Uma Visão Geral sobre a da Qualidade de Software

Álvaro Augusto Neto Nilson Sant'Anna Germano Kienbaum
Laboratório de Computação e Matemática Aplicada
Instituto Nacional de Pesquisas Espaciais
alvaro@comp.ita.br nilson@lac.inpe.br germano@lac.inpe.br

Resumo

Este trabalho apresenta uma visão geral sobre a qualidade de software. Inicialmente apresentam-se os principais conceitos envolvidos e como eles evoluíram ao longo do tempo. Em seguida, mostram-se as características dos principais modelos utilizados atualmente, pelo setor aeroespacial, para gerenciar a qualidade de software, representados pelo CMM e CMMII do SEI, pelas Normas ISO 15504 e pela Norma ECSS–Q–80B da European Cooperation for Space Standardization. Finalmente, descrevem-se as características das de algumas metodologias utilizadas para melhoria dos processos de produção de software e apresenta-se uma proposta para sua evolução.

Palavras-chave: Qualidade de Software, Modelagem de Processos, Simulação de Processos.

1. Introdução

O desenvolvimento de software é uma atividade que apresenta importância sócio-econômica crescente, na medida em que ele está sendo incorporado a quase todos os produtos e atividades da sociedade moderna.

Nos dias de hoje, produzir software com alta qualidade e produtividade é considerado um objetivo estratégico a ser perseguido por muitas organizações, que relacionam o sucesso no alcance deste objetivo como um dos fatores críticos para o sucesso nas suas áreas de negócios [1].

Apesar de apresentar este crescimento significativo, observa-se que os produtos de software nem sempre atendem satisfatoriamente as necessidades de seus usuários, principalmente com relação a três elementos distintos associados à qualidade, prazos de desenvolvimento e custos de produção e manutenção dos projetos [2], [3], [4].

As questões relacionadas com a melhoria da qualidade de software acompanham o seu desenvolvimento desde o princípio. Na célebre conferência patrocinada pelo Comitê de Ciências da North Atlantic Treaty Organisation-NATO em 1968, Fritz Bauer propôs o seguinte enfoque para as atividades de desenvolvimento de software: "A criação e utilização de sólidos princípios de engenharia a fim de obter software de maneira econômica, que seja confiável e que trabalhe eficientemente em máquinas reais" [5], [apud 6]. Como se pode verificar, algumas questões típicas da área de qualidade, tais como confiabilidade e eficiência já eram abordadas desde a conferência onde a expressão engenharia de software foi consolidada. Observa-se que apesar do grande progresso alcançado desde então, algumas das questões básicas daquela época permanecem em voga até os dias de hoje.

2. Contextualização da qualidade de software

O conceito de qualidade é bastante abstrato e multifacetado. Sua conceituação precisa apresenta dificuldades pelo fato do termo ser empregado com muitos sentidos, além de ter recebido múltiplas abordagens ao longo do tempo. Procura-se a seguir contextualizá-lo no âmbito deste trabalho, que envolve as atividades envolvidas com a produção e manutenção de software.

Do ponto de vista do produtor, a qualidade associa-se à concepção e produção de um produto que vá de encontro às necessidades do cliente. Do ponto de vista do cliente, a qualidade está associada ao valor e à utilidade reconhecida nos produtos, sendo que em alguns casos este fato pode estar associado ao seu preço. Neste caso, deve-se atentar para não confundir qualidade com o que Crosby [7] chama de "douração" de um produto ou serviço, representado pelo acréscimo desnecessário de componentes associados ao luxo, sofisticação, ostentação, etc. com o objetivo de valorizá-lo perante seus consumidores.

Observa-se que as questões ligadas à gerência da qualidade têm sido enfocadas sob diversos aspectos, tais como as características do produto ou serviço executado; as particularidades do processo de produção utilizado; os aspectos mercadológicos associados ao grau de satisfação dos usuários; as práticas de engenharia; o controle da produção e a gerência das operações. Como resultado, pôde-se observar uma série de perspectivas competindo entre si, cada uma delas empregando um esquema de análise diferente e utilizando-se de sua própria terminologia.

Segundo Garvin [8], os estudiosos das áreas de filosofia, economia, marketing e gerência de operações, vêm analisando a qualidade, focalizando sua abordagem sob um ponto de vista diferente. A filosofia tem se concentrado nas questões de definição; a economia, na maximização dos lucros e no equilíbrio de mercado; o marketing, nos determinantes do comportamento dos compradores e na satisfação dos clientes; e a gerência de operações, nas práticas de engenharia e no controle da produção.

O autor identifica cinco abordagens principais para definição da qualidade [8]: a transcendente, a baseada no produto, a baseada no usuário, a baseada na produção e a baseada no valor. Apesar do potencial conflito, afirma que as empresas podem tirar proveito dessas perspectivas múltiplas, pois confiar em uma única definição da qualidade pode representar uma visão simplista da realidade empresarial.

Ao longo do tempo a gestão da qualidade nas organizações evoluiu de uma restrita disciplina técnica, cujo objetivo era detectar problemas de fabricação, para um campo muito mais amplo, que abrange todo o espectro das atividades empresariais, indo desde a concepção e projeto de um produto, até sua assistência pós-venda e disposição final. A responsabilidade pela sua implementação transferiu-se paulatinamente do departamento de inspeção, para os grupos de engenharia e produção, indo culminar na sua condução pela alta gerência. Também nesse período, houve incrementos progressivos das atividades de medição, controle, análise e planejamento da qualidade.

No entanto, é necessário enfatizar que a justificativa estratégica para a melhoria da qualidade tem sido o fator de atração para toda esta evolução. Ela se encontra na busca de objetivos empresariais básicos, como redução de custos, aumento da participação no mercado e aumento da lucratividade [3]. No âmbito do desenvolvimento de software, em escala empresarial, pode-se observar a adoção de um enfoque semelhante.

2.1. Enfoque adotado para a qualidade de software

Neste trabalho aborda-se a produção de software sob um enfoque de engenharia, com o objetivo de estabelecer um maior embasamento técnico para a proposição de melhorias nos produtos e processos envolvidos na sua elaboração. Dentro desta linha de raciocínio, seu ciclo de produção, como dos demais produtos de engenharia, envolve [9], [10]:

- Análise e definição dos problemas a serem resolvidos;
- Desenvolvimento do projeto e análise das soluções propostas;
- Execução dessas soluções;
- Verificação da sua adequabilidade e conformidade;
- Gerenciamento de todos os elementos técnicos, humanos, sociais e financeiros envolvidos na produção.

A adoção desta abordagem traz como conseqüência a necessidade de se formular os requisitos do sistema desde o início do processo de produção, de forma que ao seu final eles possam ser verificados e validados. Também acarreta na adoção de processos de produção cujas etapas devem ser formalmente planejadas, definidas e implementadas, de maneira que seja possível verificar continuamente a sua conformidade e adequação.

Esta visão do problema vai de encontro aos preceitos de Crosby [11], que ao formular o seu "primeiro princípio absoluto da qualidade", afirmou taxativamente: "a qualidade deve ser definida como cumprimento dos requisitos, não como adequação".

Um enfoque semelhante também é utilizado pela Norma ISO 9000:2000 [12], que define qualidade como sendo o "o grau no qual um conjunto de características inerentes satisfaz a requisitos". Essas características são definidas pela Norma como propriedades diferenciadoras e o fato de serem inerentes é empregado com o sentido contrário de atribuído [12], ou seja, com a intenção de consolidar a idéia da existência de características diferenciadoras permanentemente capazes de satisfazer aos requisitos.

De maneira similar, a NBR 13596 [13], que foi baseada na Norma ISO 9126, define como características de qualidade de software o "Conjunto de atributos de um produto de software através do qual sua qualidade é descrita e avaliada. Uma característica de qualidade de software pode ser detalhada em múltiplos níveis de subcaracterísticas".

A estas abordagens sobrepõe-se ainda a questão econômica, tratada por Boehm e Papacio [14], que constataram que o custo para identificar e resolver um problema na etapa de identificação de requisitos salta de hipotéticos US$ 1,00, para US$ 5,00 se a solução ocorrer na fase de projeto, US$ 10,00 se ela ocorrer na fase de codificação, US$ 20,00 se ela ocorrer durante os testes e, US$ 200,00 se ela só ocorrer depois do sistema ter sido entregue.

Este fato também foi explorado no "modelo de amplificação de defeitos", apresentado pela IBM [apud 6] onde, através de uma série de dados coletados em grandes projetos de desenvolvimento, demonstra-se que o custo de correção dos erros cresce exponencialmente, conforme o desenvolvimento de software avança ao longo do seu ciclo de produção. Assim, um erro ocorrido na fase de projeto e cuja correção custasse uma unidade monetária, poderá ter este valor ampliado para 6,5 unidades, se sua detecção só ocorrer antes da fase de

testes; 15 unidades se a descoberta ocorrer durante os testes, e entre 60 e 100 unidades, se sua ocorrência só for percebida após ter sido liberado para os usuários finais.

Da mesma forma, este modelo demonstra que num caso hipotético, onde apenas a metade dos erros ocorridos numa fase do ciclo de produção seja capaz de ocasionar novos erros na fase seguinte, os custos de desenvolvimento podem saltar de 783 unidades monetárias, quando as revisões técnicas são efetuadas, para 2177 unidades, quando elas não são executadas. Pressman sintetiza este fato através da máxima: "pague agora ou pague depois muito mais" [6].

Sendo assim, verifica-se que a abordagem que trata qualidade como conformidade com os requisitos, encontra respaldo técnico e econômico que justifica sua adoção em consonância com os objetivos deste trabalho.

3. Principais motivações para a evolução da qualidade de software

Nos últimos anos observou-se uma grande difusão de sistemas computadorizados que foram incorporados a sistemas maiores, cuja função principal não é a realização de computações [15]. Esta classe de aplicações, que usualmente recebem a denominação de sistemas computadorizados embutidos ("*embedded computer system*"), tem sido empregadas em numerosas áreas da indústria, possibilitando que os sistemas resultantes apresentem capacidades que até então eram quase que exclusivas dos seres humanos, tais como o comando, o controle, a decisão e a percepção [15].

Este movimento teve início na área aeroespacial, razão pela qual o software necessário para operar engenhos deste tipo é usualmente chamado de *software embarcado* [16]. Uma categoria especial dos sistemas embutidos é composta pelos chamados sistemas críticos [9], nos quais a ocorrência de falhas pode causar perdas de vidas humanas, danos físicos significativos e prejuízos econômicos elevados [17]. Para esta categoria de sistemas existem algumas características da qualidade, tais como a confiabilidade, disponibilidade, integridade e segurança, que exigem uma abordagem bastante rígida e baseada em métodos eficientes e amplamente comprovados, de forma a obter continuamente resultados estáveis com relação ao seu funcionamento.

De modo geral, os progressos obtidos na gestão da qualidade no desenvolvimento desta classe de aplicações têm impulsionado os principais progressos tecnológicos na gestão da qualidade de software [18].

Outro fator que tem impulsionado os principais avanços com relação a qualidade tem sido a ocorrência cada vez mais freqüente de acidentes e incidentes de todos os tipos, ocasionados por falhas de software.

Relatos nesse sentido têm apresentado situações complexas e custosas, como as dificuldades encontradas pelo principal fabricante mundial de carros híbridos, que teve de substituir um software instalado em mais de 20.000 veículos, devido a problemas intermitentes que causavam o acendimento indevido de luzes de advertência e a paralisação intempestiva do motor [19]. Em outros casos, elas envolvem situações verdadeiramente catastróficas, como a explosão do foguete Ariane 5, logo após o seu lançamento, causando a perda de quatro satélites e prejuízos superiores a quinhentos milhões de dólares [20], [19].

No ambiente de negócios os problemas desta natureza também são comuns e têm causado prejuízos financeiros significativos, como o ocorrido com um dos principais bancos dos Estados Unidos, que creditou indevidamente mais de 920 milhões de dólares na conta de 823 clientes [19].

Em casos extremos as falhas ou interrupções no funcionamento de sistemas baseados em software podem inviabilizar a própria existência de um negócio como, por exemplo, a interrupção das vendas em um site de comércio eletrônico, ou a paralisação da produção de uma empresa, cujos principais processos tenham sido automatizados através de sistemas computadorizados. Em maio de 2004, problemas desse tipo fizeram com que fosse necessário desabilitar simultaneamente um software utilizado para gerência de sites, até que o problema fosse corrigido [21]. Os prejuízos foram enormes, pois o produto era utilizado por companhias que representam uma porcentagem significativa do tráfego da internet.

Dificuldades desta natureza são cada vez menos toleradas pela sociedade moderna, que exige, cada vez mais, que o progresso tecnológico represente reais melhorias com relação à segurança, confiabilidade, precisão, exatidão, etc. dos sistemas computadorizados que utiliza regularmente.

Como a sociedade globalizada do final do século XX e início do século XXI depende de maneira intrínseca dos softwares para gerir suas atividades técnicas, de negócios, e até mesmo, o seu dia-a-dia, verifica-se que as exigências por maior qualidade de software tendem a aumentar nos próximos anos. Superar este gargalo constitui-se num dos maiores desafios tecnológicos da atualidade.

4. Abordagens para a qualidade de software

Ao longo dos anos foram desenvolvidas diversas abordagens para os problemas relacionados com a qualidade de software. As mais difundidas envolvem melhorias nos produtos de software e no seu processo de produção.

De maneira complementar a este enfoque,

Kitchenham e Pfleeger [22] propuseram que a questão da qualidade de software fosse analisada de pelo menos três maneiras: a qualidade do produto, a qualidade do processo que resulta no produto, e a qualidade do produto no contexto do ambiente de negócios onde ele será utilizado [23].

Mais recentemente considera-se que devido a certas características peculiares dos softwares, tais como o seu caráter imaterial e intangível [24], [16], a complexidade e variabilidade do seu processo de produção, torna-se necessário o desenvolvimento de processos de gerenciamento da produção específicos, diferentes, portanto, dos demais produtos e serviços [25].

Neste trabalho propõe-se o enfoque do tema sob três pontos de vista complementares:

a) O primeiro, utiliza uma abordagem centrada na engenharia do produto e enfoca a questão da qualidade dos produtos de software sob a perspectiva das características técnicas observáveis no produto final. Um paralelo desta abordagem, quando aplicada à indústria automobilística, envolveria o aprimoramento da qualidade dos automóveis produzidos.

b) O segundo, emprega uma abordagem centrada na engenharia de produção e envolve as melhorias nos processos utilizados para produzir software. Se comparada com a indústria automobilística, ela trataria das melhorias nas linhas de produção, de maneira a torná-las capazes de produzir automóveis com menos defeitos de fabricação (embora pudessem apresentar erros de projeto).

c) O terceiro, faz uso de uma abordagem centrada no gerenciamento da produção e focaliza os aspectos ligados ao planejamento e controle da produção, envolvendo os insumos utilizados, o gerenciamento das pessoas envolvidas, a organização do ambiente produtivo, a gestão da tecnologia utilizada e a estruturação e organização do trabalho [47]. Na indústria automobilística esta abordagem corresponderia ao setor administrativo da fábrica.

A seguir apresentam-se as principais características encontradas em cada uma dessas abordagens:

4.1. Abordagem centrada na engenharia do produto

As primeiras abordagens bastante difundidas sobre qualidade de software [26], [27] foram propostas por Boehm [28] e McCall [29].

O trabalho de Boehm [28] propôs um conjunto de definições e medidas para um conjunto de atributos de um produto de software. A sistemática proposta enfoca a avaliação de um conjunto de propriedades que um produto de software deve apresentar, de tal forma que elas permitam avaliar quantitativamente em que ponto ele se encontra dentro de um conjunto de estágios sucessivos, representado pela progressão de um determinado "índice" de qualidade. Essas características em grande parte foram associadas a certas peculiaridades apresentadas pelo código-fonte, de maneira a permitir sua fácil mensuração. Assim, as características avaliadas por Boehm estão intrinsecamente ligadas aos detalhes do produto, sob o ponto de vista de seus usuários, desenvolvedores e mantenedores.

O modelo de Boehm aborda um aspecto importante que é representado pela necessidade de melhoria da "Engenharia do Produto de Software", ou seja, das características relacionadas com o aperfeiçoamento de certas peculiaridades do produto que permitam a produção de softwares melhores, mais eficientes e fáceis de usar ou manter.

A principal crítica a esta abordagem refere-se à forma de implementar proposta pelo autor, que associa a qualidade do produto, com a qualidade do seu código-fonte. Com isto despreza-se uma imensa gama de outras características que um software de qualidade deve apresentar. As definições sobre o que é software, apresentadas nos principais livros da área, envolvem muito mais itens do que apenas o seu código-fonte [23], [6], [9]. Verifica-se portanto, a prática de tentar analisar o todo através de apenas uma de suas partes.

No dia-a-dia esta abordagem pode ser observada através do desenvolvimento de tecnologias que têm propiciado a melhoria nas interfaces homem-máquina, sistemas multimídia, facilidades para manutenção e atualização dos produtos, aumento da robustez e tolerância à falhas, desenvolvimento de compiladores que produzem código mais eficiente e preciso, desenvolvimento de protocolos e padrões para interface entre sistemas, etc.

Dentro deste enfoque observa-se que as melhorias da qualidade na "Engenharia do Produto de Software" permanecem associadas à própria evolução da tecnologia de software e da informática como um todo.

4.2. Abordagem centrada na engenharia da produção

A segunda abordagem para a questão da qualidade de software envolve características associadas à "Engenharia da Produção de Software", representadas pelo aperfeiçoamento das atividades realizadas durante o seu processo de desenvolvimento, visando à produção de softwares com maior qualidade.

4.2.1. Modelo de McCall

O primeiro trabalho a utilizar esta abordagem foi proposto por McCall [29] e, segundo o autor, era especificamente orientado para o desenvolvimento de

aplicações na área aeroespacial [29].

Nessa abordagem ele formulou um modelo cujo objetivo era relacionar onze fatores representativos da qualidade de software, sob o ponto de vista de seus usuários, com determinados critérios quantificáveis do seu processo de produção [27]. Esses fatores foram agrupados segundo três dimensões relacionadas respectivamente com a operação, manutenção e migração de um produto de software, conforme pode ser visualizado na Figura 1.

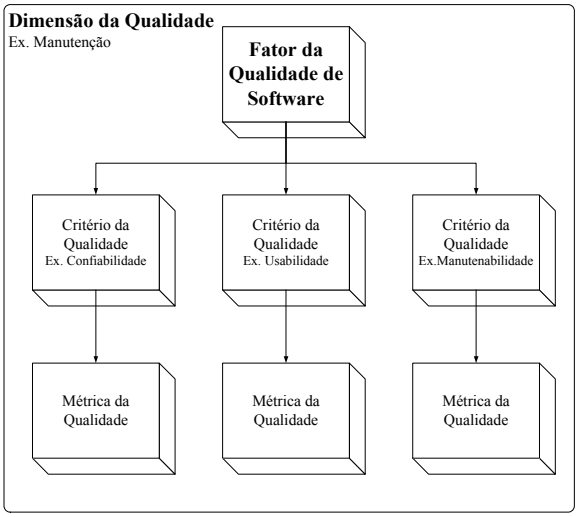

Fig. 1 – Modelo hierárquico de McCall

As condições ou características que contribuem para a qualidade do produto receberam a denominação de fatores da qualidade e se relacionam com diversos critérios mensuráveis de maneira direta ou indireta

Ao propor um conjunto de critérios independentes dos atributos do produto, ou do processo de produção utilizado, McCall introduziu uma visão gerencial para a questão da qualidade de software, que possibilitou julgar, avaliar quantitativamente a qualidade de um produto [29]. As métricas estabelecidas possibilitaram avaliá-la em todas as fases do desenvolvimento, de forma a prover os gerentes de desenvolvimento de um indicador dos progressos alcançados.

As principais críticas que podem ser feitas a esta sistemática relacionam-se com as dificuldades para encontrar métricas capazes de avaliar corretamente características subjetivas como usabilidade, manutenibilidade, flexibilidade, etc. Para isso foram utilizadas medidas indiretas de propriedades não diretamente mensuráveis, o que muitas vezes pode não refletir adequadamente o fenômeno observado [30]. Em outros casos, a obtenção de valores confiáveis depende da realização de tarefas intensivas, cujo custo torna-se proibitivo quando comparado aos benefícios que podem ser obtidos [31].

4.3. Modelo de maturidade da capacitação para software

As abordagens voltadas para a melhoria dos processos de produção tiveram grande progresso com o trabalho de Humphrey [32], [33], que desenvolveu inicialmente uma sistemática com o intuito de avaliar de maneira objetiva os diferentes estágios de progresso em que se situavam as organizações produtoras de software. A estrutura (*framework*) proposta para esta avaliação possibilitou um enfoque gerencial que permitiu aos produtores identificar as áreas prioritárias para melhoria dos processos executados. Além deste enfoque gerencial, Humphrey também salientou a importância de duas outras áreas para o aperfeiçoamento da qualidade: o talento das pessoas envolvidas no processo de produção e a excelência do projeto (*design*) adotado [33].

A abordagem centrada na melhoria dos processos e das pessoas adotada por Humphrey evoluiu rapidamente para o Modelo de Maturidade da Capacitação para Software (Capability Maturity Model – CMM) [34], [35] desenvolvido pelo Software Engineering Institute - SEI da Carnegie Mellon University - CMU, sob encomenda do Departamento de Defesa dos EUA.

O CMM surgiu da necessidade do governo federal dos EUA de criar uma sistemática que permitisse avaliar a capacitação de seus fornecedores de software. Anteriormente o SEI já havia publicado um método para avaliar o nível dos processos de desenvolvimento utilizados por uma organização e outro, para identificar fornecedores qualificados para realizar o desenvolvimento. Após quatro anos de experiência na aplicação desses procedimentos, o SEI evoluiu esses métodos e pôs em prática o CMM.

Para as organizações que contratam o desenvolvimento externo de software, o CMM tem sido visto como um auxílio eficiente para se avaliar o nível de capacitação de seus fornecedores [34]. As empresas produtoras de software passaram a empregar o CMM como um modelo no qual procuravam se basear para melhorar seus processos, visando atender os requisitos de seus clientes internos e externos.

O CMM pode ser considerado como um modelo de Sistema da Qualidade que descreve os princípios e práticas utilizados na avaliação da maturidade e na melhoria de processos de software, através da identificação das áreas-chave de processo (*Key Process Areas-KPAs*) das empresas desenvolvedoras de software.

O CMM caracteriza-se por definir 5 níveis definidos de maturidade, apresentados sucintamente na Figura 2. Ao determinar o nível em que se situam

os processos de uma equipe de desenvolvimento, ele permite identificar os seus pontos fortes e fracos e aponta um roteiro para implementação das melhorias necessárias para atingir o próximo nível.

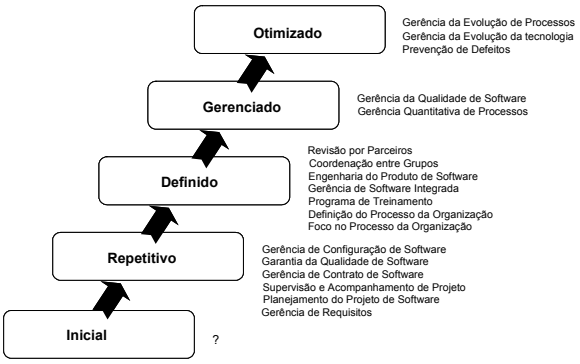

Fig. 2 - CMM-Áreas-chave do processo

O CMM foi bastante usado como o critério de seleção de fornecedores de software nos EUA, já que se trata de um modelo preciso, detalhado e específico. Além disso, desde o início de sua aplicação foram mencionados ganhos de competitividade como conseqüência direta da sua implementação nas organizações [64]. Anualmente o SEI publica relatórios que apresentam dados estatísticos sobre a aplicação mundial do modelo. Embora demonstrem uma aceitação crescente, eles carecem de uma análise mais detalhada com relação ao retorno sobre o investimento feito para certificação segundo o modelo [36].

Neste ponto residem as maiores críticas ao CMM: o processo de certificação é demorado, caro e implementa uma série de controles burocráticos, que muitas vezes acabam por dificultar a implantação de iniciativas criativas e inovadoras para melhoria dos processos de produção. Outra crítica freqüente refere-se ao fato do modelo ser empírico e não possuir maior embasamento teórico, fundamentando-se apenas nas recomendações de especialistas, como Humphrey. Para validar suas prescrições seria desejável que ele fosse comparado em condições controladas com outros modelos, como por exemplo, o Bootstrap [apud 37], ou o de Weinberg [38], [39].

Existe ainda um paradoxo envolvendo o CMM: grande parte das companhias líderes na produção de software no mundo tais como a Microsoft, Nokia, Motorola, Ericson, etc. não adotaram o modelo, ou não conseguiram alcançar os níveis mais altos de maturidade [40]. Segundo seus críticos, isto ocorre devido aos problemas mencionados acima, principalmente com os relacionados com o fato da estrutura proposta dificultar a introdução de processos novos e criativos. Fazendo-se uma análise crítica do principal documento sobre o modelo [34], verifica-se que atividades empresariais básicas, como o controle de custos e das perdas decorrentes da falta de qualidade ou do retrabalho [3], não são requisitos exigidos por nenhum dos níveis de maturidade. O impacto sobre a rentabilidade dos negócios provavelmente explica o paradoxo mencionado [40].

4.3.1. Integração dos modelos de maturidade da capacitação

O fato de o CMM ter se difundido com sucesso entre os vários setores da indústria de software, notadamente entre os fornecedores do Departamento de Defesa dos EUA, incentivou o SEI a estender o modelo, e seus métodos de avaliação, para outros domínios de aplicação. A mais óbvia delas foi a área de engenharia de sistemas, pois, em última análise, o software é apenas um dos componentes necessários ao correto funcionamento de um sistema computadorizado.

Para atender essa necessidade, o SEI desenvolveu um conjunto de modelos de maturidade para melhoria de processos baseados no CMM. Esses modelos, embora utilizem uma estrutura bastante voltada para o desenvolvimento de software, contemplam o desenvolvimento multidisciplinar, envolvendo outras áreas. Esse conjunto recebeu a denominação de CMMI e apresenta uma visão integrada dos modelos de maturidade da capacitação.

Até o momento, existem quatro áreas que foram incorporadas pelo CMMI [41]: Engenharia de Sistemas (*Systems Engineering-SE*) [42] [43], Engenharia de Software (*Software Engineering-SW*), Desenvolvimento Integrado de Produtos e Processos (*Integrated Product and Process Development-IPPD*) e Desenvolvimento de Fornecedores **(***Supplier Sourcing-SS*).

O CMMI-SW foi desenvolvido com o propósito de auxiliar as organizações no estabelecimento de objetivos e prioridades para melhoria dos processos utilizados no desenvolvimento de software, de forma a aperfeiçoá-los e prover meios de assegurar que eles sejam estáveis, capazes e amadurecidos [42]. Para isso foram desenvolvidas duas representações do modelo: contínua e por estágios.

A representação contínua destina-se preferencialmente às organizações com maior experiência na área e que pretendam selecionar uma seqüência comprovada de melhorias nos seus processos, que se mostre mais adequada aos seus objetivos de negócios e minimize os riscos operacionais. Também deve ser utilizada pelas organizações que pretendem realizar comparações com os resultados obtidos por outras instituições. Direciona-se ainda para aquelas que pretendam migrar do padrão EIA/IS 731 para o CMMI-SW, ou comparar os resultados obtidos com os da Norma ISO/IEC 15504.

A representação por estágios destina-se a preferencialmente às organizações que estejam

iniciando as atividades de aperfeiçoamento dos processos e fornece uma seqüência comprovada para melhorias, que se iniciam com práticas básicas de gerência e progridem através de um roteiro predefinido composto de níveis sucessivos, onde cada um deles serve de base para o seguinte. Esta representação também permite comparações cruzadas entre organizações através de seus níveis de maturidade. Fornece ainda uma migração fácil do SW-CMM para o CMMI-SW e permite comparações entre organizações através de um critério único que sumariza os resultados obtidos nas avaliações.

Ambas as representações foram projetadas para fornecer resultados equivalentes, independentemente do fato de serem utilizadas para a melhoria dos processos, ou para avaliações quanto a sua maturidade e capacitação. Seus componentes são comuns e divididos entre: áreas de processo, objetivos específicos, práticas específicas, objetivos genéricos, práticas genéricas, produtos típicos do trabalho, subpráticas, extensões das disciplinas associadas às práticas específicas, detalhamento da aplicação das práticas genéricas, e referências relacionadas com as áreas de processo.

As áreas de processo são grupos de práticas relacionadas que, se executadas em conjunto, satisfazem uma série de objetivos considerados importantes para obter melhorias significativas nessa área. Todas as áreas de processo são comuns às duas representações, embora na representação contínua elas sejam organizadas por processo, enquanto que na por estágio, são organizadas por nível de maturidade.

Com o objetivo de apoiar e direcionar as melhorias, a representação por estágios organiza as áreas dos processos em cinco níveis de maturidade e indica quais delas devem ser satisfeitas para atingir cada nível. Esses níveis representam a posição em que os processos da organização se encontram no caminho da evolução. Essa representação utiliza ainda quatro conjuntos de práticas comuns para organizar as práticas genéricas. A Figura 3 apresenta esquematicamente os principais componentes nessa representação.

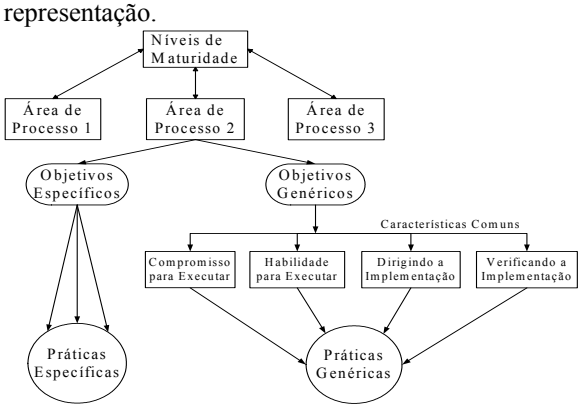

Fig.3–CMMI-SW-Componentes do modelo na representação por estágios

A representação contínua emprega seis níveis de capacitação para organizar os componentes do modelo. Ela agrupa as áreas do processo em categorias que apresentam afinidade e estabelece os níveis de capacitação necessários para alcançar as melhorias dentro de cada uma delas. Esses níveis representam o estágio em que as organizações se situam na busca por melhorias nas várias áreas do processo, fornecendo também uma seqüência recomendada para abordá-las. Essa representação permite alguma flexibilidade ao estabelecer a ordem em que as áreas do processo devem ser abordadas. Ela utiliza objetivos genéricos e específicos para organizar as práticas que devem ser utilizadas. Na Figura 4 apresenta-se esquematicamente como são organizados os principais componentes dessa representação.

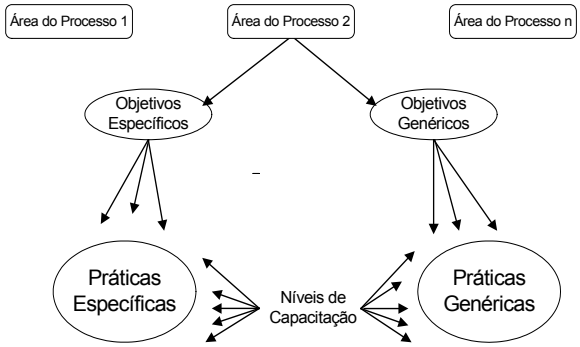

Fig.4–CMMI-SW-Componentes do modelo na representação contínua

Finalmente, pode-se constatar que a representação contínua utiliza níveis de capacitação para avaliar as melhorias nos processos, enquanto que a representação por estágios utiliza níveis de maturidade [42]. A principal diferença entre esses conceitos está relacionada com a maneira como eles são aplicados em cada representação.

O CMMI-SW recebe da comunidade de desenvolvedores de software as mesmas críticas que antes eram destinadas ao CMM. Como o modelo é bem mais completo e muito mais detalhado que o seu antecessor, elas enfatizam os pontos relacionados com as dificuldades e custos envolvidos na sua implementação. Essas críticas são pertinentes em muitos casos, notadamente naqueles que envolvem organizações de pequeno porte e projetos de extensão reduzida. Embora esta adaptação tenha sido mencionada no modelo [42], até hoje ele permanece mais adequado para a realidade dos grandes projetos do Departamento de Defesa dos EUA. Esse fato tem feito com que o SEI venha incentivando o desenvolvimento de estudos que visem adaptar o modelo a esses casos.

4.3.2. Normas ISO 15504

O conjunto de Normas ISO/IEC 15504 [44] representam a evolução do projeto SPICE (*Software Process Improvement and Capability dEtermination*), desenvolvido pela ISO em conjunto com a comunidade internacional. No Brasil esse grupo foi representado através do Grupo de Estudos da ABNT CB-21/SC-10: Subcomitê de Software CE-21:1001.4: Avaliação dos Processos de Software.

As Normas ISO 15504 destinam-se à realização de avaliações dos processos de desenvolvimento de software, com o objetivo de: proporcionar o seu aperfeiçoamento através da identificação das deficiências existentes e conseqüente proposição de melhorias que visem eliminá-las; e de possibilitar uma análise padronizada sobre a capacidade das organizações produtoras e de potenciais fornecedores.

O modelo definido pela Norma apresenta duas dimensões: a primeira, que define os processos a serem avaliados, e a segunda, onde é estabelecido um modelo de medição que permite determinar a capacidade de um processo para atingir os seus objetivos e gerar resultados satisfatórios quanto à qualidade.

A dimensão de processo é fornecida por um Modelo de Referência de Processos externo, onde cada processo é descrito pelos seus objetivos e características que detalham como ocorre sua implementação, incluindo práticas base e possíveis produtos de entrada e saída. Ela envolve cinco categorias, relacionadas com o tipo de atividade que endereçam: Cliente-Fornecedor, Engenharia, Suporte, Gerenciamento e Organização.

A segunda dimensão de capacidade dos processos consiste numa estrutura (framework) de medição que contém seis níveis de capacitação (incompleto, executado, gerenciado, estabelecido, previsível e otimizado) e os respectivos atributos que podem ser associados para a caracterização da capacidade em cada um deles [44]. Cada atributo define um aspecto particular da capacidade do processo e durante a avaliação para definição do nível de capacidade existente verifica-se qual o grau de atendimento de cada um deles.

O Processo de Avaliação, segundo a Norma, deve conter pelo menos as seguintes atividades: Planejamento da avaliação a ser desenvolvida e documentada; Coleta de Dados requeridos para avaliar os processos dentro do escopo a ser analisado; Validação dos Dados com relação à objetividade, representatividade e consistência; Avaliação dos Atributos do Processo baseada na validade dos dados de cada atributo do processo e Relatório com os resultados obtidos.

A Figura 5 apresentada a seguir, mostra de maneira esquemática o relacionamento entre os principais elementos da Norma:

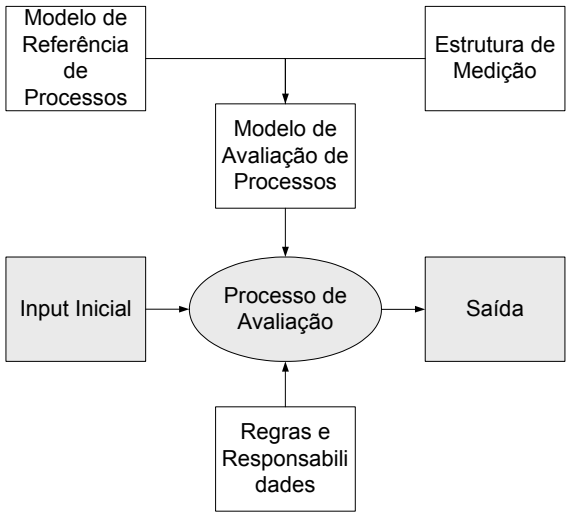

Fig. 5–Norma ISO 15504- Principais componentes do modelo

Assim como no CMMI, o processo de avaliação proposto pela ISO 15504 também é complexo e dispendioso para ser executado. Por ser um padrão internacional é provável que os custos de sua operacionalização tornem-se menores que os do CMMI. No Brasil a sua utilização ainda não encontra grande acolhida, embora existam iniciativas que objetivam adaptá-lo à realidade das micro e pequenas empresas de desenvolvimento de software do país [45].

Apesar do projeto SPICE ter sido desenvolvido durante muitos anos, as Normas ISO 15504 só foram publicadas oficialmente em 2004 e sua utilização ainda não se difundiu como o CMM/CMMI. Com o passar dos anos elas devem obter aceitação crescente, tendo em vista tratar-se de um padrão internacional e desenvolvido com a participação da comunidade acadêmica e empresarial de muitos países. Nos EUA e em alguns países como a Índia, a hegemonia do CMMI ainda deve permanecer por muitos anos, tendo em vista os vultosos investimentos feitos pelas empresas para certificação e adaptação ao modelo.

Embora o SEI assegure que todos os produtos desenvolvidos pelo CMMI sejam consistentes e compatíveis com as Normas ISO 15504, verifica-se algumas diferenças de abordagem entre os dois modelos, embora na sua essência eles sejam bastante semelhantes. Mesmo assim, existem grandes desafios para harmonizar as avaliações, tendo em vista a dificuldade prática para estabelecer critérios uniformes e consistentes nos dois modelos.

Este fato envolve grandes disputas tecnológicas e mercadológicas, não apontando para uma solução a curto prazo. Com a difusão da sua utilização da ISO 15504 as empresas de desenvolvimento de software que atuam no mercado internacional, correm o risco de precisar se certificar segundo os dois modelos

para continuar atuando. Os custos envolvidos são bastante elevados e os benefícios muito pequenos dada a similaridade dos modelos.

4.3.3. Norma ESA ECSS–Q–80B

A Norma ECSS–Q–80B da European Cooperation for Space Standardization [17] define um conjunto de requisitos de garantia dos produtos de software utilizados no desenvolvimento e manutenção de sistemas espaciais. Esses sistemas incluem naves espaciais tripuladas ou não, veículos lançadores, cargas úteis, experimentos e equipamentos e facilidades em solo. Ela abrange ainda os componentes de software existentes nos firmwares e o desenvolvimento do software que embora utilizado de maneira indireta, possa afetar diretamente a qualidade de um produto ou o serviço fornecido por um sistema espacial.

Segundo a Norma, a garantia da segurança de um produto de software envolve a garantia do processo e a garantia da qualidade dos produtos associados a esse processo [17].

A ECSS-Q-80B está estruturada em três partes: a primeira trata da implementação de um programa de garantia dos produtos de software; a segunda da garantia dos processos e a terceira da garantia da qualidade dos produtos. Em todas elas a Norma especifica quais os resultados esperados e os artefatos envolvidos. Na Figura 5 apresenta-se de maneira resumida como esses componentes se relacionam.

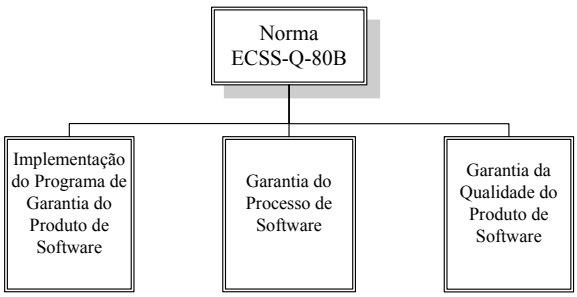

Fig.6- Estrutura da Norma ECSS-Q-80B

A implementação do programa de garantia dos produtos de software aborda as questões relacionadas com: a organização e responsabilidade sobre os processos envolvidos; os aspectos contratuais; o gerenciamento do programa de garantia dos produtos de software; a gerência de riscos e o controle dos itens críticos; a seleção e controle de fornecedores; as atividades de aquisição; as ferramentas e o ambiente de apoio ao desenvolvimento; e a avaliação e melhoria dos processos.

O desenvolvimento de garantia dos processos de software utilizados envolve, segundo a Norma, os aspectos que estão diretamente relacionados com o ciclo de vida que será utilizado no desenvolvimento; os requisitos aplicáveis a todos processos de engenharia de software; e os requisitos aplicáveis a processos ou atividades específicas de engenharia de software.

A garantia da qualidade dos produtos de software empregados na área espacial encerra os objetivos de qualidade do produto e sua medição; os requisitos de qualidade do produto; a documentação de apoio; a padronização do hardware utilizado pelo sistema final; e o firmware eventualmente utilizado pelo sistema.

O modelo proposto pela ECSS-Q-80B é bastante sucinto e relativamente simples de aplicar, tendo em vista que todas as saídas esperadas são especificadas pela Norma. No entanto, ele não propicia um método de avaliação que permita comparar os estágios de evolução em que se encontram os processos utilizados por diferentes organizações.

Apesar de focalizar sua abordagem no estabelecimento e utilização de um conjunto de métodos, técnicas e ferramentas, que representem as melhores práticas de engenharia de software, sua abordagem representa uma evolução em relação às demais apresentadas neste trabalho. Isto se deve ao fato da ECSS-Q-80B empregar um enfoque mais abrangente que envolve a qualidade do produto, a qualidade do seu processo de produção, e ainda, a implantação de um programa de garantia da qualidade de ambos. Este enfoque provavelmente se deve ao fato dela ser direcionada para o emprego no desenvolvimento de sistemas críticos.

4.4. Abordagem centrada no gerenciamento da produção

Ao analisar a evolução das sistemáticas utilizadas para garantia da qualidade de software ao longo dos anos, verifica-se que houve uma tendência das abordagens migrarem de uma orientação voltada para as características do produto, para outra, mais direcionada aos seus processos de produção. Observa-se, no entanto, que elas carecem de um enfoque direcionado ao emprego de melhores técnicas de gerenciamento [46]. Muitos problemas relacionados com a qualidade que podem ser observados no dia-a-dia das organizações produtoras, têm sua origem nas dificuldades gerenciais relacionadas com a estrutura organizacional, as pessoas, as tarefas, a tecnologia utilizada e ambiente empresarial [47].

A prática tem demonstrado que a aplicação direta das abordagens para melhoria da qualidade, usadas em processos de manufatura, não funcionam satisfatoriamente na área de desenvolvimento de software, pois sua produção é influenciada por um espectro muito grande de fatores específicos relacionados com os processos, produtos e recursos utilizados [46].

Assim, as atividades administrativas relacionadas

com as tarefas de planejar, organizar, dirigir e controlar o sistema produtivo visando a melhoria da qualidade precisam ser devidamente adaptadas às características específicas dos produtos de software, e do seu processo de produção [48]. Por se tratar de um setor relativamente novo, ainda não existem muitas técnicas e processos administrativos consagrados nessa área. Mesmo assim, pode-se constatar que certas práticas desenvolvidas em outros setores da indústria para o gerenciamento de projetos têm sido adaptadas com sucesso [49], embora se mostrem insuficientes para atender as demandas requeridas pelos futuros ambientes de desenvolvimento [50].

Ao analisar a produção de software verifica-se que ela apresenta características típicas do modelo de produção "por encomenda", evidenciado pela realização de empreendimentos temporários voltados à produção de produtos ou serviços únicos, cuja realização envolve certo grau de incerteza [49]. Atualmente as melhores práticas para o gerenciamento deste tipo de atividade são expressas palas sistemáticas propostas pelo PMBOK-Project Management Body of Knowledge [51], elaborado pelo PMI-Project Management Institute.

Os objetivos do PMBOK podem ser sintetizados através de um conjunto de conhecimentos, técnicas e práticas de engenharia que visam a utilização de melhores:

- Métodos, técnicas, ferramentas, serviços, sistemas de apoio e materiais utilizados na produção;
- Métodos e técnicas para definição, organização e padronização dos trabalhos e procedimentos;
- Técnicas para planejamento qualitativo e quantitativo da produção;
- Sistemas para mensuração e comparação dos resultados obtidos com os padrões de desempenho estimados;
- Práticas para eliminar ou minimizar os desvios e efeitos adversos sobre a produção que tenham ocorrido; e
- Métodos para aperfeiçoamento contínuo do trabalho e do sistema de produção.

Para facilitar sua aplicação, o PMBOK divide esse conjunto em cinco fases: inicialização, planejamento, execução, controle e monitoração, encerramento [51].

Na fase de inicialização é definido o esboço do escopo do projeto que deverá ser entregue ao final e como será estruturada a sua organização básica. Na etapa de planejamento o escopo do produto e do projeto são detalhados, define-se a equipe responsável pela execução, as estimativas de prazo e custo, os riscos envolvidos, as ações corretivas e a forma de comunicação entre a equipe. Na fase de execução os trabalhos são efetuados, seguindo as diretrizes estabelecidas no planejamento, acrescidas das mudanças e correções que tenham sido relatadas. No estágio de monitoração e controle é feito o acompanhamento contínuo das atividades, avaliando seu progresso e recomendando os ajustes necessários. Na fase de encerramento o produto é finalizado, instalado no cliente e feita a sua aceitação pelos usuários. A documentação é verificada e atualizada, os resultados obtidos são registrados, e executados os procedimentos administrativos formais para o encerramento do projeto.

Para apoiar, de maneira automatizada, os diversos processos envolvidos na gerência da produção de software, têm sido desenvolvidas novas ferramentas e ambientes integrados para desenvolvimento de software, representadas pelos I-CASE-Integrated Computer-Aided Software Engineering [52], [25], que procuram integrar as características técnicas e gerenciais abrangidas pela produção de software.

O gerenciamento da produção de software tendo em vista a melhoria de sua qualidade é uma tarefa complexa, pois envolve a negociação contínua de demandas concorrentes com relação ao escopo, prazos, custos e riscos envolvidos no projeto. Durante o processo de desenvolvimento frequentemente as partes envolvidas apresentam diferentes necessidades e expectativas com relação aos resultados obtidos. Identificar e compatibilizar todos esses requisitos envolve capacidade para planejar, organizar, dirigir, executar, e controlar que vão muito além dos limites de um guia sobre o gerenciamento de projetos como o PMBOK.

No entanto, a prática tem demonstrado que empresas que praticam o gerenciamento de projetos de maneira consistente e baseada nos seus preceitos têm obtido vantagem competitiva sobre as demais [49], [46].

5. Melhoria nos Processos

Segundo Kellner [53] e Paulk [35], "A qualidade de um produto de software é diretamente determinada pela qualidade dos processos empregados no seu desenvolvimento e manutenção". Huphrey vai além e afirma que "se um processo não está sob controle estatístico, não será possível obter um progresso sustentável até que isto aconteça" [33]. Essas afirmativas realçam os aspectos dos principais enfoques que têm sido propostos para melhoria da qualidade de software nos últimos anos.

Praticamente todos os modelos apresentados anteriormente neste trabalho enfatizam o fato que para melhorar a qualidade de software é necessário aperfeiçoar o seu processo de desenvolvimento. Sem dúvida esta premissa é correta, no entanto, deve-se ter em mente que a produção de software não algo é tão linear, ou feito de maneira tão disciplinada e mecanizada, como uma linha de produção industrial. Algumas atividades envolvidas no seu desenvolvimento, como levantamento de requisitos,

ou projeto de alto-nível, exigem alto grau de criatividade e colaboração entre os membros da equipe, que podem ser desestimuladas caso seja implantada um estrutura muito rígida, como a proposta em alguns modelos [40].

Desta maneira, a postura mais coerente aponta para a formalização dos processos utilizados no desenvolvimento, mantendo-se um equilíbrio constante entre disciplina e criatividade, como proposto por Boehm [54].

Para melhorar a qualidade de software têm sido propostas diversas estratégias que se baseiam no modelo PDCA proposto por Deming [55] e envolvem: Planejar, Executar, Controlar e Agir no sentido de corrigir continuamente os problemas encontrados.

Para implementá-lo têm sido utilizadas diversas abordagens, dentre as quais, destaca-se pela simplicidade, a proposta por Falconi [56], que consiste em estabelecer inicialmente um ciclo de trabalho claro e definido, de forma que seja possível obter maior previsibilidade dos resultados obtidos. Para isto estabelecem-se padrões de trabalho em cada etapa e descreve-se como verificar a existência de problemas ou resultados não desejados. Na etapa seguinte desse ciclo, atua-se no sentido de corrigir os desvios encontrados e suas causas, de tal modo que os resultados desejados possam ser seguidamente alcançados.

Segundo Shingo [57], os processos podem ser melhorados de duas maneiras: a primeira consiste em melhorar o produto em si, através da análise de valor de cada uma das etapas envolvidas, visando a eliminação de tarefas que consumam recursos de produção, mas não agreguem valor significativo ao produto. A segunda consiste em melhorar os métodos de produção sob o ponto de vista da engenharia e tecnologia de fabricação. Em ambos os casos pressupõe-se que os processos sejam estáveis e bem definidos.

No desenvolvimento de software a padronização dos processos de produção usualmente é representada pela modelagem dos processos, que é desenvolvida com os seguintes objetivos: propiciar melhor entendimento do processo, projetar novos processos com base em experiências anteriores, fornecer treinamento para a equipe que irá executá-los, propor melhorias na sua execução, e fornecer suporte sistemático para as pessoas envolvidas na produção [53] [33].

Ao longo do tempo diversas técnicas foram desenvolvidas para sua realização. Elas evoluíram dos diagramas semelhantes aos DFD's utilizados por Humphrey [33] até as notações gráficas e linguagens de modelagem de processos e fluxos de trabalho dos dias atuais.

Nos últimos anos muitas abordagens têm sido desenvolvidas com esse objetivo. Este fato criou certa dificuldade para a seleção do conjunto mais adequado para cada empresa ou projeto. Existem desde enfoques formais, que empregam uma sintaxe e semântica formal para descrever os processos, até outras bem menos rígidas, que empregam apenas uma sintaxe formal aplicada a uma notação gráfica.

A seguir, apresentam-se três abordagens distintas para a modelagem de processos. Elas foram selecionadas, dentre um grande número de alternativas disponíveis, devido aos seguintes fatores: empregam padrões abertos e de ampla aceitação, dispõem de ambientes automatizados para sua utilização, possuem ampla base de aplicação, e finalmente, por apresentam grande potencial para pesquisas e desenvolvimento tecnológico do setor.

5.1.1. SPEM

O SPEM-Software Process Engineering Metamodel Specification é um metamodelo desenvolvido pelo consórcio OMG-Object Management Group para descrever um processo de desenvolvimento de software, ou uma família de processos relacionados [58].

O SPEM utiliza a notação da UML-Unified Modeling Language, que usualmente é empregada no projeto de software orientado a objetos, para descrever os processos utilizados no seu desenvolvimento. O metamodelo do SPEM é formalmente definido como uma extensão de um subconjunto da UML denominado SPEM_Foundation [58].

O objetivo do SPEM é definir e documentar os processos de desenvolvimento de software, especialmente aqueles que envolvem ou obrigam a utilização da UML, como por exemplo o RUP-Rational Unified Process. Com isto pretende-se que eles sejam analisados, avaliados e aperfeiçoados por toda a equipe de desenvolvimento.

A notação gráfica do SPEM define uma série de ícones para representar graficamente os elementos envolvidos nos processos desenvolvimento de software tais como Produto de trabalho, Definição de trabalho, Atividade, Executor de processo, Papel, Pacote de trabalho, Fase, Processo, Documento e Modelo UML. Eles são empregados em diagramas UML tais como casos de uso, atividades, pacotes, etc. para descrever e documentar os processos. A Figura 6 mostra o exemplo de um processo modelado através do SPEM.

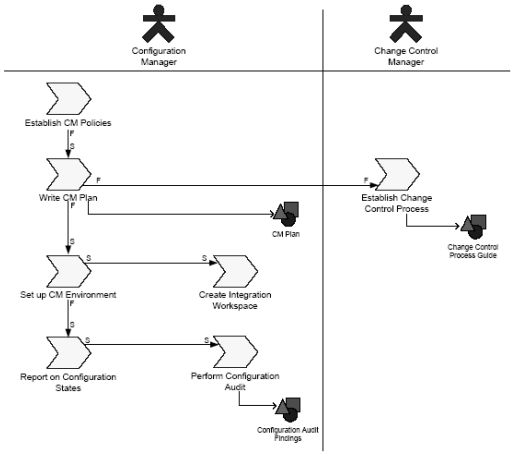

Fig. 7 – Exemplo de um processo modelado com SPEM

Fonte: SPEM: Extending the UML into the Process Engineering Domain [59]

A utilização pelo SPEM de uma linguagem baseada na UML é ao mesmo tempo sua maior qualidade e sua maior limitação. Qualidade pois utiliza para descrição dos processos uma linguagem e padrão gráfico amplamente conhecido pelos desenvolvedores. Limitação porque a UML foi criada para especificar, projetar e detalhar as características de um software, e não do seu processo de produção. Apesar das extensões da UML que o SPEM implementa, elas não se mostram suficientes para descrever adequadamente o comportamento dinâmico dos processos envolvidos.

Dentro do OMG existem atualmente outras iniciativas voltadas para a modelagem de processos e fluxos de trabalho [58]. Em uma delas, relacionada com a definição de processos, o OMG já iniciou uma série de consultas para adaptação do padrão denominado Business Process Modeling Notation (BPMN), desenvolvido pela Business Process Management Initiative – BPMI [60], que a princípio parece ser mais adequado para modelagem de certos tipos de processos de desenvolvimento de software.

5.1.2. Business Process Modeling Notation

A Business Process Modeling Language [61] e respectiva notação gráfica, denominada Business Process Modeling Notation-BPMN [62] são iniciativas da Business Process Management Initiative, uma organização não lucrativa que tem por objetivo promover e desenvolver um conjunto de padrões abertos e livres do pagamento de royalties, que facilitem o gerenciamento dos processos de negócio. Os padrões propostos são baseados na XML visando o intercâmbio de dados com outros sistemas de informação e ferramentas para modelagem de processos.

A especificação da BPML fornece um modelo abstrato para expressar os diversos aspectos dos processos de negócio e suas entidades de apoio. Ela descreve as atividades envolvidas e o contexto onde elas ocorrem. Os processos são definidos como atividades complexas e o contexto define o ambiente onde ocorre sua execução, que pode ser instanciada de acordo com condições locais. O padrão possui ainda outros componentes que auxiliam na contextualização dos processos e descrição de seus fluxos e controles ao longo do tempo.

Por ter sido criada para modelar processos de negócios, e não apenas àqueles que envolvem o desenvolvimento de software, a notação é bastante extensa e completa, permitindo modelar fluxos de trabalhos complexos e de grande amplitude.

Seus principais elementos são os objetos de fluxo, capazes de definir o comportamento do processo, representados pelos eventos, atividades e os gateways utilizados para controlar a convergência e divergência dos fluxos dos processos. Existem três maneiras de conectar esses fluxos: através de fluxos de seqüência ou de mensagens e ainda através de associações entre elas. Os modelos podem ser agrupados em raias (swinlanes) ou em associações (pools). A notação traz ainda uma série de artefatos, utilizados para fornecer informações adicionais sobre os modelos, que incluem objetos de dados, grupos de atividades e anotações.

Cada um desses elementos é composto por vários componentes gráficos que permitem detalhar minuciosamente os processos, inclusive com relação ao controle dos fluxos normais e de exceção existentes nos processos.

A figura 7 mostra o exemplo de um processo modelado com a BPMN:

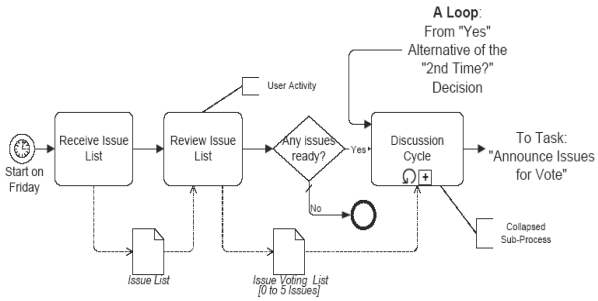

Fig. 8 – Exemplo de um processo modelado com BPMN

Fonte: Business Process Modeling Notation-BPMN-v. 1.0 [62]

A BPMN pretende padronizar uma notação gráfica, e respectiva semântica, para modelagem dos Diagramas de Processos de Negócio, provendo um meio simples de comunicar aos outros usuários uma grande variedade de informações sobre os processos executados numa organização. Provê também o mapeamento formal para uma linguagem de execução associada (BPML4WS-Business Process Execution Language for Web Services), que se baseia na XML para oferecer portabilidade entre os

modelos.

A BPMN representa atualmente uma das iniciativas mais promissoras para estabelecer um padrão "de fato" para a modelagem de processos, inclusive daqueles envolvidos na produção de software. Um grande passo nesse caminho já foi dado com a sua integração aos trabalhos do OMG.

No entanto, assim como o SPEM, a BPMN só consegue descrever os aspectos de um processo quando analisados estaticamente, o que não corresponde à realidade dinâmica da vida empresarial. Para contornar essa limitação têm surgido iniciativas que visam integrar a modelagem estática com os aspectos dinâmicos analisados pelos modelos de simulação de processos.

5.1.3. Simulação de processos

A utilização da simulação de sistemas discretos tem sido utilizada para modelar uma ampla gama de aplicações, visando otimizar os processos envolvidos. Ela se mostra bastante adequada para modelar e analisar situações em que o processo pode ser representado por um conjunto atividades, diante das quais se formam uma série de filas, que competem pelos recursos disponíveis [63].

Por esta razão ela frequentemente é utilizada na análise de desempenho dos sistemas de produção, com o objetivo de identificar eventuais gargalos que possam ocorrer, e no projeto de estratégias diferenciadas para sua operação em caso de situações anômalas ou de crise. Nesses casos elas se mostram muito superiores às outras representações, pois permitem simular previamente a ocorrência de crises, possibilitando a tomada de medidas preventivas. Se alimentadas em tempo real durante as crises, permitem determinar os processos que irão saturar primeiro, permitindo a tomada de medidas que minimizem o impacto indesejado.

As metodologias e ferramentas utilizadas para a modelagem estática dos fluxos de trabalho apresentam semelhanças muito grandes com aquelas empregadas na simulação de processos. Embora não permitam analisar seus aspectos dinâmicos, elas possibilitam um melhor conhecimento do relacionamento entre os processos, permitindo sua rápida reestruturação para acomodar novas necessidades do negócio, como um novo projeto, uma nova filial, ou mesmo um novo processo.

A partir dessa constatação, verificou-se a oportunidade de fundir as duas abordagens de maneira a permitir uma análise dos aspectos estáticos e dinâmicos dos sistemas de produção. Assim, através do desenvolvimento de uma abordagem integrada seria possível acrescentar facilidades típicas de uma área e que não estejam presentes na outra. Outra possibilidade seria a construção de ferramentas computacionais que propiciassem um ambiente computacional integrado para simulação e gestão de processos.

Como a gestão da qualidade de software prevê a utilização de processos estáveis e bem definidos, a utilização conjunta desses conceitos permitiria a simulação dos problemas, antes e durante as crises eventualmente ocorridas na produção, permitindo calcular os riscos envolvidos e tomar medidas preventivas antecipadamente.

O Núcleo de Estudos em Modelagem de Sistemas - Nemesis, sediado no Laboratório Associado de Computação e Matemática Aplicada do INPE, têm desenvolvido estudos sobre a modelagem e simulação de sistemas discretos, de caráter conceitual e de aplicação, visando a sua aplicação na melhoria dos processos de desenvolvimento de software. A perspectiva é de, em trabalhos complementares, prosseguir no detalhamento desta proposta rumo a implementação de ações específicas.

6. Conclusões

Este trabalho apresenta uma visão geral sobre a melhoria da qualidade de software. Foram apresentadas as principais questões envolvidas na sua melhoria e proposta a sua abordagem segundo três pontos de vista complementares: a engenharia do produto, a engenharia da produção, e o gerenciamento da produção.

Foram apresentados e analisados criticamente os principais modelos e normas técnicas utilizados para gerência da qualidade de software no setor aeroespacial brasileiro, representados pelo CMM e CMMII do SEI, pelas Normas ISO 15504 e pela Norma ECSS–Q–80B da European Cooperation for Space Standardization.

Também foi realizada uma análise sobre o gerenciamento da produção de software e apresentadas as maiores dificuldades administrativas encontradas para planejar, organizar, dirigir e controlar seu sistema de produção, visando à melhoria da qualidade dos produtos desenvolvidos.

Foram também descritas as características de algumas metodologias utilizadas para melhoria dos processos de produção de software e apresentada uma proposta para sua evolução através da simulação de processos. Mostrou-se também que a evolução destas técnicas aponta para uma convergência entre os métodos utilizados para descrever processos, fluxos de trabalho e simulação.

Praticamente todos os modelos apresentados neste trabalho enfatizam o fato que para melhorar a qualidade de software é necessário aperfeiçoar o seu processo de desenvolvimento. Sem dúvida esta premissa é correta, no entanto, deve-se ter em mente que a produção de software não algo é tão linear, ou feito de maneira tão disciplinada e mecanizada, como uma linha de produção industrial. Algumas

atividades envolvidas no seu desenvolvimento, exigem criatividade e colaboração entre os membros da equipe, que podem ser desestimuladas caso seja implantada um estrutura muito rígida, como a proposta em alguns modelos citados.

Finalmente, pode-se concluir que o gerenciamento da qualidade de software é uma tarefa complexa e multidisciplinar. Envolve a negociação contínua de demandas concorrentes com relação ao escopo, prazos, custos e riscos envolvidos no projeto. Durante o processo de desenvolvimento frequentemente as partes envolvidas apresentam diferentes necessidades e expectativas com relação aos resultados obtidos.

Como a atual tecnologia de produção de software ainda é uma atividade intensiva em mão-de-obra, as habilidades gerenciais exercem grande influência nos resultados obtidos com relação à qualidade dos produtos desenvolvidos. Este aspecto tem sido relativamente menosprezado nos estudos da área, que têm concentrado suas abordagens nos aspectos tecnológicos da questão.

Conciliar todos esses fatores é uma tarefa que vai muito além dos limites tecnológicos já alcançados. Verifica-se, portanto, que apesar de todos os progressos obtidos ao longo dos últimos anos, as atividades de garantia da qualidade de software ainda encontram grande espaço para evoluir tendo em vista as necessidades crescentes da sociedade moderna. Nesse contexto, o uso integrado da modelagem e simulação poderá trazer bons resultados.

Bibliografia

[1] Moura, R.M.; Albertin, A.A. *Benefícios da Tecnologia da Informação no Desempenho Empresarial*. In: Moura, R.M.; Albertin, A.A. (Organizadores) *Tecnologia da Informação*. São Paulo: Atlas, 2004.

[2] Collier, B.; Demarco, T.; Fearey, P. *A Defined Process for Project Postmortem Postmortem Reviews*. IEEE Software, v.13 n.4, p.65-72. 1996.

[3] Augusto Neto, Á.. *Uma estratégia para gerência da qualidade e produtividade no desenvolvimento de software*. Dissertação (Mestrado em Informática) - Instituto Tecnológico de Aeronáutica, São José dos Campos, 1997

[4] Augusto Neto, Á.; Sant'Anna, N. *Uma Estratégia para Gerência do Processo Baseada nos Custos da Qualidade de Software*. In: Simpósio Brasileiro de Qualidade de Software, 2, Fortaleza. Anais...Fortaleza: UNIFOR, 2003, p. 32-46.

[5] Naur, P.; Randell, B. *Report on a conference sponsored by the NATO Science Committee*. Garmisch, Germany, 1968. Disponível em: <http://en.wikipedia.org/wiki/List_of_publications_in_computer_science#Software_engineering:_Report_of_a_conference_sponsored_by_the_NATO_Science_Committee>. Acesso em 1/8/2005.

[6] Pressman, R.S. *Engenharia de Software*. 5.ed. Rio de Janeiro:McGraw-Hill, 2002

[7] Crosby, P.B. *Qualidade é investimento*. Rio de Janeiro: José Olympio, 1992.

[8] Garvin, D. A. *Gerenciando a qualidade: a visão estratégica e competitiva*. Rio de Janeiro: Qualitymark, 1992.

[9] Sommerville, I. *Engenharia de Software*. São Paulo: Addison Wesley, 2003.

[10] Abran, A.; Moore, J.W. *Guide to the Software Engineering Body of Knowledge (SWEBOK Guide)*. Los Alamitos: IEEE Computer Society, 2004.

[11] Crosby, P. B. *Qualidade sem lágrimas: a arte da gerência descomplicada*. Rio de Janeiro: José Olympio, 1992

[12] Associação Brasileira de Normas Técnicas. *NBR ISO 9000 Sistemas de Gestão da Qualidade - Fundamentos e Vocabulário.* Rio de Janeiro: ABNT, 2000.

[13] Associação Brasileira de Normas Técnicas. *Tecnologia da Informação – Avaliação de produto de software – Características de qualidade e diretrizes para o seu uso*. Rio de Janeiro: ABNT, 1996.

[14] Boehm, B.W.; Papaccio, P.N. *Understanding and controlling software costs*. IEEE Transactions on Software Engineering, vol. 14, nº 10, p. 1462-1477, 1988.

[15] Moura, C.A.T; Santellano, J.; Augusto Neto, Á. *Software e segurança de sistemas aeroespaciais*. In: II Encontro de Iniciação Científica e Pós-Graduação do ITA, 1996, São José dos Campos. Anais.São José dos Campos: 1996. p.166-170.

[16] European Cooperation for Space Standardization. *ECSS-E-40B – Space Engineering – Software*. Noordwijk: 1999.

[17} European Cooperation for Space Standardization. *ECSS-Q-80B – Space Product Assurance - Software Product Assurance*. Noordwijk: 2003.

[18] Heil, J.H. *Practical Applications of Software Quality Assurance to Mission-Critical Software.* In: Schulmeyer, G.G.; McManus, J.I. (Editors) *Handbook of Software Quality Assurance*. 3rd ed. Upper Saddle River: Prentice Hall, 1998.

[19] Hower, R. *What are some recent major computer system failures caused by software bugs?* Disponível em <http://www.softwareqatest.com/qatfaq1.html#FAQ1_3> Acesso em 18/09/2005.

[20] Levenson, N. *The Role of Software in Spacecraft Accidents*. AIAA Journal of Spacecraft and Rockets, Vol. 41, No. 4, July 2004. Disponível em <http://sunnyday.mit.edu/papers/jsr.pdf>. Acesso em 14/07/2005.

[21]Martins, E. *Software Testing: an Overview*. In: Palestra apresentada durante o Workshop do Projeto Plavis. INPE – S.José dos Campos – 22-24/03/2005.

[22] Kitchenham, B.A.; Pfleeger, S.H. *Software Quality: the elusive target.* IEEE Software, v.13, n.1, p.12-21, 1996.

[23] Pfleeger, S.L. *Engenharia de Software: teoria e prática*. São Paulo: Prentice Hall, 2004.

[24]Fairley, R. E. *Software engineering concepts*. New York: McGraw-Hill, 1985.

[25] Sant´Anna,N. *Um ambiente Integrado para o apoio ao desenvolvimento e gestão de projetos de software para sistemas de controle de satélites*. Tese de Doutorado. Instituto Nacional de Pesquisas Espaciais. São José dos Campos: INPE, 2000.

[26]Dunn, R.H. *Software Quality: Concepts and Plans*. Englewood Cliffs: Prentice Hall, 1990.

[27] Gillies, A.C. *Software Quality: Theory and Management*. London: Chapman & Hall, 1992.

[28] Boehm, B.W.; Brown, J.R.; Lipow, M. *Quantitative evaluation of software quality*. In: Basili, Victor R. *Tutorial on models and metrics for software management and engineering*. Catalog Nº EHO-167-7, IEEE Press, 1980.
[29] McCall, J.A. *An introduction to software quality metrics*. In: Cooper, J.; Fisher, M.J. *Software quality management*. Princeton: Petrocelli Books, 1979.
[30] Voas, J. *Software Quality´s Eight Greatest Myths*. IEEE Software, v.16, n.5, p.118-120, 1999.
[31] Voas, J. *Software´s Secret Sauce: The –ilities*. IEEE Software, v.21, n.6, p.14-15, 2004.
[32] Humphrey. W.S. *Characterizing the Software Process: A Maturity Framework*. IEEE Software, v.5, n.2, p.73-79, 1988.
[33] Humphrey. W.S. *Managing the Software Process*. Reading: Addison-Wesley, 1990.
[34] Paulk, M.C. et al. *Capability maturity model for software, version 1.1. (CMU/SEI-93-TR-024)*. Pittsburgh: Software Engineering Institute, Carnegie Mellon University, 1993.
[35] Paulk, M. C. et al. *The Capability Maturity Model: Guidelines for Improving the Software Process*. Reading: Addison-Wesley Longman, 1995.
[36] van Solingen, H. *Measuring the ROI of Software Process Improvement*. IEEE Software, v.21, n.3, p.32-38, 2004.
[37] Zahran, S. *Software Process Improvement: Practical Guidelines for Business Success*. Harlow: Addison Wesley, 1998.
[38] Weinberg, G.M. *Quality Software Management – Systems Thinking v.1*. New York: Dorset House, 1991.
[39] Weinberg, G.M. *Quality Software Management – First-order Measurement v.2*. New York: Dorset House, 1993.
[40] Conradi, R. Fuggetta, A. *Improving Software Process Improvement*. IEEE Software. V.19, n.4, p.92-99, 2002.
[41] CMMI Product Team. *Capability Maturity Model Integration-Version 1.1-Continuous Representation. CMU/SEI-2002-TR-011*. Pittsburgh: Software Engineering Institute/Carnegie Mellon University, 2002.
[42] CMMI Product Team. *Capability Maturity Model Integration-Version 1.1-CMMI for Software Engineering Staged Representation. CMU/SEI-2002-TR-029*. Pittsburgh: Software Engineering Institute/Carnegie Mellon University, 2002.
[43] CMMI Product Team. *Capability Maturity Model Integration-Version 1.1-CMMI for Software Engineering Continuous Representation. CMU/SEI-2002-TR-028*. Pittsburgh: Software Engineering Institute/Carnegie Mellon University, 2002.
[44] International Organization for Standardization, International Eletrotechnical Commission. *ISO/IEC Std. 15504: Information Technology-Software Process Assessment – part 1 to 5*. Genève: 2003-2005.
[45] Wangenhein, C.G., Anacleto, A., Salviano, C.F. *Mares-A Methodology for Process Assessment in Small Software Companies*. Relatório Técnico LQPS001.04E do Laboratório de Qualidade e Produtividade de Software da Universidade do Vale do Itajaí. São José: UNIVALI, 2004.
[46] Fernandes, A.A., Teixeira, D.S. *Fábrica de Software: implantação e gestão de operações*. São Paulo: Atlas,2004.
[47] Chiavenato, I. *Administração-Teoria, Processo e Prática*. São Paulo: Makron Books, 2000.
[48] Clements, P.C. et al. *Project Management in a Software Product Line Organization*. IEEE Software, v.22, n5, p.54-63, 2005.
[49] Martins, J.C.C. *Gestão de Projetos de Desenvolvimento de Software (PMI-UML)*. São Paulo: Brasport, 2002
[50] Nidiffer, K.E., Dolan, D. *Evolving Distributed Project Management*. IEEE Software, v.22, n.5, p.63-72, 2005.
[51] PMBOK Guide 2004 Update Project Team. *A Guide to the Project Management Body of Knowledge (PMBOK Guide) – 3rd. ed*. Newtown Square: Project Management Institute, 2004.
[52] Humphrey, W.S. *Software and the factory paradigm*. Software Engineering Journal. p.370-376. IEEE Computer Society, 1991.
[53] Kellner, M.I.; Hansen, G.A. *Software Process Modeling (CMU/SEI-88-TR-9)*. Pittsburgh: Software Engineering Institute, Carnegie Mellon University, 1988.
[54] Boehm, B., Turner, R. *Balancing Agility and Discipline: A Guide for the Perplexed*. Boston: Pearson Education, 2004.
[55] Deming, W.E. *Qualidade: a revolução da administração*. Rio de Janeiro: Marques Saraiva, 1990.
[56] Campos, V.F. *Qualidade Total: Padronização de Empresas*. Belo Horizonte: Editora de Desenvolvimento Gerencial, 1999.
[57] Shingo, Shigeo. *O Sistema Toyota de Produção do ponto de vista da Engenharia de Produção*. Porto Alegra: Bookman, 2002.
[58] OMG-Object Management Group. *SPEM-Software Process Engineering Metamodel Specification – Version 1.1*. Disponível em : < http://www.omg.org/docs/formal/ 05-01-06.pdf>. Acesso em 14/7/2005.
[59] Suen, V. *SPEM: Extending the UML into the Process Engineering Domain*. Palestra disponível em <http://www.omg.org/news/meetings/workshops/UML%20 2003%20 Manual/01-3_Suen.pdf >. Acesso em 14/7/2005.
[60] Nemiah, J., Lee, P. *BPMI.org and OMG Announce Strategic Merger of Business Process Management Activities*. Disponível em: <http://www.bpmi.org/downloads/BPMI-OMG_ Merger.pdf>. Acesso em 27/9/2005.
[61] Business Process Management Initiative. *Business Process Modeling Language-BPML-Nov 2002*. Disponível em < http://www.bpmi.org/downloads/BPML1.0.zip>. Acesso em 18/07/2005.
[62] Business Process Management Initiative. *Business Process Modeling Notation-BPMN-v. 1.0 - May 2004*. Disponível em <http://www.bpmi.org/downloads/BPMN-V1.0.pdf>. Acesso em 18/7/2005.
[63] Travassos, P. R. N.; Kienbaum, G.S. *Gerenciamento de projetos e simulação de processos: uma abordagem integrada.* Anais do III Worcap - INPE - São José dos Campos: nov. 2003.
[64] Herbsleb, J. et al. *Benefits of CMM-based software process improvement: initial results* (CMU/SEI-94-TR-013). Pittsburgh: Software Engineering Institute, Carnegie Mellon University, 1994.