas, mídias, insumos, técnicas e metodologias. Em suas palavras, garantia de qualidade é “Um planejamento é um padrão sistemático de todas as ações necessárias para prover confiança adequada ao item ou produto de acordo com as exigências técnicas estabelecidas”.

Para Buckley, a qualidade estava diretamente ligada ao que o usuário espera do software produzido, porém de acordo com Pressman a visão deve ser mais generalizada visando além da qualidade explícita uma mescla com a qualidade implícita. Para alguns tópicos, uma breve descrição será mais detalhada com o intuito de abranger maior o foco dos conceitos aplicando-os à prática.

Diante deste cenário, a dificuldade está em descobrir o que deve ser controlado, gerido e auditado para que o desenvolvimento de software possua qualidade. Diante deste impasse, apenas uma certeza podemos ter em relação ao contexto, que será a qualidade do software caso as métricas que acompanham o desenvolvimento de software indiquem qualidade esperada ou superior. Uma forma simplista seria cada

gerente, de acordo com sua experiência adquirida, estruturar uma espécie de tabela que através dos requisitos salientados irá definir pontuações para níveis de qualidade e diante deste insumo fazer avaliações com a realidade do projeto.

O uso de uma lista de requisitos, ou checklist – Tabela 1, pode ser eficiente em projetos menores, onde o gerente irá fazer uma auditoria homem a homem, acompanhando sua forma de codificação e sua tendência em adotar os padrões e especificações acertadas para o projeto. A partir desta análise, alguns pontos anteriormente citados devem ser avaliados, sendo eles a forma de documentação e comentário de código realizadas avaliando a pertinência e significado do texto utilizado. A possibilidade de estar utilizando funcionalidades genéricas reduzindo ao máximo especializações destas. Através da rastreabilidade podemos também, por exemplo, acompanhar em quais e quantas partes do código uma mesma função é utilizada, onde quanto maior esse número maior a qualidade do código.

Requisitos	Categoria			Nível de Atendimento 0 - 5
	Necessário	Normal	Atrativo	
Confiabilidade	X			
Eficiência	X			
Flexibilidade			X	
Eficiência			X	
Reusabilidade		X		
Desempenho		X		
Modularidade	X			
Documentação		X		

Tabela 1 – Exemplo de CheckList em Projetos Menores

Para projetos maiores, o acompanhamento do software não deve ser tão minucioso para evitar atrasos. Neste caso deve-se definir bem a equipe a participar do projeto, garantindo um alto nível de domínio e conhecimento sobre a forma de trabalho. O gerente deve indicar coordenadores responsáveis por fazer esta avaliação e reportar somente informações relevantes para o acompanhamento geral. É preciso ainda treinar e definir bem o que cada um deverá fazer e especificar de forma simples e eficaz o papel de cada um no projeto. Um ponto relevante também será definir a forma como serão tratadas as mudanças de escopo e a integração de novos participantes ao projeto quando este já tiver sido iniciado [RB010].

Para estes projetos, uma característica importante é que se o produto final não atende o nível de qualidade que o cliente espera, o custo geralmente pode ser muito alto para reparar a falha do que reconstruir este produto novamente. O desenvolvimento de software não possui passos intermediários que possam ser verificáveis e validados durante o seu ciclo de vida, sendo necessário chegar ao fim do planejamento para realizar tais atividades. Estas porém, a cada novo ciclo devem ser novamente validadas perante um ponto mais genérico e global do software.

Para tentar se adaptar a diversidade e dinamismo que o software enfrenta, onde a cada nova apresentação do produto aos cliente novos requisitos, restrições e modelos são solicitados, as metodologias de gerenciamento de projetos ágeis estão sendo estudadas, difundidas e adotadas. A metodologia ágil é fundamentada nos padrões RUP e PMBOK [RB021], porém com visões mais simplistas e versáteis para seu

gerenciamento e utilização, evitando grandes burocracias e restrições que inviabilizam um desenvolvimento rápido do software.

Um detalhe importante é que requisitos detalhados não são o escopo, e que são tratados todo e qualquer escopo inicial como preliminar. Toda metodologia tradicional prega que o escopo é um conjunto de objetivos detalhados de alto nível do projeto e não requisitos detalhados. Juntamente com objetivos, o escopo também são riscos, restrições, premissas e somente uma lista dos requisitos mais importantes sem detalhamento.

4. Gestão da Qualidade no Desenvolvimento

Nesta seção, serão discutidas as formas sugeridas para avaliações, planejamentos, gerenciamento, monitoramento, controle e melhoria dos processos de desenvolvimento, operação, evolução e suporte de software.

Estes conceitos podem ser distintos, onde as empresas definem seus próprios modelos de gestão da qualidade, dependendo do seu tipo de negócio e de suas características. Dentre as diretrizes propostas na ISO 9000-3, a gestão da qualidade deve compreender o entendimento comum entre as partes envolvidas (contratante e contratado) para requisitos funcionais e o uso de metodologias consistentes para o desenvolvimento de software e gerenciamento do projeto como um todo, da concepção até a manutenção. Estas diretrizes são divididas em 3 partes:

- **Estrutura:** descreve aspectos organizacionais, relacionados ao sistema de qualidade. São detalhadas as responsabilidades e ações relacionadas à qualidade que devem ser tomadas tanto pelo fornecedor quanto pelo comprador. Os pontos abordados são: responsabilidade da administração, sistema de qualidade, auditorias internas do sistema de qualidade e ação corretiva.
- **Atividades do ciclo de vida:** descreve as atividades de desenvolvimento de software. A Norma define que o desenvolvimento de software deve ser feito segundo um determinado modelo de ciclo de vida, e as atividades relacionadas à qualidade devem ser planejadas e implementadas de acordo com a natureza deste modelo. Independentemente do modelo de ciclo de vida estabelecido pela organização, a Norma define que as atividades do ciclo de vida devem ser agrupadas em nove categorias: análise crítica do contrato; especificação dos requisitos do comprador; planejamento do desenvolvimento; planejamento da qualidade; projeto e implementação; ensaios e validação; aceitação; cópia, entrega e instalação; e manutenção.
- **Atividades de suporte:** descreve atividades que apóiam as atividades do ciclo de vida de desenvolvimento. Estão organizadas em nove itens: gestão de configuração; controle de documentos; registros da qualidade, medição; regras, práticas e convenções; ferramentas e técnicas; aquisição; produto de software incluído; e treinamento.

Estas diretrizes podem ser adotadas para a elaboração do escopo de acompanhamento do projeto, onde as atividades fins estarão discriminadas com seu tempo, seu responsável e sua descrição. Através destas atividades, que compreendem desde codificação de funcionalidades até auditoria e inspeção do desenvolvimento em si, o gerente irá gerir de forma ampla todo o andamento do projeto. Estas atividades devem ser geridas de forma macro focando o tempo gasto para a ação esperada e através de reuniões formatadas e reuniões formais, o gerente deverá se encontrar com líderes

dos projetos que irão roteirizar como está o andamento do projeto. Nestas reuniões deverão ser relatados os fatores positivos e negativos, os problemas e soluções tomados para documentação de ações corretivas e preventivas, o desempenho e melhoria por parte dos envolvidos no projeto e descrição sobre o futuro do projeto quanto à probabilidade de atendimento das demais atividades dentro do tempo determinado.

Este escopo sugerido deve refletir toda a demanda explícita e implícita contida na análise de requisitos realizada junto ao cliente. Em conformidade com o que foi especificado em questões de custo e prazo, esta gestão deve ser mantida fiel e atualizada para cada passo dado pelo projeto como forma de apoio à tomada de decisões gerenciais. Ou seja, caso o projeto não venha atender ao planejamento inicial, cabe ao gestor em tempo hábil especificar o motivo dos problemas e apontar soluções que poderão ser aceitas de forma imediata, ou em alguns casos, sugerir a necessidade de nova avaliação de escopo junto ao cliente devido ao alto volume de mudanças do escopo inicial [RB007].

Segundo o processo SPICE, Figura 3, podemos perceber a importância que também deve ser dada ao acompanhamento da qualidade e eficiência da forma de gestão do projeto. Ou seja, o escopo que define as etapas e formatos para avaliações de qualidade no desenvolvimento de software deverão também passar por auditorias para averiguação do real benefício trazido ao projeto por estas avaliações.

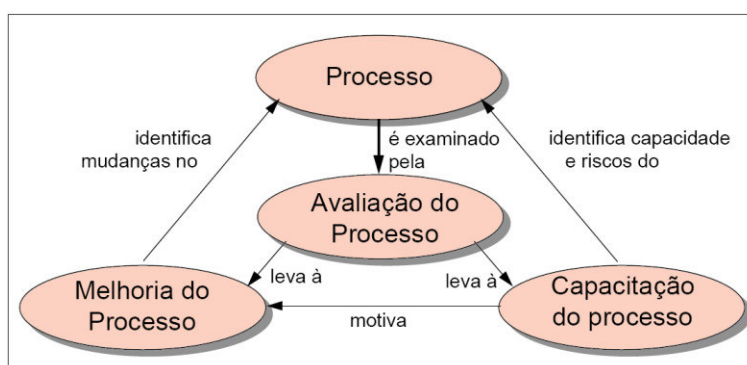


Figura 3 – Avaliação de Processo de Software – SPICE [RB013]

Dentro do contexto de melhoria de processos, a avaliação significa a caracterização das práticas correntes de uma organização, unidade organizacional ou projeto em termos da capacidade dos processos selecionados. A análise dos resultados é feita em relação às necessidades de negócio da organização, identificando os aspectos positivos e negativos, e os riscos associados aos processos. Isto leva a determinar se os processos estão atingindo efetivamente seus objetivos e identificar causas da baixa qualidade, alto custo e tempo excessivo, indicando a priorização na melhoria dos processos.

Existe ainda o modelo SCRUM [RB020] [RB021] que é uma metodologia ágil direcionada para planejamento e acompanhamento do projeto, sem entrar nos detalhes de como é feita a engenharia de software. Alguns conceitos nesta metodologia são consenso entre diversos stakeholders do mundo, e que são tratados como valores. Esses valores são: Pessoas e iterações são mais importantes que processos e ferramentas; Software funcionando é mais importante que uma documentação extensa; O relacionamento com o cliente é mais importante que a negociação do contrato;

Responder às mudanças é mais importante que seguir o planejamento. Porém, vale ressaltar que o termo “é mais importante” não inibe a importância da segunda característica, somente ressalta-se como valor mais objetivado.

Essas práticas ágeis como builds rápidos, refactorings, desenvolvimento iterativo, feedback constante dos stakeholders, interação contínua, gerenciamento de projeto baseado em objetivos, reuniões diárias, etc, já são adotadas para projetos complexos e críticos, onde o retorno das avaliações gerenciais são fatores decisivos para mudanças de escopo.

5. Conclusão

A demanda por qualidade é fator motivador na comunidade de desenvolvimento de software, gerando aceitação para modelos de melhoria e qualidade de software. O que se percebe é que a qualidade esperada pelo ponto de vista do cliente está no produto final, indicando que este produto atende de forma aceitável a suas expectativas. Porém, a qualidade tem sido tratada de forma mais ampla, onde esta se origina no desenvolvimento de software passando por processos consolidados e que fazem um controle rigoroso do atendimento aos requisitos implícitos e explícitos do produto.

O conceito de qualidade não é mensurável ou padrão, ele varia de acordo com o segmento, região, usuários, dentre outros muitos fatores, levando então a se perceber que a qualidade deve ser estudada caso a caso de acordo com o que está sendo estabelecido no levantamento de requisitos. O que é qualidade em um caso específico pode não ser qualidade para o mesmo caso se for tratado por pontos de vistas distintos.

Cabe a empresa ou gerente do projeto definir em cada situação o nível de qualidade aceitável para cada requisito encontrado para o projeto, e para se alcançar esta qualidade esperada deve-se manter um acompanhamento rigoroso periodicamente do atendimento das atividades durante o ciclo de vida do projeto. Reconhecer que o desenvolvimento de software requer práticas especiais de gerenciamento de projeto, é um importante passo para se chegar à qualidade esperada. Este processo é definitivamente inconstante o que descaracteriza todas as regras que são aplicadas em um projeto de Engenharia Civil por exemplo. “Desenvolvimento de Software = Gerenciamento de Incerteza”.

A qualidade se reflete muito na satisfação do cliente, mas está sendo vista também por outros fatores como facilidade de intercomunicação entre outros software e possibilidade de inovações sem causar danos ou atrasos a demais áreas no desenvolvimento de Software.

6. Referências

- [RB001] Guide to the Software Engineering Body of Knowledge, SWEBOK 2004, Capítulo 11. IEEE Computer Society.
- [RB002] Pádua, Wilson de .Engenharia de Software - Fundamentos, Métodos e Padrões. Segunda Edição, Capítulo 11. LTC – 2003.
- [RB003] Qualidade de Software: Teoria e Prática, Orgs. Rocha, Maldonado, Weber, Prentice-Hall, São Paulo, 2001. Capítulo 17.

- [RB004] Punit Ahluwalia, Member, IEEE, and Upkar Varshney, Member, IEEE. A Framework for Transaction-Level Quality of Service E_Commerce. IEEE Transactions on Mobile Computing, vol 6, no 7, July 2007.
- [RB004] Antonio Ruiz, Rafael Corchuelo, Amador Durán. Automated Support for Quality Requirements in Web-Service-Based Systems. Universidad de Sevilla, Spain.
- [RB005] Hoffmann, M., Kühn, N., Weber, M., Bitter, M. (2004) "Requirements for Requirements Management Tools", 12o. IEEE International Requirements Engineering Conference.
- [RB005] Jacobson, I., Booch, G., Rumbaugh, J. (1999) "The Unified Software Development Process", Addison Wesley, 1a edição.
- [RB006] ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. Gestão da qualidade e garantia da qualidade: terminologia - NBR ISO 8402. Rio de Janeiro: [s.n.], 1993, 14p.
- [RB007] Tsukumo, Alfredo N., Rêgo, Claudete M., Salviano, Clenio F.. Qualidade de Software: Visões de Produto e Processo de Software. ATAQS CTI. Campinas SP.
- [RB008] Software Engineering Institute – CMMI web page, acessado em agosto/07, <http://www.sei.cmu.edu/cmmi/>.
- [RB009] IEEE Std 730-1998, IEEE Standard for Software Quality Assurance Plans.
- [RB010] Kenett, Ron; Baker, Emanuel (1999). Software Process Quality: Management and Control. CRC Press, p. 130 ff. ISBN 9780824717339 .
- [RB011] CMMI Product Team (2006) "CMMI for Development, Version 1.2", Software Engineering Institute of Carnegie Mellon University.
- [RB012] ISO/IEC 12207-1, Software life-cycle process; mês/1994 (DIS).
- [RB013] SPICE - Project Overview - acessado em novembro/07, http://www.isospice.typepad.com/isospice_pt_is15504.
- [RB014] Rout, P. T. - "SPICE: A Framework for Software Process Assessment" - Software Process - Improvement and Practice, Pilot Issue, pp 57-66, 1995.
- [RB015] Software Engineering Intitute, Carnegie Mellon - "How does CMMI relate to ISO 9000/9001?" - acessado em novembro/07, <http://www.sei.cmu.edu/cmmi/presentations/sep03.presentations/cmmi-iso.pdf>.
- [RB016] ISO 9000 - Normas de Gestão da Qualidade e Garantia da Qualidade - Diretrizes para Seleção e Uso.
- [RB017] Kano, N., Seraku, N., Takahashi, F., Tsuji, S. - Attractive Quality and Must-Be Quality (Jan. 1984) - in TQM - Ten Elements for Implementation.
- [RB018] Thayer e Dorfman (1993). Software Requirements Engineering. R. H. Thayer, M. Dorfman. IEEE Computer Society Press. 1993.
- [RB019] Soares (2005). Introdução, Identificação e Análise em Engenharia de Requisitos. Antônio Lucas Soares. 2005.
- [RB020] SCHWABER, KEN, 2004. Agile Project Management with SCRUM.
- [RB021] Project Management Institute, 2004. Guia PMBOK.