Visão geral da qualidade de software?

Danilo de Sousa Abreu
Universidade Nove de Julho
São Paulo, Brasil.
email: danilo.danilosousa@gmail.com

***Resumo*— Este trabalho apresenta e discute pontos relevantes sobre reúso de software baseado em desenvolvimento orientado a objetos em Java. Diversas características de reuso estão contidas dentro da abordagem de orientação a objetos, no entanto, o objetivo e reuni-la e apresenta-las.**

Palavras-chave—Reúso; Orientação objetos; Java; Herança e Polimorfismo.

Abstract— In this paper it's shown and discussed some relevant points about software development based on Object-Oriented. The approach of Object-Oriented contain many characteristics of reuse, therefore, the goal is put it all together and show.

Keywords—Reuse; Object-Oriented; Java; Inheritance and Polymorphism.

1. INTRODUÇÃO

Dentro de um cenário de grande competitividade, e cada vez mais acirrada, as organizações buscam diferenciais para os seus produtos ou serviços. Cada vez mais o conjunto de relação custo benefício, versus curto prazo e produtos ou serviços com qualidade são almejados pelos consumidores. Para as organizações que pretendem manterem-se vivas dentro deste cenário, devem estar cada vez mais ligadas as novas necessidades buscadas por esse mercado exigente e extremamente fértil atualmente [2], [3] e [5].

No mercado de desenvolvimento de soluções de software obter diferencial através de custo ou prazo é extremamente difícil. Levando em consideração que a mão de obra desse setor que possui uma qualificação bem nivelada, onde profissionais têm quase um mesmo nível de aptidão técnica resultando em um custo de mão de obra e prazo bem equilibrado entre as empresas, a qualidade desponta como um grande diferencial. Obter um processo de qualidade que promove uma cultura organizacional de garantia e controle da qualidade dos seus produtos torna-se essencial. As empresas que alcançam um nível de maturidade de processo de qualidade, certificado através de instituições independentes, reconhecidas pelo mercado, possui um grande diferencial. Empresas com esse tipo de certificado podem usufruir de um processo seguro, eficaz e controlado, capaz de disseminar uma cultura organizacional para gestão racional do desenvolvimento de seus produtos de software, além explorarem como apelo comercial [2] e [5].

A qualidade de software é um processo sistemático que focaliza todas as etapas e artefatos produzidos com o objetivo de garantir conformidade de processos e produtos prevenindo e eliminando defeitos (BARTIÉ, 2002). De acordo com BARTIÉ é impossível obter um software de qualidade com processos de desenvolvimento frágeis e deficientes. Vemos então que qualidade de software está intrinsecamente ligada a qualidade dos processos de produção deste produto. Podemos estabelecer então duas dimensões fundamentais da qualidade do software: qualidade de processo e qualidade do produto. Diante de um desafio de garantir a qualidade de um software, de fato busca-se uma cultura de inibição de falhas e erros, estruturando processos que possuam mecanismos para mitigar brechas de possíveis fontes de defeitos. Esses processos estruturados buscam avaliar a qualidade de todas as saídas geradas durante o ciclo de desenvolvimento através de procedimentos e métricas. Dessa forma todo o

processo de desenvolvimento deste produto estará coberto e garantido. Portanto, para se alcançar a qualidade de software é preciso que a garantia da qualidade de software seja parte integrante do ciclo de desenvolvimento e esteja presente em todas as fases.

A Garantia da qualidade de Software, *Software Quality Assurance* - SQA, deve ser um processo gerido de forma independente do processo de desenvolvimento do produto e se possível deve estabelecer uma relação de "um-para-um" entre as fases de desenvolvimento e as atividades a serem desempenhas por uma equipe de qualidade (BARTIÉ, 2002). Para PRESSMAN (PRESSMAN, 1995), a garantia da qualidade de software é uma "atividade de guarda-chuva" que é aplicada ao longo de todo o processo de engenharia de software. Isso nada mais é que, para se garantir a qualidade é esperado que haja para cada etapa de produção, um processo de avaliação da aderência e conformidade da saída produzida e dos processos, obtendo assim ao final de cada iteração um resultado com mais qualidade em sua totalidade, ou seja, diversos pedaços com qualidade remontam um todo com mais qualidade.

Partimos então deste ponto para tentar descrever o que é Garantia de Qualidade de Software. Dentre os mais notáveis trabalhos sobre engenharia de software (PRESSMAN, 1995), temos sobre garantia da qualidade de software uma abordagem que abrange:

1: métodos e ferramentas de análise, projeto, codificação e teste;

2: revisões técnicas formais que são aplicadas durante cada fase de engenharia de software;

3: uma estratégia de teste de múltiplas fases;

4: controle da documentação de software e das mudanças feitas nela;

5: um procedimento para garantir a adequação aos padrões de desenvolvimento de software (quando aplicáveis);

6: mecanismo de medição e divulgação.

Podemos observar que Pressman prevê um conjunto robusto de processos para que o produto de software esteja em conformidade com os padrões de qualidade. Assim como um produto de manufatura, um produto de software com qualidade pode ser entendido como algo que possui uma adesão estrita e consistente com os padrões mensuráveis e verificáveis para alcançar a uniformidade da produção que satisfaça os requisitos específicos do cliente ou do usuário.

Invariavelmente a garantia da qualidade de software está atrelada a verificação e validação através de modelos de gerenciamento e de controle, como *Capability Maturity Model Integration* (CMMI), ISO 9000-3, ISO/IEC 15504, ISO/IEC 25051 e MPS.BR (MPS-SW) que foram motivados pelas falhas nos processos de gerência e manutenção durante o desenvolvimento de software (CESAR, 1997) e ainda a antiga ISO/IEC 9126 revisada pela ISO/IEC 25010:2011 para avaliação da qualidade do produto de software através de um conjunto de características (BARBACCI, 1995). Dessa forma definimos que as duas dimensões de qualidade de software: qualidade do processo e qualidade do produto está amplamente cobertas por modelos difundidos no mercado, desenvolvidos e amadurecidos ao longo dos anos através de pesquisa e desenvolvimento prático.

O objetivo deste trabalho é apresentar uma abordagem de garantia da qualidade de software alinhada com os processos contidos no modelo de maturidade do CMMI-DEV e apresentar boas práticas contidas em outros modelos, como PMP, ITIL e Normas ISO também voltadas para garantia da qualidade.

2. QUALIDADE

2.1. CONCEITO

Qualidade é um conceito subjetivo e está relacionado às percepções de cada indivíduo sob os diversos fatores como: cultura, crenças, valores, posição social e entre outros. Em termos de produtos e serviços qualidade se relaciona

diretamente com as exigências, expectativas e necessidades dos clientes, como por exemplo: valor agregado, relação custo-benefício, durabilidade e acessibilidade.

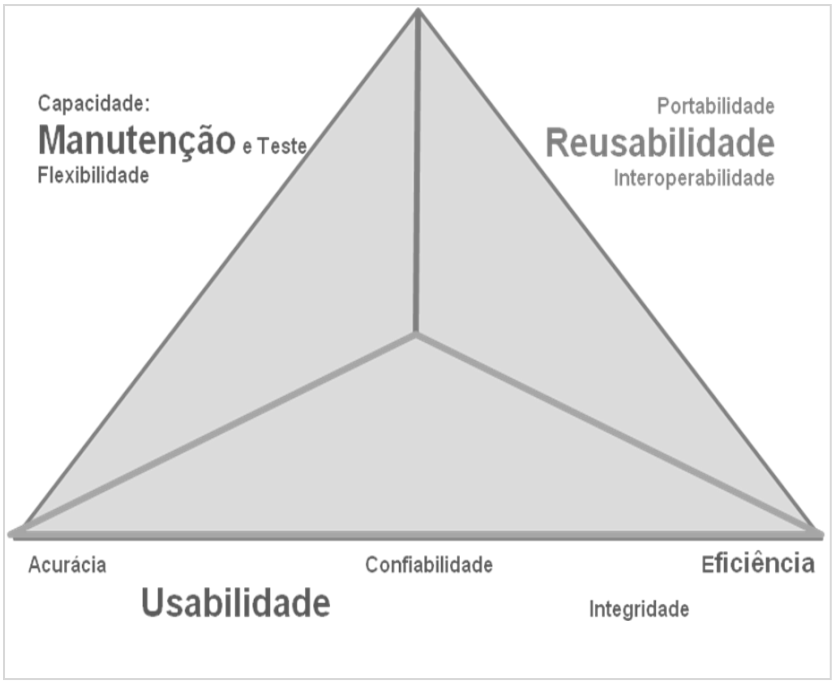


Figura 1. Modelo McCall (1977)

Seguindo nesta direção temos algumas definições importantes já conhecidas e estudadas. "Qualidade é a habilidade de um conjunto de características inerentes a um produto, componente de produto ou processo atenderem aos requisitos dos clientes" (SEI 2006). Já para a norma ISO 8402 (ABNT, 1994), "Qualidade é a totalidade das características de uma entidade, que lhe confere a capacidade de satisfazer às necessidades explícitas e implícitas dos *stakeholders*".

Podemos observar que qualidade está diretamente vinculada com as expectativas e necessidades do cliente. Segundo Philip Crosby (CROSBY, 1979) qualidade é estabelecer conformidade com os requisitos. Para a norma ISO9000 estabelece também que a "qualidade é o grau no qual um conjunto de características inerentes satisfaz a requisitos" (ISO9000: 2005).

Desta forma, assim como para outros setores, qualidade é fator crítico de sucesso para a indústria de software (MR-MPS-SW:2012, 2012), tendo em vista o acelerado avanço tecnológico rápido e inovador e também cada vez mais presente no cotidiano dos indivíduos.

2.2. QUALIDADE DE SOFTWARE

A Qualidade de Software é também algo complexo, no entanto, alguns modelos ou conjunto de atributos podem classificar a qualidade. McCall (MACCALL, 1977) propõe uma categorizarão dos fatores da qualidade de software, com ênfase em três aspectos importantes: Características manutenção, operação, e sua capacidade de adaptabilidade.

Boehm acrescentou algumas características com o modelo de McCall com ênfase na capacidade de manutenção do produto de software (BOTELLA, 2004). Além disso, o modelo inclui considerações envolvidas na avaliação de um software produto no que diz respeito à utilidade do programa. O modelo proposto por Boehm é similar ao modelo de McCall e que representa uma hierárquica estrutura de características, cada um dos quais contribui para a qualidade total. A noção de Boehm inclui as necessidades dos usuários, assim como McCall, no entanto, o modelo de Boehm contém apenas um diagrama, sem qualquer sugestão de como medir as características de qualidade (BEHKAMAL, 2008).

Outro modelo também muito conhecido foi o FURPS (WATSON, 2006) proposto por Robert Grady e HewlettPackard Co., que decompõe características em duas categorias diferentes de requisitos:

Os requisitos funcionais: Definido pela entrada e saída esperada.

Os requisitos não funcionais: Usabilidade, confiabilidade, desempenho e capacidade de suporte.

Uma desvantagem do modelo FURPS é que ele não leva em conta a portabilidade do produto de software (BEHKAMAL, 2008).

Muitos outros modelos surgiram, contudo, havia a necessidade de um modelo padrão, e com isso deu-se início a ISO/IEC JTC1 que buscou desenvolver consensos necessários e incentivar a normalização a nível mundial sobre os modelos de qualidade de software. As primeiras considerações foram feitas em 1978, e dando início em 1985 a ISO/IEC 9126. A ISO/IEC 9126 é parte da família ISO 9000, que é o padrão mais importante para a garantia da qualidade, no entanto, foi revisada em 2011 pela norma ISO/IEC 25010:2011 (ISO/IEC 25010, 2011), também conhecida como **SQuaRE** (*Software product Quality Requirements and Evaluation*).

A norma ISO/IEC 25010, propõe um conjunto de atributos de qualidade, distribuídos em 8 (oito) conjuntos principais, com cada uma deles divididos em sub características descritas pela norma. A nova norma tem oito conjuntos de características de qualidade do produto, em contraste com a ISO 9126 com seis, e 31 sub características.

- **Funcionalidade**: A capacidade de um software prover funcionalidades que satisfaçam o usuário em suas necessidades declaradas e implícitas, dentro de um determinado contexto de uso.

- **Confiabilidade**: O produto se mantém no nível de desempenho nas condições estabelecidas.

- **Operabilidade**: Observa a adequação, fácil utilização, não exposição de erros ao usuário, cuidado com a estética da interface com o usuário e acessibilidade,

- **Segurança**: Preocupação com confidencialidade, integridade, responsabilidade e autenticidade.

- **Compatibilidade**: coexistência e interoperabilidade.

- **Eficiência**: O tempo de execução e os recursos envolvidos são compatíveis com o nível de desempenho do software.

- **Capacidade de manutenção**: A capacidade (ou facilidade) do produto de software ser modificado, incluindo tanto as melhorias ou extensões de funcionalidade quanto às correções de defeitos, falhas ou erros.

- **Portabilidade**: A capacidade de o sistema ser transferido de um ambiente para outro.

Podemos observar que temos a partir da ISO/IEC 25010 um conjunto bem amadurecido e suficientemente capaz que cobrir os diversos modelos já propostos de forma integrada.

Ainda assim podemos dizer que existem duas visões de qualidade de software, uma dos clientes e outra dos que desenvolvem o software, mas ambas concordam que o software não pode ter defeitos. O cliente avalia o software sem conhecer seus aspectos internos, está apenas interessado na facilidade do uso, no desempenho, na confiabilidade dos resultados obtidos e também no preço do software. Os que desenvolvem o software avaliam aspectos internos como taxa de defeitos, confiabilidade, facilidade de manutenção e também aspectos de conformidade em relação aos requisitos dos clientes (ISO, 1991), (CAROSIA, 2003). Para PRESSMAN (PRESSMAN, 1995) essas duas visões podem ser exemplificadas como:

1. Fatores que podem ser medidos diretamente (por exemplo, erros/KLOC/unidade de tempo)

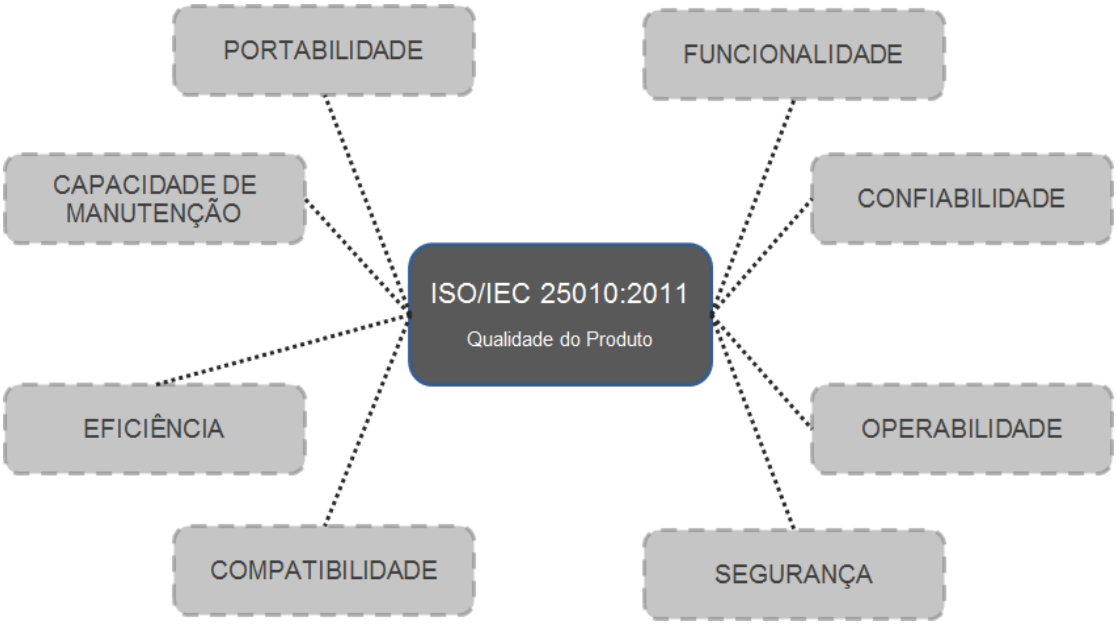
2. Fatores que podem ser medidos apenas indiretamente (por exemplo, usabilidade ou capacidade de manutenção).

2.3. QUALIDADE DO PROCESSO

Antes de discutir sobre qualidade do processo, devemos definir o que é entendido como processo. Processo são um conjunto de passos ordenados, constituídos por atividades, métodos, práticas e transformações, usado para atingir uma meta. Esta meta geralmente está associada a um ou mais resultados concretos finais, que são os produtos da execução do processo (PÁDUA, 2003).

Um processo de software é um conjunto de ferramentas, métodos e práticas usadas para produzir software. O processo de software é representado por um conjunto sequencial de atividades, objetivos, transformações e eventos que integram estratégias para cumprimento da evolução de software (PRESSMAN, 1995). A norma ISO 9000, define processo como um conjunto de atividades inter-relacionadas ou interativas que transformam entradas em saídas.

Como a qualidade do produto pode ser vista como uma consequência da qualidade do processo, os certificados mais valiosos são aqueles que certificam o processo de produção de um produto e não os que certificam simplesmente o produto. Os estudos sobre qualidade, mais recentemente, são voltados para o melhoramento do processo de software, pois ao garantir a qualidade do processo, já se está dando um grande passo para garantir também a qualidade do produto (CAROSIA, 2003).



Um processo de qualidade, de maneira geral, é ter um processo documentado, estabelecidos a partir de modelos amadurecidos, e que buscam a melhoria contínua. Os principais modelos da atualidade são o *Capability Maturity Model Integration* (CMMI) e a norma ISO/IEC 15504-5. No cenário nacional, o Modelo de Referência MPS que faz parte do MPS.BR (Melhoria de Processo do Software Brasileiro).

Figura 2. Figura ISO/IEC 25010 (2011)

TODO: INCLUIR PILARES DA QUALIDADE

3. MODELO DE MATURIDADE CMMI

3.1. CMM (*Capability Maturity Model*)

Antes de descrever sobre o modelo CMMI, é extremamente importante colocar seu contexto histórico e motivação. O CMM, como descrito a seguir, é o precursor do modelo CMMI, motivados pela necessidade de se ter sob o controle processos de software e uma padronização no processo de

desenvolvimento durante um período descrito como a crise de software.

O CMM (PAULK, 1993) foi desenvolvido pelo SEI *(Software Engineering Institute)*, ligado a Universidade Carnegie Mellon e financiado pelo Departamento de Defesa Norte-Americano, com o objetivo de estabelecer um padrão de qualidade para o software desenvolvido para as Forças Armadas. Este modelo foi proposto por Watts S. Humphey, a partir de conceitos de qualidade total estabelecido por Crosby. Segundo Tingey (1997), Crosby mostrou que a implementação de sistemas de qualidade em empresas segue um amadurecimento gradativo em patamares denominados "incerteza, despertar, esclarecimento, sabedoria e certeza"

Em 1987, a partir do trabalho de Humphrey, o SEI lançou uma breve descrição de um ambiente de maturidade de processo de software e um questionário de maturidade para avaliar o estado corrente das práticas de software e identificar as áreas que necessitavam de melhoria. Em 1991, o SEI criou o CMM, como uma evolução deste ambiente. A partir do lançamento e divulgação da versão 1.1 do CMM, em 1993, o tema de melhoria do processo foi ganhando força. Esta força foi consequência dos resultados práticos obtidos pelas organizações que realizaram programas de melhoria com o CMM como modelo de referência (GONÇALVES, 2001) (CARIOSA, 2003 p.34).

Desde sua primeira versão, lançada em 1991, na esteira de sucesso do SW-CMM, diversos outros modelos foram criados visando cobrir outras áreas de interesse, como por exemplo: - SA-CMM - *Software Acquisition* CMM: usado para avaliar a maturidade de uma organização em seus processos de seleção, compra e instalação de software desenvolvido por terceiros.

- SE-CMM - *System Engineering* CMM: avalia a maturidade da organização em seus processos de engenharia de sistemas, concebidos como algo maior que o software. Um sistema inclui o software, o hardware e quaisquer outros elementos que participam do produto completo.
- IPD-CMM - *Integrated Product Development* CMM: ainda mais abrangente que o SE-CMM, inclui também outros processos necessários à produção e suporte ao produto, tais como suporte ao cliente e processos de fabricação.
- P-CMM - *People CMM*: avalia a maturidade da organização em seus processos de administração de recursos humanos no que se refere a software, recrutamento e seleção de profissionais e treinamento.

O surgimento de todos estes outros modelos acabou trazendo alguns problemas. Como nem todos os modelos usavam a mesma terminologia, um mesmo conceito podia receber nomes diferentes em cada modelo. Os modelos tinham diferentes números de níveis ou formas diferentes de avaliar o processo. Outro problema era os altos custos de treinamento, avaliação e harmonização para organizações que quisessem usar mais de um modelo. Assim, tornou-se necessária uma padronização. Além disso, a experiência no uso de SW-CMM, durante uma década, serviu para identificar pontos em que o modelo poderia ser melhorado. O surgimento do projeto SPICE levou à necessidade de tornar o CMM compatível com a norma ISO 15504 (1997), resultado deste projeto.

Por estas razões, o SEI iniciou um projeto chamado CMMI - CMM *Integration*. Seu objetivo era gerar uma nova versão do CMM que resolvesse esses problemas. Concretamente, a primeira ideia, como o nome sugere, foi integrar os diversos CMMs em uma estrutura única, todos com a mesma terminologia, processos de avaliação e estrutura. O projeto se preocupou ainda em tornar o CMM compatível com a norma ISO/IEC 15504, de modo que avaliações em um modelo sejam reconhecidas como equivalentes aos do outro. E, naturalmente, incorporar ao CMM as sugestões de melhoria surgidas ao longo dos anos.

O Modelo de Maturidade da Capacidade Integrado, ou CMMI, foi resultado da integração dos modelos (CARIOSA, 2003):

- SW-CMM V2C - *Capability Maturity Model for Software V2.0, draft C.*

- SECM - *EIA Interim Standard 731 – System Engineering Capability Model*.
- IPD-CMM - *Integrated Product Development Capability Maturity Model, draft 0.98.*

O CMMI é claramente uma melhoria sobre o CMM. No entanto, apesar das melhorias incorporadas ao CMMI, seu precursor - o CMM - continua sendo um modelo bastante eficiente, (CARIOSA, 2003 p.41e 42).

3.2. CMMI (*Integration*)

O CMMI, que é o sucessor do modelo de maturidade (CMM) ou CMM Software, como já foi descrito. O CMMI foi desenvolvido no período de 1987 até 1997. Em 2002, foi lançada a versão 1.1 do CMMI, seguido da versão 1.2 em agosto de 2006, e CMMI versão 1.3 em novembro de 2010. Algumas das principais mudanças no CMMI versão 1.3 são o apoio de *Agile Software Development* (Desenvolvimento de ágil de Software), e melhorias nas práticas de maturidade, assim como, o alinhamento da representação (estágio e contínua).

Representação Continua: Mais flexível, porém mais complexa de administrar. Permite a seleção da ordem de melhoria dos processos que melhor se adéqua aos objetivos de negócio da organização, além de permitir que sejam feitas comparações entre áreas de processo entre diferentes organizações ou através dos resultados apresentados de acordo com a equivalência de estágios. Possui uma estrutura compatível com padrão ISO/IEC 15504.

Representação por Estágios: organiza as áreas de processo em cinco níveis de maturidade. Esses níveis de maturidade servem para dar suporte e orientar a melhoria do processo, indicando quais áreas estabelecer para atingir cada nível.

O CMMI é originado em engenharia de software, mas tem sido muito generalizado ao longo dos anos para abraçar outras áreas de interesse, tais como o desenvolvimento de produtos de hardware, a entrega de todos os tipos de serviços e aquisição de produtos e serviços. A palavra "software" não aparece nas definições de CMMI. Esta generalização de conceitos de melhoria faz do CMMI extremamente abstrato. Não é tão específico para engenharia de software como o seu antecessor, o Software CMM. O CMMI atualmente aborda três áreas de interesse:

Produtos e serviços de desenvolvimento - *CMMI for development* (CMMI-DEV);

Gestão de estabelecimento de serviços - *CMMI for Services* (CMMI-SVC);

Aquisição de produtos e serviços - *CMMI for Acquisition* (CMMI-ACQ).

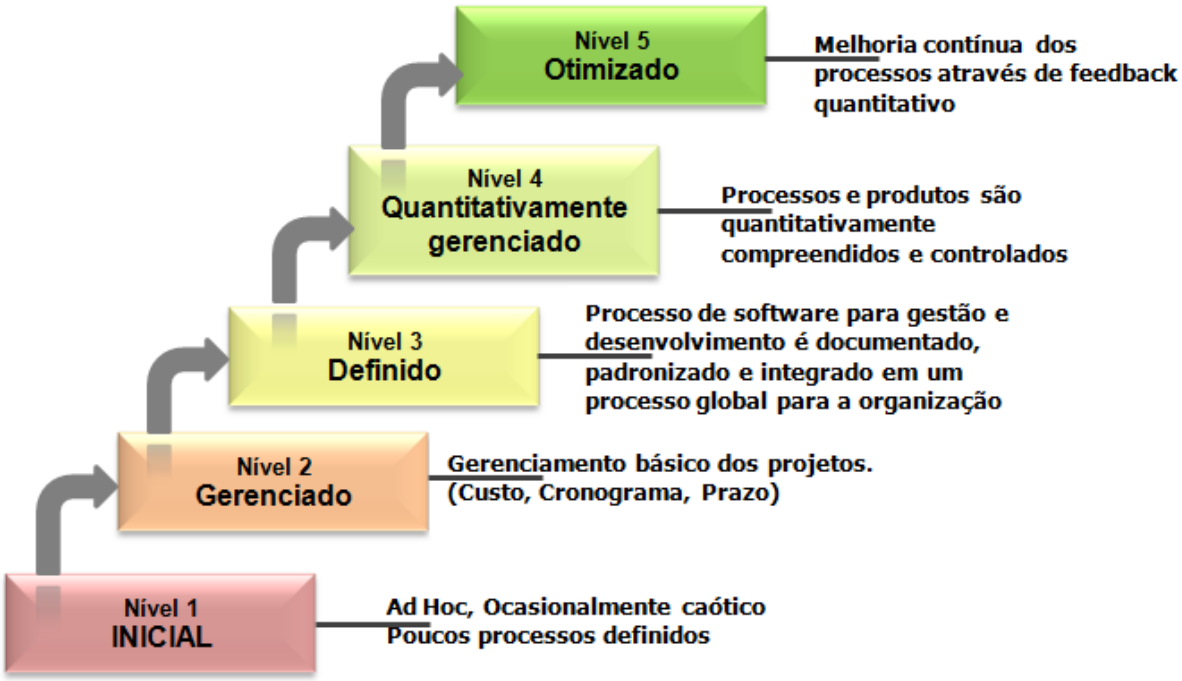
O modelo com maior aderência para empresas de desenvolvimento de software é o CMMI-DEV, pois suporta duas representações: contínua e em estágios, sendo considerado também um modelo flexível. A primeira representação é voltada para avaliação e definição do nível de capacidade de processos escolhidos para serem avaliados em uma determinada unidade organizacional, enquanto a segunda representação é voltada à avaliação de um conjunto de processos pré-definidos em cada nível de maturidade do modelo CMMI (SEI, 2006).

CMMI-DEV (*for development*)

O modelo CMMI-DEV V1.3 (CMMI, 2010) é um conjunto de melhores práticas de desenvolvimento usado pelo governo (Governo dos Estados Unidos da América) e da indústria que é gerado a partir do CMMI V1.3 Arquitetura e Framework. O Framework CMMI é a estrutura básica que organiza componentes CMMI e as combina em constelações e modelos CMMI. Uma constelação é uma coleção de componentes CMMI que são usados para construir modelos, materiais de treinamento e documentos de avaliação relacionada para uma área de interesse (por exemplo, desenvolvimento, aquisição, serviços).

O CMMI for *development* fornece orientação necessária para aplicar as melhores práticas do CMMI em uma organização, para se concentrar em atividades de desenvolvimento de produtos e serviços de qualidade para atender às necessidades dos clientes e usuários finais.

Este modelo é baseado na CMMI *Model Foundation* ou CMF (ou seja, os componentes do



modelo comuns a todos os modelos CMMI e constelações) e incorpora o trabalho das organizações de desenvolvimento para adaptar CMMI para uso no desenvolvimento de produtos e serviços.

Ele contém 22 áreas de processo. Dessas áreas de processos, 16 são áreas de processo do núcleo (core), 01 é uma área de processo compartilhado e 05 são áreas específicas do processo de desenvolvimento. Uma área de processo principal é uma área de processo que é comum a todos os modelos CMMI. Uma área de processo comum é compartilhada por pelo menos dois modelos CMMI, mas não todas elas.

É um modelo de referência abrange as atividades para o desenvolvimento de produtos e serviços em diversas organizações do setor industrial, incluindo aeroespacial, bancário, hardware, software, de defesa, indústria automobilística e de telecomunicações e contém práticas que abrangem gestão de projetos, gestão de processos, engenharia de sistemas, engenharia de hardware, engenharia de software, e de outros processos de apoio utilizadas no desenvolvimento e manutenção (CMMI, 2010).

3.2.1. NÍVEIS DE MATURIDADES CONTIDAS NO CMMI-DEV

Antes de abordar a diretamente a área de processo de Garantia da Qualidade (PPQA) é importante apresentar de forma sucinta a estrutura do CMMI-DEV. No CMMI-DEV há 05 (cinco) níveis de maturidade. No entanto, as avaliações dos níveis de maturidade são concedidas para níveis de 02 a 05 (cinco). Cada nível de maturidade agrega um conjunto de áreas de processos que devem ser praticados para se alcançar o nível de maturidade.

Figura 3. Níveis de maturidade do CMMI. Adaptado de CMMI (2010)

ÁREAS DE PROCESSOS AGRUPADAS POR NÍVEL DE MATURIDADE

Nível	Sigla em Inglês	Descrição
2	CM	Gestão da configuração
	MA	- Medição e Análise
	PMC	- Projeto de Monitoramento e Controle
	PP	- Planejamento de Projetos
	PPQA	**- Garantia da qualidade do Processo e Produto**
	REQM	- Gerenciamento de Requisitos
	SAM	- Gerenciamento de acordo com o fornecedor

3	DAR	- Análise de decisão e resolução
	IPM	- Gestão Integrada de Projetos
	OPD	- Definição do Processo Organizacional
	OPF	- Foco no Processo Organizacional
	OT	- Treinamento Organizacional
	PI	- Integração de Produto
	RD	-Desenvolvimento de Requisitos
	RSKM	- Gestão de Riscos
	TS	- Solução Técnica
	VAL	- Validação
	VER	- Verificação
4	OPP	- Desempenho do Processo Organizacional
	QPM	- Projeto quantitativamente gerenciado
5	CAR	- Análise e Resolução de Causas
	OPM	- Gestão de Desempenho Organizacional

Tabela 1. Áreas de processo agrupadas por nível. Adaptado de CMMI (2010)

Área de processo é um grupo de práticas relacionadas com um processo específico que quando executado satisfaz um conjunto de objetivos considerados importantes para a melhoria deste processo (SEI, 2006). As áreas de processo possuem componentes que possuem objetivos e práticas, específicas, genéricas ou informativas.

Práticas Específicas – (GP *Generic Practices*): atividades que são consideradas importantes na satisfação de uma meta específica associada.

Práticas genéricas - (SP *Spedific Practices*): oferecem uma institucionalização que assegura que os processos associados com a PA serão eficientes, repetíveis e duráveis.

Objetivos Genéricos - (GG *Generic Goals*): Aparecem em diversas áreas de processo.

Objetivos específicos – (SG *Specific Goals*): Aparecem numa área processo

Informativas: Produtos de trabalho típicos, exemplos de saídas de uma prática específica ou genérica. Práticas secundárias: descrições detalhadas que fornecem um direcionamento para a interpretação de práticas específicas ou genéricas.

Não será descrito neste trabalho, as práticas e objetivos genéricos a todas as áreas, tendo em vista que há uma grande e vasta descrição em detalhes presente na literatura (GONÇALVES, 2001), (CARIOSA, 2003), (SEI, 2006), (CMMI, 2010) e não é o objetivo central deste trabalho.

3.2.2. GARANTIA DA QUALIDADE DO PROCESSO E PRODUTO, (PPQA).

A área de processo de garantia da qualidade do produto e processo do CMMI-DEV, (*Process and Product Quality Assurance* - **PPQA**) está contigo dentro do nível de maturidade 02, ou seja, o primeiro nível a ser buscado.

IV. CONCLUSÃO

Neste trabalho, foram apresentados aspectos positivos contidos dentro da orientação objetos que possibilitam e promovem o reúso de software. Foram apresentados conceitos como: herança e polimorfismo, *Design Patterns, Frameworks*. Dentro do cenário do atual sucesso absoluto das linguagens e plataformas orientadas a objetos e importante reconhecer os fatos que permitiram chegar ao contexto atual. O reúso é sem dúvidas uma das principais características que pavimentaram, e permitiram o avanço destas linguagens. Neste trabalho foi utilizada como exemplo a linguagem e plataforma Java, presente em grande parte das aplicações de software existentes nos dias de hoje.

REFERÊNCIAS

[1] BARBACCI, Mario; Klein, Mark; Longstaff, Thomas; & Weinstock, Charles. *Quality Attributes* (CMU/SEI-95-TR-021). Software Engineering Institute, Carnegie Mellon University, 1995. disponível http://resources.sei.cmu.edu/library/asset-

view.cfm?AssetID=12433 acessado 08 de março, 2015.

[2] BARTIÉ, Alexandre. Garantia da qualidade de software: adquirindo maturidade organizacional / Alexandre Bartié - Rio de Janeiro : Elisevier 2002.

[3] BEHKAMAL, Behshid., Mohsen Kahani, Mohammad Kazem Akbari. *Customizing ISO 9126 quality model for evaluation of B2B applications*. Copyright 2008 Elsevier B.V. September 2008, Setembro 2008.

[4] BOTELLA P., X. Burgues, J.P. Carvallo, X. Franch, C. Quer, *Using Quality Models for Assessing COTS Selection*, in: Proceeding of MPEC'04 and ICSE'04, 2004.

[5] CAROSIA, J. S. Levantamento da qualidade do processo de software com foco em pequenas organizações / J. S. Carosia. – São José dos Campos: INPE, 2003.

[6] CESAR. Centro de Estudos e Sistemas Avançados do Recife - Informática Brasileira em Análise - ano 1, número 2, junho de 1997.

[7] CMMI, *CMMI Product Team, "CMMI for Development, Version 1.3," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Technical Report* CMU/SEI-2010-TR-033, 2010. Disponível em http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9661. Acessado em 13 Março 2015.

[8] GONÇALVES, J. M; Villas Boas, A. Modelo de maturidade da capacidade de software (CMM). Campinas: Fundação CPqD. Versão 1.2. Tradução não oficial do CMU/SEI-93-TR-24-(CMM V1.1), 2001. 60 p.

[9] ISO/IEC 25010. *Systems and software engineering - Systems and software Quality Requirements and Evaluation* (SQuaRE) - *System and software quality models*, 2011. Disponível em http://www.iso.org/iso/catalogue_detail.htm?csnumber=35733, acessado em 14 Março 2015.

[10] KOCUR, George. 1.264J *Database, Internet, and Systems Integration Technologies*, Fall 2013. Massachusetts Institute of Technology: MIT OpenCourseWare, disponível em http://ocw.mit.edu, acessado 07 de março, 2015.

[11] MACCALL, J., P. Richards e G. Walters, "*Factors in Software Quality*", três volumes, NTIS AD-AO49-014, 015, 055. Novembro 1977.

[12] MEYER, B., *Object-oriented Software Construction*, Prentice-Hall, 1988.

[13] MR-MPS-SW:2012. MPS.BR - Melhoria de Processo do Software Brasileiro. Guia Geral MPS de Software. Copyright 2012 - SOFTEX, 2012.

[14] PÁDUA, Wilson de P.F. Engenharia de Software: fundamentos, métodos e padrões, LTC Livros Técnicos e Científicos, 2003.

[15] PAULK, M. C.; Curtis, B; Chississ, M; Weber, C. V. *Capability maturity model version* 1.1. Pittsburgh: Software Engineering Institute, Carnegie Mellon University. Technical Report (CMU/SEI-93-TR-024/ESC-TR-93-177), Fevereiro 1993. 82 p.

[16] PRESSMAN, Roger S. Engenharia de Software: José Carlos Barbosa dos Santos - Sao Paulo : Person Makron Books, 1995.

[17] WATSON, Mike. *Managing Smaller Projects: A Practical Approach*. Multi-Media Publications Inc., 2006.