Melhoria de Processos Individuais e de Equipe

Mateus Y. Utiyama[1], **Rafael F. Goncalves** [1], **Vitor Alberto**[1]

[1]Centro de Ciências Tecnológicas - Universidade do Norte do Paraná (UENP)
Bandeirantes, PR - Brazil

yudiutiyama@gmail.com, rafael@itapecerica.com.br, souzavitor96@gmail.com

Abstract. *This meta-paper will discuss quality improvement in software engineering, firstly describing what are the diverse processes in software engineering, what process model are available, their positive and negative, then we treat the quality improvement, with it's characteristics, the current approaches and the paper's conclusion.*

Resumo. *Este meta-artigo irá tratar da melhoria da qualidade de processos de desenvolvimento de sotfware, primeiramente explicando sobre o que são os processos no desenvolvimento de sotfware, quais modelos de processos existem, e suas vantagens, depois é apresentado o tópico de melhoria de processos, suas características, as abordagens existentes e alguns dos modelos existentes para realizar uma melhoria na qualidade. E por fim é apresentado uma conclusão sobre tudo que foi apresentado.*

1. Introdução

Em um ambiente altamente competitivo, é essencial estar sempre pronto e totalmente funcional para acompanhar as mudanças de mercado e inovações, e para satisfazer as necessidades dos clientes [Büyüközkan and Feyzioğlu 2005].

No mundo atual existe uma demanda não satisfeita por *software* de qualidade. Organizações estão sofrendo fortes pressões para desenvolver sistemas de informações em pouco tempo. Os sistemas precisam ser escalonáveis e integrados com outros sistemas já existentes ou que estão sendo desenvolvidos [Crespo et al. 2004].

De acordo com [Jiang et al. 2004] 15% de todos os projetos de desenvolvimento de *software* nunca entregam um produto final, 80% estão extremamente atrasados e acima do valor.

Para o desenvolvimento de *software* com qualidade, dentro de prazos e custos controlados e compatíveis com o mercado, é fundamental a melhoria dos processos de engenharia de *software*. Para tanto abordagens e experiências para a melhoria de processo de *software* baseadas em modelos têm sido utilizadas com sucesso pelas organizações de *software*. Os mais utilizados SW-CMM, ISO/IEC 12207, ISO/IEC 15504 e CMMI. Esses modelos identificam processos fundamentais e identificam testes de *software*. Teste é fundamental para a avaliação do *software* desenvolvido. Entretanto testar *software* exige conhecimentos, habilidades e infra-estrutura específicos. [Crespo et al. 2004].

2. Processo

Tipicamente, um processo de desenvolvimento é subdividido em fases com objetivos distintos, onde determinadas disciplinas (como análise de requisitos, implementação e

teste) podem ser exercitadas com exclusividade ou predominância. Processos usualmente são definidos como conjunto estruturado de atividades para as quais são determinados artefatos de entrada e saída, papéis de responsáveis e participantes, além de recursos necessários. Atividades podem ser detalhadas pela definição de passos de execução, procedimentos e regras. Porém, esta definição ainda é simples, pois no geral processos não são apenas conjunto de atividades, mas de atividades estruturadas. Também, há mais coisas envolvidas em um processo, como pessoas, recursos restrições, padrões a serem seguidos etc [Wazlawick 2013].

Existem muitos processos de software diferentes, porém todos devem incluir quatro atividades fundamentais para a engenharia de *software*:

1. Especificação de *software*. A funcionalidade do *software* e as restrições a seu funcionamento devem ser definidas.
2. Projeto e implementação de *software*. O *software* deve ser produzido buscando atender às especificações.
3. Validação de *software*. O *software* deve ser validado para garantir que atenda às demandas do cliente.
4. Evolução de *software*. O *software* deve evoluir para atender às necessidades de mudanças dos clientes [Sommerville 2011].

Processo é algo que ocorre em um tempo determinado. Consiste na execução concreta de um conjunto de atividades com o intuito de criar um produto específico[Wazlawick 2013].

Outra expressão relacionada com processo é o modelo de processo, que é um conjunto de regras mais abstratas que especificam a forma geral de processos. Quando uma empresa decide adotar um processo, deve também buscar um modelo e adaptar a filosofia e as práticas recomendadas para criar seu próprio processo. Então, todos os processos da empresa deverão seguir o processo definido [Wazlawick 2013].

Alguns modelos de processos são:

1. Modelo em cascata. Ele considera as atividades fundamentais do processo de especificação, desenvolvimento, validação e evolução, e representa cada uma delas como fases distintas, como: especificação de requisitos, projeto de software, implementação, teste e assim por diante.
2. Desenvolvimento incremental. Esta abordagem intercala as atividades de especificação, desenvolvimento e validação. O sistema é desenvolvido como uma série de versões (incrementos), de maneira que cada versão é um incremento de uma funcionalidade à anterior.
3. Engenharia de *software* orientada a reúso. Essa abordagem tem como base a existência de um número significativo de componentes reusáveis. O processo do desenvolvimento do sistema é concentrado na integração desses componentes em um sistema já existente em vez de desenvolver um sistema a partir do zero [Sommerville 2011].

Esses modelos não são mutuamente exclusivos e muitas vezes são usados em conjunto, especialmente para o desenvolvimento de sistema de grande porte. Em um sistema de grande porte, faz sentido utilizar várias das melhores características do modelo em cascata e dos modelos de desenvolvimento incremental. É necessário ter informações so-

bre os requisitos essenciais do sistema para projetar uma arquitetura de *software* que dê suporte a esses requisitos [Sommerville 2011].

Como vantagens da utilização de modelos temos:

- Tempo de treinamento reduzido: processo bem definidos e documentados, é mais fácil encaixar novos indivíduos na equipe do que quando não se dispões deles.
- Produtos podem ser mais uniformizados: a existência do processo não garante a uniformidade na qualidade dos produtos, porém uma equipe com um processo bem definido tende a ser mais previsível do que a mesma equipe sem processo
- Possibilidade de capitalizar experiências: um processo bem definido deve dar a oportunidade de desenvolvedores modificarem a maneira de ser feito o processo [Wazlawick 2013].

3. Melhoria de processos

Atualmente, existe uma procura constante da indústria por softwares mais baratos e melhores, os quais precisam ser entregues em prazos de entrega (*deadlines*) cada vez mais rigorosos. Então, muitas empresas de software focaram em melhoria de processos de software como uma forma de melhorar a qualidade de seu software, reduzindo custos ou acelerando seus processos de desenvolvimento. A melhoria de processos implica a compreensão dos processos existentes e sua mudança para aumentar a qualidade de produtos e/ou reduzir custos e o tempo de desenvolvimento [Sommerville 2011].

Duas abordagens bastante diferentes são utilizadas para a melhoria e a mudança de processos:

1. A abordagem de maturidade de processo, a qual se centra em melhorar o gerenciamento de processos e projetos e em introduzir boas práticas de engenharia de software em uma organização. O nível de maturidade de processos reflete o grau em que as boas práticas técnicas e de gerenciamento foram adotadas nos processos de desenvolvimento de software da organização. Os principais objetivos desta abordagem são produtos com uma melhor qualidade e previsibilidade de processo.
2. A abordagem ágil, que se centra em um desenvolvimento iterativo e na redução de despesas gerais no processo de software. As principais características dos métodos ágeis são a entrega rápida de funcionalidade e a capacidade de resposta às mudanças de requisitos de cliente [Sommerville 2011].

3.1. O processo de melhoria de processos

Os processos de software podem ser observados em todas as organizações, desde empresas com uma pessoa até as grandes multinacionais. Esses processos são de tipos diferentes, dependendo do grau de formalidade do processo, dos tipos de produtos desenvolvidos, do tamanho da organização e assim por diante. Não existe um processo de software 'ideal' ou 'padrão' que seja aplicável para a todas as organizações. Cada empresa desenvolve seu próprio processo dependendo de seu tamanho, de seu conhecimento e das habilidades de seu pessoal, do tipo de software que esteja sendo desenvolvido, do cliente e dos requisitos do mercado, bem como da cultura da empresa [Sommerville 2011].

Portanto, a melhoria dos processos não significa apenas a adoção de métodos particulares ou ferramentas ou o uso de um processo genérico publicado. Embora

organizações que desenvolvem o mesmo tipo de *software*, claramente, tenham muito em comum, sempre existem fatores organizacionais locais, procedimentos e normas que influenciam o processo. Raramente, você terá sucesso na introdução de melhoria de processos, caso tente alterar o processo para um processo usado em outros lugares. Sempre deve considerar o ambiente e a cultura local e como tais fatores são alterados por propostas de mudanças de processos. Também deve ser considerado quais aspectos você deseja melhorar. [Sommerville 2011].

O processo de melhoria de processos é um processo cíclico (medir, mudar, analisar) e envolve três subprocessos:

1. Medição de processo. Os atributos do projeto atual ou dos produtos são medidos. O objetivo é melhorar as medidas de acordo com os objetivos da organização envolvida na melhoria de processos. Isso forma uma base que o ajuda a decidir se as melhorias no processo foram eficazes.
2. Análise de processo. O processo atual é avaliado e os gargalos e pontos fracos são identificados. Os modelos de processo que descrevem o processo podem ser desenvolvidos durante essa fase. A análise pode centrar-se levando em consideração as características de um processo, como rapidez e robustez.
3. Mudanças de processo. As mudanças de processos são propostas para resolver alguns dos pontos fracos identificados no processo. Elas são introduzidas e o ciclo recomeça para a coleta de dados sobre a eficácia das mudanças [Sommerville 2011].

A melhoria de processos é uma atividade de longo prazo, pois cada uma das fases do processo de melhoria pode durar vários meses. Também é uma atividade contínua, pois quaisquer novos processos introduzidos alterarão o ambiente de negócios e os novos processos terão de evoluir para levar em conta tais mudanças [Sommerville 2011].

4. Métricas de Qualidade de Software

Como feito por [Humphrey 1999] quando analisando os resultados do PSP (Personal Software Process) é comum o uso de métricas de software para a analise com fins de comparação e validação entre soluções de desenvolvimento de software por exporem detalhes específicos das mesmas, essas métricas a pesar de suas limitações são muitas vezes capazes de julgar certos critérios como citado por [Schneidewind 1992]:

1. Associação
2. Consistência
3. Poder de Discriminação
4. Rastreamento
5. Previsibilidade
6. Repetibilidade

Esses critérios por fim nos ajudam a julgar fatores da solução de software como:

- Performance
- Reusabilidade
- Facilidade de Manutenção
- Eficiência

E isso é utilizado como guia ao quantificar as qualidades de cada estagio de um processo.

5. *Framework* CMMI de melhorias de processos

O Instituto de Engenharia de Software (SEI, do inglês *Software Engineering Institute*) dos Estados Unidos foi criado visando a melhoria das capacidades da indústria norte-americana de software. O SEI iniciou um estudo sobre como avaliar as capacidades dos prestadores de serviços de software, em meados de 1980. Como resultado foi o Modelo Maturidade e Capacidade (CMM, do inglês *Capability Maturity Model*). Ele tem sido tremendamente influente em convencer a comunidade de engenharia de software a levar a sério a melhoria de processos. O modelo foi seguido por vários outros modelos de maturidade e capacidade, incluindo o Modelo de Maturidade e Capacidade de pessoas (CMM-P, do inglês *People Capability Maturity Model*) e o Modelo de Capacidade de Engenharia de Sistemas [Sommerville 2011].

Em uma tentativa de integrar a multiplicidade de modelos de capacidade com base na noção de maturidade de processos, o SEI embarcou em um novo programa para desenvolver um modelo de capacidade integrado (CMMI). O *framework* do CMMI substitui os CMMs para Software e Engenharia de Sistemas e integra outros modelos de maturidade e de capacidade. Ele possui duas instâncias, por estágio e contínua, e aborda alguns dos pontos fracos. O modelo CMMI tem como objetivo ser um *framework* de melhoria de processos com ampla aplicabilidade em uma gama de empresas. Sua versão por estágios é compatível com o CMM para Software e permite que o desenvolvimento de sistemas e processos de gerenciamento de uma organização seja avaliado e que a ele seja atribuído um nível de maturidade classificado como de 1 a 5. Na versão contínua é possível uma classificação de granularidade mais baixa de maturidade de processo [Sommerville 2011].

O modelo CMMI é muito complexo. E possuí como principais componentes:

1. Conjunto de áreas de processo relacionadas às atividades de processos de software.
2. Um número de metas, que são descrições abstratas de um estado desejável a ser atingido por uma organização.
3. Um conjunto de boas práticas, que são as descrições das formas de como alcançar uma meta [Sommerville 2011].

Uma avaliação de CMMI envolve examinar os processos em uma organização e classificar esses processos ou áreas de processo em uma escala de seis pontos relacionada ao nível de maturidade em cada área de processo. Quanto mais maduro um processo, melhor. A escala atribui níveis de maturidade para uma área de processo, da seguinte forma:

1. Incompleto/Inicial. Pelo menos uma das metas especificas associadas com a área de processo não está satisfeita.
2. Executado. Metas associadas com a área de processo são satisfeitas e, para todos os processos
3. Gerenciado. Metas associadas com a área de processo são cumpridas e as políticas organizacionais definem quando cada processo deve ser usado.
4. Definido. Concentra-se na padronização e implantação de processos organizacionais.
5. Gerenciado quantitativamente. Existe uma responsabilidade organizacional de usar métodos estatísticos e outros métodos quantitativos para controlar os subprocessos.

6. Em otimização. A organização deve usar medições de processo e produto para dirigir a melhoria de processos [Sommerville 2011].

6. MPS.BR

MR-MPS (Modelo de Referência para Melhoria do Processo de Software), ou MPS.BR é um modelo de avaliação de empresas produtoras de software brasileiro criado através de uma parceria entre a SOFTEX (Associação para Promoção da Excelência do Software Brasileiro), o governo federal e academia (pesquisadores em geral). O modelo brasileiro é independente, mas compatível com as Normas ISO 12207 e 15504 (SPICE), bem como com o CMMI [Sommerville 2011].

A principal justificativa para a criação deste modelo foram os altos custos dos processos de avaliação ou certificação internacionais, que se tornam proibitivos para pequenas e médias empresas. Assim, o MPS.BR tem um custo significativamente mais baixo, por ter consultores e avaliadores residentes no Brasil e também pelo fato de que apresenta 7 níveis de maturidade em vez de apenas 5, como no CMMI. Com isso a escala de progressão na melhoria de processos têm degraus mais suaves, especialmente nos níveis baixos, ou seja, é possível subir um nível com menos esforço do que seria necessário para subir um nível no CMMI [Sommerville 2011].

Os níveis de maturidade do MPS.BR são os seguintes:

- A - Em otimização.
- B - Gerenciado quantitativamente.
- C - Definido.
- D - Largamente definido.
- E - Parcialmente definido.
- F - Gerenciado.
- G - Parcialmente gerenciado [Sommerville 2011].

7. ISO/IEC 15504 - SPICE

A Norma ISO/IEC 15504, também conhecida como SPICE (*Software Process Improvement and Capability dEtermination*, foi criada para complementar a já existente ISO/IEC 12207 (definição de processos do ciclo de vida de desenvolvimento de software) e seu objetivo é orientar a avaliação e a autoavaliação da capacidade da empresa em processos e, com essa avaliação, permitir a melhoria dos processos. A Norma 15504 é estruturada em duas dimensões:

1. Dimensão de processos: quais são os processos avaliados
2. Dimensão de capacidade: qual a capacidade da empresa avaliada em cada um desses processos [Sommerville 2011].

E os níveis de capacidade são:

- 0 - incompleto: existe uma falha geral em se ater aos objetivos de um processo ou à ausência dele.
- 1 - processo realizado: o propósito do processo geralmente é obtido, porém não necessariamente de forma planejada ou rastreável.
- 2 - processo gerenciado: os projetos entregam produtos com uma qualidade aceitável respeitando os prazos e orçamentos definidos.

- 3 - processo estabelecido: a gerência de projetos é realizada de acordo com um projeto estabelecido, onde os bons princípios de engenharia de software são empregados. Esse nível necessita de um gerenciamento planejado e da utilização de um processo padrão.
- 4 - processo previsível: os projetos são realizados de forma consistente dentro de limites de controle. Existem medidas de performance e a performance é gerenciada de forma objetiva e a qualidade do trabalho é quantitativamente conhecida.
- 5 - processo otimizado: a realização do processo é otimizada buscando satisfazer às necessidades correntes e futuras do negócio, e os processos satisfazem repetidamente essas necessidades. São estabelecidos metas quantitativas de eficiência e efetividade para processos [Sommerville 2011].

E existem 4 níveis de avaliação dos níveis de capacidade: 'N' (não obtido), 'P' (parcialmente obtido), 'L' (amplamente obtido) e 'F' (atingido totalmente) [Sommerville 2011].

7.1. Processo de Avaliação

De acordo com [Sommerville 2011], para a realização da avaliação, a própria Norma 15504 apresenta um guia, que inclui um modelo e um processo de avaliação, bem como as ferramentas para proceder com a avaliação. A avaliação pode ser feita, basicamente, buscando dois objetivos: determinar capacidade e a melhoria de processo. E o processo de avaliação inclui as seguintes atividades:

1. Iniciar a avaliação por parte do interessado.
2. Selecionar o avaliador e sua equipe.
3. Planejar a avaliação.
4. Reunião de pré-avaliação.
5. Coletar dados.
6. Validar dados.
7. Atribuir nível de capacidade aos processos.
8. Relatar os resultados da avaliação.

Apenas a consulta à norma, não é o suficiente para que ocorra o processo de avaliação, é necessário a contratação de pessoas com treinamento específico em avaliação SPICE, que irá coletar dados de diversas maneiras, incluindo entrevistas, análise de dados estatísticos e registros de qualidade [Sommerville 2011].

8. Modelo SQO-OSS

O modelo SQO-OSS (*Software quality observatory for open source software*) foi criado com o foco especifico no desenvolvimento de código aberto (*open-source*), os modelos tradicionais tem uma metodologia de avaliação geralmente baseada na analise hierárquica dos modelos de qualidade, o que não se adapta bem em grandes projetos de código aberto, pois são naturalmente segmentados, diminuindo a eficacia da analise feita com esses métodos tradicionais. [Samoladas et al. 2008] concluiu em seu trabalho que o modelo foi bem aceito por permitir que o avaliador altere os pesos usados na avaliação, com fim de alterar os critérios usados. O modelo SQO-OSS continua usando uma abordagem hierárquica porem adequada para a modularização dos projetos de código aberto, ele se diferencia de outros modelos de qualidade para código aberto em 6 pontos distintos:

1. O modelo foi construído com foco na automação, enquanto outros modelos necessitam que o usuário faca a coleção de métricas de forma manual.
2. O modelo trata a monitoração do sistema de forma continua, garantindo o melhor resultado com dados recentes.
3. O modelo não avalia funcionalidades, pois essas requerem uma entrada do avaliador, deixando a julgamento mais subjetivo.
4. O modelo foca no código fonte, aplicando métricas e peso para avaliação do projeto como um todo.
5. O modelo leva em consideração fatores da comunidade código aberto, que podem ser analisados também de forma automática
6. O modelo permite que seus parâmetros de avaliação sejam alterados pelo avaliador, deixando-o determinar seus critérios

9. PSP e TSP

PSP (*Personal Software Process*) e TSP (*Team Software Process*) são abordagens com o difícil intuito de melhoria de processos invidiais e em equipes. [Humphrey 1999] define a problematização como uma questão de convicção, pois os engenheiros de software tem por definição praticas diferentes na escrita de novos programas, algo causado pelas nossas atuais metodologias de aprendizagem, que alguns engenheiros podem melhorar suas técnicas mas que muitos raramente vão selecionar quais métodos e praticas usar de acordo com a melhor escolha objetiva, pois preferem utilizar alguma abordagem que já tenha experiencia previa.

9.1. *Personal Software Process*

O método de [Humphrey 1999] para verificar a eficacia e eficiência do processo foi segundo ele bem agressiva, pois seria o método mais decisivo de mudar a mentalidade dos desenvolvedores. Eles foram removidos dos seus projetos atuais e foram dados exercícios de programação com um novo método de trabalho, suas performances seriam analisadas durante todo o desenvolvimento das tarefas.

9.1.1. Resultados

Os resultados mostraram que a melhoria constante (analisado com métricas de software como KLOC, *defects per 1000 lines of code*) na qualidade das soluções, e que essa progressão foi intensificada com o nível de experiencia dos profissionais. Grandes empresas como Baan, Boeing, Motorola e Teradyne estão utilizando a abordagem PSP para melhor qualidade de desenvolvimento.

9.2. Team Software Process

Segundo [Humphrey 1999] essa abordagem estende e refina os métodos CMM e PSP, aplicando-as a grupos de engenheiros encarregados de implementação ou manutenção de software, o método foca em construir um ambiente de trabalho que estimule uma alta produtividade de trabalho, com seus 5 objetivos:

1. Construção de equipes independentes, que consigam determinar e seguir seus objetivos e processos.

2. Instruir gerentes para que possam motivar as equipes.
3. Acelerar a melhoria de processos, fazendo com que o Nível 5 do CMM seja o esperado.
4. Providenciar instruções de melhorias para empresas de alta maturidade
5. Facilitar o ensino dessas habilidades em universidades

O beneficio principal do TSP é instruir o como produzir produtos de melhor qualidade para os mesmos custos e prazos planejados.

10. Fatores Humanos

Apesar da existência de boas referências para melhoria de processos e equipes bem intencionadas, a literatura reporta que cerca de 70% das iniciativas falham ou sequer se iniciam. Ocorre que, por mais detalhados que sejam os modelos de melhoria de processos, tanto o trabalho de melhoria quanto o trabalho com os processos melhorados acabam sendo feitos por pessoas. Portanto, é necessário considerar os fatores humanos envolvidos com as atividades de mudança dentro da empresa. O processo de mudança é complexo e demanda grande esforço para que se obtenha sucesso. O processo de adoção de mudanças é dividido em 8 estágios, organizados em 3 fases:

1. Preparação: nessa fase ocorrem os estágios de contato e conhecimento. Se após o contato a educação necessária para chegar ao conhecimento não for estabelecida, a iniciativa de mudança poderá falhar por falta de conhecimento.
2. Aceitação: nessa fase ocorrem os estágios de compreensão e percepção positiva. Se após o conhecimento não houver compreensão, a iniciativa poderá falhar devido à confusão. Se durante a compreensão houver percepção negativa sobre a possibilidade de mudança, esta também poderá não ocorrer. Porém, mesmo se houver compreensão e percepção positiva, a iniciativa ainda poderá não ser implementada em função de decisão baseada em custo/benefício ou outros fatores.
3. Compromisso: nessa fase ocorrem os estágios de instalação, adoção, institucionalização e internalização. Mesmo se após a instalação e o uso inicial, os novos processos ainda poderão ser abandonados antes de serem efetivamente adotados pela organização. E, mesmo após a adoção final e seu uso extensivo dos novos processos, ainda poderão ser abandonados se não forem efetivamente institucionalizados. A internalização é o estágio final no qual o que foi institucionalizado passa a fazer parte da cultura organizacional e nem é mais percebido como algo à parte [Sommerville 2011].

11. Prevenção de Erros

No desenvolvimento de software a prevenção de erros é algo normalmente negligenciado por engenheiros, pois não traz um retorno imediato, porem sua utilidade é demonstrada ao projeto tomar qualquer tempo de vida, pois segundo [Kumaresh and Baskaran 2010] quanto mais tempo um erro existe dentro de um sistema mais caro ele se torna. DP (*Defect Prevention*) é um método eficaz e barato que reduz custos, melhorando tempos de entrega e recursos, seu único objetivo é a detecção de erros, achando qualquer imperfeição de software, para que essa não volte a ser um problema, o que afetaria diretamente a produtividade. DP possui 5 estágios no seu fluxo de trabalho:

1. Identificação do defeito: nesse estagio os defeitos são identificados por atividades planejadas para esse fim, como testes de código.

2. Classificação do defeito: é feita uma classificação com o defeito do estagio anterior com o ODC (*Orthogonal Defect Classification*), onde o defeito é agrupado em tipos de defeitos ao invés de ser considerado individualmente. A metodologia ODC utiliza atributos fixos para essa categorização, aumentando a velocidade desse estagio e habilitando uma visão mais ampla da causa dos defeitos.
3. Analise do defeito: a informação do defeito nesse estagio é utilizada para melhora continua dos processo, utilizando RCA (*Root Cause Analysis*) que tem como objetivo identificar a causa principal daquele defeito em especifico afim de eliminarla.
4. Prevenção do defeito: esse estagio tem a fincão de prevenir que as causas de defeitos (detectadas nos últimos estágios) sejam resolvidas para que não cause novos defeitos futuros.
5. Melhoria do processo: as alterações preventivas são documentadas em manuais de qualidade, a serem utilizadas em futuros processos, melhorando a qualidade dos mesmos evitando que tenham os defeitos já detectados previamente.

11.1. Root Cause Analysis

RCA pode se referir a identificação da causa principal de defeitos específicos ou ate de processos, e tem a intenção de eliminar esses problemas de forma definitiva, tem dois princípios:

- Reduzir os defeitos para uma melhora de qualidade de software: informação da analise deve ser útil para prevenção de defeitos e a detecção mais rápida possível de problemas recorrentes.
- Uso de experiencia loca e de terceiros: acreditasse que as melhores soluções seriam achadas ao discutir esses problemas com quem entende do assunto.

Seu foco é determinar a origem do problema, de uma forma quantitativa, quanto mais exemplos anualizados mais preciso sera a identificação da causa, é comum a utilização da técnica que consiste do uso de diagramas causa e efeito, onde são relacionados graficamente múltiplos fatores que lidam a certas situações.

12. Conclusão

Podemos concluir que a qualidade de processos é uma parte fundamental do desenvolvimento de software, que necessita de uma grande atenção pois o processo de melhoria de qualidade é um processo demorado e que se mal feito pode gerar grandes despesas, e infelizmente vem sendo negligenciada por equipes e gerentes, porém com a grande competitividade do mercado, estão sendo obrigados a darem a devida atenção ao tema para se manterem relevantes no mercado, contudo existem diversas soluções por meio de metodologias capazes de remediar a situação, melhorando a qualidade dos processos, economizando assim recursos como tempo e dinheiro.

References

Büyüközkan, G. and Feyzioğlu, O. (2005). Group decision making to better respond customer needs in software development. *Computers & Industrial Engineering*, 48(2):427–441.

Crespo, A. N., Silva, O. J., Borges, C. A., Salviano, C. F., Argollo, M., and Jino, M. (2004). Uma metodologia para teste de software no contexto da melhoria de processo. *Simpósio Brasileiro de Qualidade de Software*, pages 271–285.

Humphrey, W. S. (1999). Pathways to process maturity: The personal software process and team software process. *SEI Interactive*, 2(2):1–17.

Jiang, J. J., Klein, G., Hwang, H.-G., Huang, J., and Hung, S.-Y. (2004). An exploration of the relationship between software development process maturity and project performance. *Information & Management*, 41(3):279–288.

Kumaresh, S. and Baskaran, R. (2010). Defect analysis and prevention for software process quality improvement. *International Journal of Computer Applications*, 8(7):42–47.

Samoladas, I., Gousios, G., Spinellis, D., and Stamelos, I. (2008). The sqo-oss quality model: measurement based open source software evaluation. In *IFIP International Conference on Open Source Systems*, pages 237–248. Springer.

Schneidewind, N. F. (1992). Methodology for validating software metrics. *IEEE Transactions on software engineering*, 18(5):410–422.

Sommerville, I. (2011). *Engenharia de Software, 9 edição*, volume 9.

Wazlawick, R. (2013). *Engenharia de software: conceitos e práticas*, volume 1. Elsevier Brasil.