A Software Development Process Model Integrated to a Performance Measurement System

G. L. Baptista, R. M. Vanalle and J. A. A. Salles

***Abstract*— Software development process has been considered engineering for less than 50 years. During its evolution, many papers were presented proposing software development models that allow the adaptation of Software Engineering basis to the reality of the companies. It is also known that this industry has been suffering to attend cost, deadlines and/or functionalities. Besides, there are more companies requiring tight targets and the use of performance measurement systems becomes important to enable control, evaluation, prevention and understanding about processes, products and services. Focusing these two issues, this study was conducted, having as a goal the creation of a software development process model that integrates performance measurement systems concepts. As a result of this research, the GMQA Model, acronym from Brazilian Portuguese to Management, Measurement, Quality, and Documentation, is presented. This research details how the model was design and evolved based on a set of presentations to companies from the software industry.**

***Keywords*— Software Engineering, Performance Measurement System, Lean, Stage-gates.**

I. INTRODUÇÃO

A CONSTRUÇÃO de *software* vem sendo estudada nas últimas quatro décadas e sabe-se que tal tarefa é crítica. Isso porque a engenharia de *software*, disciplina que se preocupa com todos os aspectos da produção de *software*, vem sofrendo ao longo dos anos dificuldades para cumprir prazos, custos e funcionalidades esperados, o que torna tais fatores desafiadores para a área [1, 2, 3]. Um exemplo disso pode ser visto nos resultados apresentados pelo *Chaos Report*, que apontam sucesso em apenas 32% dos projetos de *software* [4].

Nos últimos anos, inúmeros paradigmas foram apresentados dentro da Engenharia de *Software* com o objetivo de minimizar os problemas até aqui apontados [5]. Nesse contexto, alguns desses paradigmas utilizaram-se de técnicas já concebidas na Engenharia de Produção, como o desenvolvimento *Lean* [6, 7] e a Inspeção Formal [8].

Todos os esforços resultaram em modelos de processo de desenvolvimento de *software* prescritivos, considerados base teórica possível de adaptação para que as empresas de desenvolvimento possam criar os processos para produção do *software* adequados para cada realidade [9].

Também é certo dizer que durante o processo de evolução da produção de *software*, normas foram criadas para auxiliar na resolução do problema inicialmente apresentado. Dentre elas, pode-se citar a própria ISO 9001 [10], responsável pelo sistema de gestão de qualidade da empresa como um todo; a ISO 12207 [11] que define etapas para o processo produtivo; a ISO 9126 [12] que avalia o produto de *software* gerado; e a ISO 15504 [13] que define uma metodologia para avaliação do processo de desenvolvimento de *software*.

Além disso, modelos para definição da capacidade e maturidade de uma empresa de desenvolvimento de *software* foram incentivados por governos para que fosse possível mensurar a qualidade do *software* que estava sendo desenvolvido. São exemplos disso o CMMI-DEV e o MPS.BR, incentivados respectivamente pelo DoD - Departamento de Defesas dos Estados Unidos da América e pelo MCT - Ministério de Ciência e Tecnologia Brasileiro [14, 15].

Apesar de todos estes esforços, existe grande discussão se os modelos são realmente aplicados na prática [16]. A questão é mais discutível ainda quando se observa o uso de tais metodologias em pequenos grupos de desenvolvimento de *software*, fato mundialmente comum [17, 18, 19]. Um estudo divulgado pela Pesquisa de Qualidade no Setor de *Software* Brasileiro [20] confirma esse cenário, já que menos de 30% das empresas pesquisadas tinham processos certificados ISO 9001, CMMI-DEV e/ou MPS.BR.

A mesma pesquisa aponta que somente 39,6% das empresas medem o desempenho do processo de *software* de forma sistemática. Em contrapartida, diversos autores e modelos afirmam que a medição de desempenho em projetos de *software* pode ser considerada uma ferramenta útil para gestores de projetos de *software* [21, 22]. É possível afirmar que os desenvolvedores de *software* estão muito distantes de

G. L. Baptista, Universidade Nove de Julho (UNINOVE), São Paulo, São Paulo, Brasil, gabriel.baptista@uninove.br.

R. M. Vanalle, Universidade Nove de Julho (UNINOVE), São Paulo, São Paulo, Brasil, rvanalle@uninove.br.

J. A. A. Salles, Universidade Nove de Julho (UNINOVE), São Paulo, São Paulo, Brasil, salles@uninove.br.

outros profissionais quanto ao estabelecimento de padrões de medição e objetivos relevantes [23].

Por tais razões, o presente artigo tem por objetivo propor um modelo de processo de desenvolvimento de *software* integrado a um sistema de medição de desempenho, propiciando assim uma nova opção para a indústria de *software* no que tange a aplicação de processos de desenvolvimento de *software*. Desta forma, tal modelo poderia ser adaptado para situações reais, criando assim um processo de desenvolvimento de software preocupado com a questão de medição de desempenho.

II. REVISÃO DA LITERATURA

De forma a embasar teoricamente a pesquisa apresentada nesse trabalho, foi realizada uma varredura na literatura existente na Engenharia de *Software* e na Engenharia de Produção. Buscou-se definir conceitos básicos em ambas as áreas para atingir o objetivo do trabalho. A literatura pesquisada foi utilizada para avaliar do surgimento da Engenharia de *Software* [24, 1, 9, 3], até modelos conhecidos [25, 24, 26, 27, 28, 29, 30, 31], com as melhores práticas aplicadas na área [25, 28]. O modelo também tomou como base os princípios *Lean* para Desenvolvimento de Software [7, 29, 32, 6, 33, 5, 30]. A definição sobre sistema de medição de desempenho para a Engenharia de Produção também foi pesquisada [34, 35, 36, 37], bem como o conceito de medição de desempenho na Engenharia de *Software* [38, 39, 14, 3, 9, 20]. Estudou-se também a questão de normas [10, 11, 13, 12, 40] e modelos nacionais [15, 41, 42] e internacionais [2, 14] que são aplicados na produção de *software* e a perspectiva dos mesmos no que diz respeito à utilização e apoio aos sistemas de medição de desempenho. Por fim, foi detalhado o papel do conceito de *stage-gates* [43, 44, 45], também aplicado no modelo.

III. METODOLOGIA

Um conjunto de etapas que foram conduzidas ao longo do trabalho de modo a validar o modelo proposto. Primeiramente, um levantamento bibliográfico estruturado e sistemático sobre modelos de processo de desenvolvimento de *software* e sistemas de medição de desempenho foi conduzido, com o objetivo de criar embasamento teórico para o modelo a ser construído.

Após a seleção dos artigos utilizados para a pesquisa, uma análise crítica das metodologias de desenvolvimento de *software* e sistemas de medição de desempenho foi realizada. Essa etapa serviu de base para a escolha dos princípios do modelo conceitual construído, associando os conceitos de Processo de Desenvolvimento de *Software* e Sistemas de Medição de Desempenho. Também foram definidos e descritos indicadores de desempenho que suportam todo o processo de desenvolvimento desenhado.

Em seguida, o modelo conceitual foi formulado. Tal modelo buscou alinhar os conceitos necessários para a construção de um *software* ao sistema de medição de desempenho escolhido, juntamente com seus indicadores.

A próxima etapa do projeto foi a de apresentação do modelo conceitual para as empresas selecionadas. Tinha-se como objetivo a validação da aplicabilidade do modelo sugerido com o objetivo de aprofundar o conhecimento acerca do problema e evoluir a teoria [46].

O modelo foi apresentado em cinco empresas que possuem equipes de desenvolvimento de *software.* Essas empresas tinham características distintas, mas todas trabalhavam com o foco de desenvolvimento de software. Um protocolo foi criado para conduzir cada um dos casos e um piloto dessa etapa foi realizado, na primeira empresa, com o objetivo de verificar se os procedimentos com base no protocolo precisariam ser aprimorados [46].

Conseguiu-se com as apresentações atingir empresas de pequeno, médio e grande porte, com participação nacional e internacional, indústrias, departamentos de TI e consultorias em desenvolvimento de software. Após a apresentação, um questionário semiestruturado foi aplicado para todos os participantes, com o objetivo de detectar a percepção e avaliação dos mesmos em relação à aplicabilidade do modelo proposto [47].

Todas as questões levantadas durante o processo de apresentação do modelo foram compiladas e resultaram em um conjunto de alterações que o modelo sofreu, sempre validando a veracidade de cada sugestão com a teoria.

IV. RESULTADOS

O resultado desse artigo é o modelo GMQA – Gestão, Medição, Qualidade e Geração de Artefatos, conceitualmente idealizado com base em um conjunto de outros modelos.

A. Modelo Conceitual

Apesar de uma série de modelos e práticas já terem sido apresentados e aprimorados na Engenharia de *Software* ao longo dos anos, não se pode dizer que somente tais práticas resolverão todos os problemas da Engenharia de *Software*, uma vez que, apesar delas existirem, números mostram que os projetos de *software* ainda falham. Os mesmos números indicam que a utilização de conceitos ágeis tem aumentado o número de sucessos em projetos de *software*.

Partindo dessa ideia, o Modelo GMQA busca unir conceitos, sem se preocupar com a questão de seguir uma determinada tendência de mercado, mas sim garantir que as práticas atualmente conhecidas sejam discutidas. Desta forma,

o modelo é embasado em quatro princípios – Gestão, Medição, Garantia da Qualidade e Geração de Artefatos, a seguir detalhados e comentados. É importante ressaltar que, de alguma forma, as normas e modelos de melhoria de processo apresentados no capítulo II tratam e mencionam os princípios escolhidos. Entretanto, os modelos de processo de desenvolvimento de software conhecidos não relacionam esses princípios da maneira como foi sugerida pelo GMQA, daí o motivo da escolha feita por esse trabalho.

Princípio de Gestão

Acredita-se que a gestão do projeto deve ser contínua, tendo os outros três princípios aplicados ao longo de um ciclo de gestão, conforme apresentado na Figura 1. Dentro desse ciclo, três fases podem ser definidas: Planejamento, Execução e Entrega. Essas etapas vão ao encontro com conceitos básicos do gerenciamento de projetos. O objetivo do planejamento está ligado diretamente a identificar o caminho que deverá ser tomado para execução da tarefa. É importante também determinar os responsáveis por cada ação. Já a atividade de execução tem relação direta com a tentativa de tornar fato o que foi planejado. Para tanto, o monitoramento deve ser uma atividade constante, idealmente sendo aplicado de forma diária. Por fim a entrega tem como objetivo verificar o resultado do que foi planejado em relação ao que foi realmente executado, apresentando o produzido até o momento. Ela é uma atividade de avaliação do conquistado, de aprendizagem e de replanejamento.

Figura 1. GMQA: Ciclo de Gestão de Projetos de *Software*.

O modelo considera que o projeto de *software* poderá ter um ou mais ciclos de gestão, organizados através da priorização de necessidades a serem implementadas. Com base em conceitos ágeis, espera-se que cada ciclo seja planejado de tal maneira que partes do *software* sejam entregues em um curto espaço de tempo, de no máximo um mês.

As atividades do projeto poderão ser executadas em um único ciclo, em caso de projetos mais simples, ou em ciclos separados, conforme ambiente em que o modelo esteja sendo aplicado. De qualquer maneira, não é recomendável a interrupção de um ciclo de gestão para atender a demandas oriundas de novas necessidades. Daí o motivo de se ter um ciclo curto

Princípio de Medição

A medição é uma prática considerada essencial no modelo GMQA para que haja controle, entendimento, avaliação e previsão das atividades que estão sendo executadas no projeto. Ela também deve ser uma atividade planejada e alinhada com os objetivos estratégicos da empresa.

Diversas medidas podem ser extraídas e observadas em um projeto, sendo que cada uma delas pode gerar trabalho/retrabalho. É importante também reportar a cada ciclo do projeto os resultados da medição e, mais do que isso, os diagnósticos e tendências sugeridos por ela.

A medição em muitos modelos é considerada atividade de suporte ao processo de desenvolvimento de software. O que o GMQA busca direcionar é que sem a medição, a gestão, a qualidade e o próprio desenvolvimento ficam sem informação suficiente para avaliar se o trabalho que está sendo executado em um projeto está de acordo com as necessidades.

A medição, portanto, passa de uma mera opção no processo de desenvolvimento de software, para algo crucial e crítico, uma vez de que sem ela não existe visibilidade no projeto. Essa visibilidade permitirá também aos participantes do projeto a visualização dos pontos críticos e das atividades que precisam ser mais bem desenvolvidas.

Em uma esfera acima de um projeto, os números coletados durante a medição poderão inclusive direcionar a empresa sobre o perfil dos projetos de desenvolvimento de software que ela está conduzindo. Os modelos de melhoria de processo CMMI-DEV e MPS.BR discutem esse uso da medição em empresas que possuem um nível mais avançado de maturidade e podem, inclusive, evoluir o processo de desenvolvimento por conta disso.

Princípio de Garantia da Qualidade

Verificar continuamente a qualidade do *software* é característica importantíssima de um processo de desenvolvimento. É sabido que, quanto mais cedo um problema é detectado e corrigido em um projeto, mais barato ele se torna. Detectar problemas relacionados ao levantamento de necessidades, ou mesmo assegurar que as necessidades levantadas estejam corretamente priorizadas de modo a

implementar o que é realmente importante pode diferenciar um projeto entre sucesso e falha.

Sendo assim, atividades de verificação e validação devem ser planejadas, executadas e reportadas em todas as macroatividades do projeto. É importante salientar que nesse artigo o processo de verificação é definido como a observação se o produto está sendo feito corretamente; já a atividade de validação observa se o produto correto está sendo construído.

Princípio de Geração de Artefatos

Qualquer documentação ou produto gerado ao longo de um projeto de *software* é chamado de artefato. A geração de artefatos durante o processo de desenvolvimento de *software* também é uma prática que resultará em um melhor controle e comprovação do andamento do projeto.

Definir quais são os artefatos a serem desenvolvidos ao longo do projeto é responsabilidade de todos os envolvidos no projeto. A escolha por apresentar mais ou menos artefatos está diretamente ligada ao tipo de projeto que está sendo executado, bem como a abordagem de análise e arquitetura definida.

Modelo GMQA

A Figura 2 apresenta o Modelo GMQA, que representa o ciclo de vida do projeto de *software*. Acredita-se que não é possível realizar a construção de um *software* profissional nos dias de hoje sem se preocupar com sua arquitetura. Também não é possível imaginar um *software* sem o seu desenvolvimento. E o *software* desenvolvido que não está em produção não irá retornar o investimento empregado. Portanto, as três macroatividades apresentadas devem ser consideradas em qualquer projeto de *software* – arquitetura, desenvolvimento e implantação.

Apesar de que visualmente está sendo representado um conjunto de tarefas em sequência, não é correto imaginar a execução do modelo como um desenvolvimento em cascata. Na verdade, o desenho representa atividades que podem até estar acontecendo paralelamente e iterativamente em um ambiente de desenvolvimento, ao longo da vida de um *software*, garantindo a sua fluidez, foco e flexibilidade, como sugerem as melhores práticas para desenvolvimento de software.

A maneira como essas atividades devem ser executadas irá depender do perfil da empresa. Uma empresa que possua departamentos específicos para cada macroatividade seguramente executará ciclos de gestão GMQA em paralelo. Já em empresas que alocam um grupo de funcionários para execução de um projeto, o ciclo de GMQA pode ser único para execução de todas as macroatividades. Empresas de pequeno porte podem ainda possuir compartilhamento de mão de obra entre as macroatividades, o que sugere que o modelo é versátil e pode ser utilizado em diversos tipos de equipe.

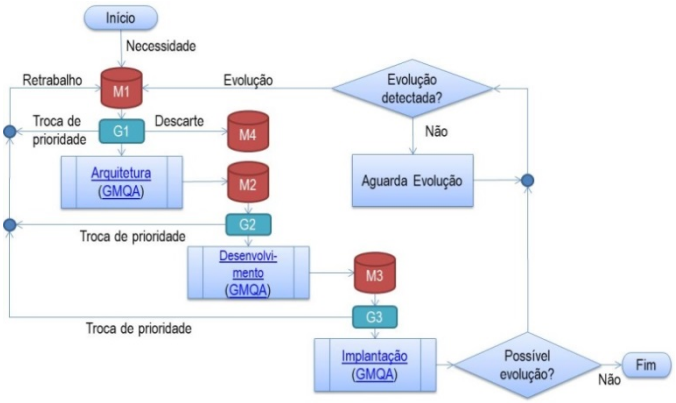

Figura 2. Modelo GMQA.

É importante destacar que o modelo além de indicar as macroatividades elementares existentes em um processo de desenvolvimento de software, apresenta os pontos mínimos de medição exigidos (M1, M2, M3 e M4), oriundos do conceito de controle de estoques, e os portões de decisão (G1, G2 e G3), oriundos do conceito de *stage-gates*.

O detalhamento das macroatividades definidas pelo GMQA com base em conceitos aplicados em diversos modelos de processo de desenvolvimento de *software* pode ser observado abaixo.

Arquitetura

A macroatividade de arquitetura é responsável pela avaliação dos pedidos encaminhados por áreas externas ao projeto ou pelo próprio cliente. Essa é a área responsável por transformar as necessidades ou melhorias contabilizadas em *software* modelado.

A área de arquitetura também deverá se preocupar com a escolha da tecnologia que será aplicada para solucionar a necessidade do cliente. Também é de responsabilidade dessa área a definição sobre o que será reutilizado. Cabe também a essa etapa a definição sobre quais partes do *software* a ser construído serão passíveis de reutilização em novos projetos.

Desenvolvimento

A macroatividade de desenvolvimento é responsável pela codificação e teste do sistema, de acordo com os requisitos sistêmicos gerados pela macroatividade de Arquitetura. Essa área também é responsável por garantir a manutenibilidade do sistema, escrevendo códigos com baixa complexidade e aderentes à abordagem de desenvolvimento selecionada.

No que diz respeito ao teste do sistema, essa área deverá definir quais são as situações que serão verificadas, bem como o conjunto de dados a ser utilizado e a cobertura dos testes executados.

Implantação

A macroatividade de implantação é a responsável pela instalação, treinamento e acompanhamento do sistema logo após a sua entrega. O planejamento dessa atividade deve observar a implantação do sistema, de acordo com as características do cliente.

A abordagem de verificação e validação da macroatividade de implantação deve ser diferente da existente na macroatividade de desenvolvimento. O foco no cliente é muito maior aqui, já que esta é a área responsável por colocar em produção o sistema desenvolvido.

O que se imagina com o modelo é que, considerando uma nova necessidade de um determinado projeto ou *software*, ela será armazenada no estoque M1 (estoque de necessidades a serem avaliadas). Após a priorização e decisão de executar a macro atividade de Arquitetura por conta do portão de decisão G1 (priorização dos requisitos de usuário que serão arquitetados), a necessidade será transformada em *software* arquitetado, ou seja, diagramas de projeto de *software* serão criados para suporta à implementação (codificação). Esse *software* arquitetado será armazenado em M2 (estoque de software arquitetado) e retirado de M1.

No momento que em G2 (definição do que será desenvolvido) se decidir que é o momento para realizar o desenvolvimento do *software* arquitetado, a macroatividade de Desenvolvimento é executada. Nesse processo, basicamente codificação e teste são realizados, entregando ao estoque G3 (definição do que será implantado) um *software* pronto para ser instalado. Nesse momento, M2 terá *software* arquitetado retirado e M3 (estoque de software construído) terá *software* pronto adicionado.

A partir daí, o *software* precisa ser colocado em produção. Essa é a responsabilidade básica da macroatividade de Implementação, que tomará a decisão por realizar a instalação ou configuração do ambiente de acordo com decisões de G3. Quando o *software* já estiver sendo utilizado pelo seu usuário final, M3 terá seu estoque livre.

É importante lembrar que a evolução de um *software* é contínua. Daí o motivo da pergunta ao final da execução da macroatividade de Implementação se é possível evoluir o *software* em questão. Quando essa possibilidade existe e uma nova necessidade é detectada, o ciclo é reiniciado.

Cabe ainda salientar que o GMQA sugere que não se interrompa um ciclo. Entretanto, quando isso ocorrer, pede-se que a necessidade que teve seu desenvolvimento interrompido seja levada ao estoque M1 para futura análise. Se em um momento oportuno for considerado que essa necessidade não é mais útil para o *software* em questão, ela deve ser movida para M4 (estoque de software descartado).

B. Ciclo de Apresentações do Modelo GMQA em Empresas

Ao longo da pesquisa, o modelo foi apresentado a um conjunto de empresas de desenvolvimento de *software*, de modo a identificar pontos de melhoria do modelo, permitindo assim o seu aprimoramento e cobertura. O propósito desse item é apresentar a evolução do projeto baseado nas reuniões ocorridas. Esse item apresenta uma análise qualitativa dos resultados obtidos, considerando que o modelo foi avaliado por 11 pessoas, de cinco empresas.

Os fatores avaliados no questionário foram: (1) Os diagramas do modelo são claros; (2) O modelo apresentado pode ser considerado uma abordagem válida para observar o estoque de software em uma empresa; (3) Já houve a preocupação com a medição do estoque de software em sua empresa (4) Existe possibilidade de aplicar os conceitos apresentados no modelo em sua rotina de trabalho; (5) O modelo apresentado é próximo à rotina de trabalho que executo em minha empresa; (6) O modelo proposto afetaria o tempo de desenvolvimento de software em sua empresa positivamente (redução de tempo de desenvolvimento); e (7) O modelo proposto afetaria o atendimento das necessidades do cliente em sua empresa (melhoria na qualidade de entrega). Eles foram definidos com base em uma escala de *Likert* de cinco pontos [48]. Além disso, foi feito o questionamento sobre a existência ou não de um processo de desenvolvimento formal e o levantamento das métricas e artefatos mais utilizados nas empresas dos respondentes.

Todos os envolvidos na apresentação do modelo mencionaram estar satisfeitos quanto aos diagramas expostos (73% sim, 27% com certeza sim). Todos também visualizaram o modelo como uma abordagem válida para observar o estoque de *software* em uma empresa (73% sim, 27% com certeza sim). No que diz respeito à medição, 45% dos respondentes disseram já ter havido tal preocupação na empresa.

Grande parte dos respondentes (82% sim, 9% com certeza sim) disse que o modelo poderia ser aplicado nas rotinas de trabalho. Entretanto, atualmente 55% dizem que o modelo apresentado é próximo dessa rotina.

Além disso, 73% afirmaram que o modelo proposto afetaria o tempo de desenvolvimento de *software* na empresa, de forma positiva, reduzindo o tempo de desenvolvimento (64% sim, 9% com certeza sim). E ainda, todos têm certeza que o modelo, se adaptado a um processo próximo da realidade da empresa, afetaria o atendimento das necessidades do cliente, melhorando a qualidade de entrega (91% sim, 9% com certeza sim).

V. CONCLUSÃO

O objetivo de desenvolver um modelo de processo de desenvolvimento de *software* integrado a conceitos de um

sistema de medição de desempenho foi atingido, visto que o modelo resultante da pesquisa tem preocupações explícitas com a medição de desempenho e foi construído em função de um considerável referencial teórico.

O referencial mencionado pode ser também considerado um objetivo alcançado pela pesquisa. Foi realizada uma avaliação profunda no que diz respeito a modelos de processo de desenvolvimento de software, normas para construção de software e modelos de melhoria de processo de software. Dessa forma, o trabalho pode contribuir em estudos futuros que precisem de informações relacionadas a esses conhecidos conceitos acadêmicos e do mercado.

Além disso, a metodologia aplicada na pesquisa é passível de adaptação em outros estudos. Acredita-se que a maneira como o modelo foi construído pode ser praticado em outras áreas, respeitando obviamente processos mais bem verificados e utilizados. O caráter prático que foi trazido ao trabalho por conta da visita nas empresas valorizou toda a pesquisa acadêmica realizada, mostrando que academia e prática alinhadas tornam os modelos mais verdadeiros.

Essa percepção pode ser observada ao longo das apresentações do modelo nas empresas do segmento de desenvolvimento de *software*. Apesar do pequeno número de apresentações e de tais terem sido feitas em uma região específica, o que pode ser dito como limitação do estudo, o retorno daqueles que avaliaram o modelo foi positivo.

Entende-se também que o modelo tem grande contribuição no que diz respeito ao entendimento do processo de desenvolvimento de *software* de uma empresa. É válido afirmar que os estoques apresentados pelo modelo foram comumente detectados nas empresas estudadas e que existem grandes indícios de que esses são estoques comuns em todas as empresas de desenvolvimento de *software*. Entretanto, por limitação da pesquisa, não é possível generalizar tal afirmação. Um estudo quantitativo mais aprofundado poderia ser aplicado para aumentar a certeza dessa hipótese. Também é válido considerar como estudo futuro a criação de um *software* que faça o controle automatizado dos estoques apresentados pelo modelo, a fim de que o acompanhamento de tais números se torne menos dispendioso.

O modelo pode ainda ser utilizado como ferramenta para definição de equipe e até mesmo como indicador para determinar a capacidade produtiva de uma empresa no que tange o desenvolvimento de *software*. Como não foi esse o foco da pesquisa, é também possível derivar um estudo para essa vertente.

Vale ressaltar que o modelo não foi utilizado em projetos de *software* profissionais, visto o curto período de tempo da pesquisa para assegurar sua eficácia e eficiência. Portanto, uma nova pesquisa pode ser conduzida para construir e aplicar um processo de desenvolvimento referência, que possa ser diretamente utilizado em empresas de pequeno, médio e até grande porte, desde que haja a preocupação com a adaptação para cada uma das realidades.

Nessa situação é sempre bom lembrar que um modelo de processo de desenvolvimento de software não pode ser simplesmente ser aplicado dentro de uma empresa. Ele deve ser adaptado, respeitando costumes, pessoas e práticas, de forma a criar um ambiente saudável de melhoria de processo.

REFERÊNCIAS

[1] E. W. Dijkstra, “The humble programmer,” em *Communications of the ACM*, ACM, 1972.

[2] W. Humphrey, A discipline for software engineering. SEI series in software engineering, Pittsburgh: Addison-Wesley, 1995.

[3] I. Sommerville, Engenharia de Software, 9ª ed., São Paulo: Pearson Prentice Hall, 2011.

[4] Standish, “CHAOS Summary 2009,” Boston, 2009.

[5] A. M. Magdaleno, C. M. L. Werner e R. M. d. Araujo, “Reconciling software development models: A quasi-systematic review,” *The Journal of Systems and Software,* 2011.

[6] K. Petersen e C. Wohlin, “Software process improvement through the Lean Measurement (SPI-LEAM) method,” *The Journal of Systems and Software,* 2010.

[7] M. Poppendieck e T. Poppendieck, Implementando o desenvolvimento Lean de Software: do conceito ao dinheiro, Porto Alegre: Bookman, 2011.

[8] M. E. Fagan, “Advances in Software Inspections,” *IEEE Transactions on Software Engineering,* 1986.

[9] R. S. Pressman, Engenharia de Software: Uma abordagem profissional, 7ª ed., Porto Alegre: AMGH, 2011.

[10] ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, “ABNT NBR ISO 9001:2008 - Sistemas de gestão da qualidade - Requisitos,” Rio de Janeiro, 2008.

[11] ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, “ABNT NBR ISO/IEC 12207:2009 Tecnologia da Informação – Processos de ciclo de vida de software,” Rio de Janeiro, 2009.

[12] ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, “ABNT NBR ISO/IEC 9126-1:2003 Engenharia de Software – Qualidade de produto Parte 1: Modelo de qualidade,” Rio de Janeiro, 2003.

[13] ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, “ABNT NBR ISO/IEC 15504-5:2008 Tecnologia da informação - Avaliação de processo - Parte 5: Um exemplo de Modelo de Avaliação de Processo,” Rio de Janeiro, 2008.

[14] Software Engineering Institute, “CMMI® for Development V1.3,” Pittsburgh, 2010.

[15] SOFTEX, “MPS.BR – Melhoria de Processo do Software Brasileiro: Guia Geral,” Campinas, 2011.

[16] A. G. Cordeiro e A. L. P. Freitas, “O cenário atual da qualidade de software,” em *XV Simpósio de Engenharia de Produção* , Bauru, 2008.

[17] A. Anacleto, C. G. von Wangenheim e C. F. Salviano, “Um Método de Avaliação de Processos de Software em Micro e Pequenas Empresas,” em *Simpósio Brasileiro de Qualidade de Software*, Porto Alegre, 2005.

[18] F. J. Pino, C. Pardo e F. Garc, “Assessment methodology for software process improvement in small organizations,” *Information and Software Technology,* 2010.

[19] M. Y. al-Tarawneh, M. S. Abdullah e A. B. M. Ali, “A Proposed Methodology for Establishing Software Process Development Improvement for Small Software Development Firms,” *Procedia Computer Science 3,* 2011.

[20] Ministério da Ciência e Tecnologia, “Pesquisa de Qualidade no Setor de Software Brasileiro 2009,” Brasília, 2010.

[21] Ö. ERALP, "Design and implementation of a software development process measurement system.," Ankara, 2004.

[22] J. GANSSLE, "Developing a good bedside manner," *Embedded Systems Design,* 2009.

[23] C. Jones, Applied Software Measurement: Global Analysis of Productivity and Quality, New York: McGraw-Hill, 2008.

[24] NATO, "Report on a Conference sponsored by the NATO Science Committee," Garmisch, Germany, 1969.

[25] G. Booch, "Leaving Kansas," *IEEE Software,* 1998.

[26] B. W. Boehm, "A Spiral Model of Software Development and Enhancement," *IEEE Compute,* 1988.

[27] W. W. Royce, "Managing the Development of Large Software Systems," em *IEEE WESCON*, WESCON, 1970.

[28] P. Kruchten, The Rational Unified Process – An Introduction, the U.S.A.: Addison-Wesley, 2003.

[29] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries e J. Kern, 2001. [Online]. Available: http://agilemanifesto.org/. [Acesso em 09 06 2012].

[30] K. Schwaber e J. Sutherland, The Scrum Guide: The Definitive Guide to Scrum, the U.S.A.: Scrum.org, 2011.

[31] J. Berrocal, J. G. Alonso, C. V. Chicote e J. M. Murillo, "A Model-Driven Approach for Documenting Business and Requirements Interdependencies for Architectural Decision Making," *IEEE LATIN AMERICA TRANSACTIONS,* 2014.

[32] T. Dyba e T. Dingsøyr, "Empirical studies of agile software development: A systematic review," *Information and Software Technology,* 2008.

[33] B. R. Staats, D. J. Brunner e D. M. Upton, "Lean principles, learning, and knowledge work: Evidence from a software services provider," *Journal of Operations Management,* 2010.

[34] N. Slack, S. Chambers e R. Johnston, Administração da Produção, São Paulo: Atlas, 2009.

[35] A. NEELY, M. GREGORY e K. PLATTS, "Performance measurement system design: A literature review and research agenda," *International Journal of Operations and Production Management,* 1995.

[36] M. A. El-Baz, "Fuzzy performance measurement of a supply chain in manufacturing companies," *Expert Systems with Applications,* 2011.

[37] R. Bhagwat e M. K. Sharma, "Performance measurement of supply chain management: A balanced scorecard approach.," *Computers & Industrial Engineering,* 2007.

[38] W. S. Humphrey, Managing the software process, Pittsburgh: Addison-Wesley Professional, 1989.

[39] R. S. Kaplan e D. P. Norton, A Estratégia em Ação - Balanced Scorecard, Rio de Janeiro: Campus, 1997.

[40] ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, "ABNT NBR ISO/IEC 15939:2009 Engenharia de sistemas e de software - Processo de medição," Rio de Janeiro, 2009.

[41] SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, "Melhorias nos processos de software," *Computação Brasil,* n. 14, pp. 14-16, 2010.

[42] R. SOLINGEN e E. BERGHOUT, The Goal/Question/Metric Method: A Practical Guide for Quality Improvement of Software Development, London: McGraw-Hill, 1999.

[43] R. G. Cooper, "Doing it Right: Winning with New Products," Hamilton, 2007.

[44] D. Karlström e P. Runeson, "Integrating agile software development into stage-gate managed product development," *Empirical Software Engineering,* 2006.

[45] I. T. M. Cutovoi, "Metodologia de desenvolvimento de produtos com foco na logística de abastecimento aplicada a uma montadora de motores," São Paulo, 2012.

[46] P. A. C. MIGUEL, "Estudo de caso na engenharia de produção: estruturação e recomendações para sua condução," *Produção,* vol. 17, n. 1, 2007.

[47] A. J. SEVERINO, Metodologia do trabalho científico, 23ª ed., São Paulo: Cortez, 2007.

[48] J. W. C. Alexandre, D. F. d. Andrade, A. P. d. Vasconcelos, A. M. S. d. Araujo e M. . J. Batista, "Análise do número de categorias da escala de Likert aplicada à gestão pela qualidade total através da teoria da resposta ao item," em *XXIII Encontro Nac. de Eng. de Produção*, Ouro Preto, 2003.

[49] J. Sutherland e K. Schwaber, The Scrum Papers: Nut, Bolts, and Origins of an Agile Framework, Paris: ScrumInc., 2011.

[50] K. Beck, "Embracing Change with Extreme Programmin," *IEEE Computer,* 1999.

[51] H. Atkinson, "Strategy implementation: a role or the balanced scorecard?," *Management Decision,* pp. 1441-1460, 2006.

[52] M. A. S. MACEDO, A. C. T. d. A. M. BARBOSA e G. T. Cavalcante, "Desempenho de agências bancárias no Brasil: aplicando análise envoltória de dados (DEA) a indicadores relacionados às perspectivas do BSC," *Revista Economia e Gestão,* vol. 9, n. 19, 2009.

[53] C. F. Salviano, "Projetos da CE-21-007-10 e Novidades da ISO/IEC 15504 (SPICE)," São Paulo, 2009.

[54] E. Y. Li, H.-G. Chen e W. Cheung, "Total Quality Management in Software Development Process," *The Journal of Quality Assurance Institute,* 2000.

[55] A. Koscianski e M. SOARES, Qualidade de software: aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software, 2ª ed., São Paulo: Novatec, 2007.

[56] A. BARBOSA, L. MENDES e M. BOTTOLI, "Análise comparativa de metodologias e padrões de qualdiade com foco no gerenciamento de projetos de software.," *Revista Ciência e Tecnologia,* 2010.

[57] Project Management Institute, "Um Guia do Conjunto de Conhecimentos em Gerenciamento de Projetos (PMBOK)," Newtown Square, 2004.

[58] M. NOGUEIRA e J. M. ABE, "Paradigmas da Engenharia de Software como fatores críticos de sucesso para a gestão de projetos de software no contexto brasileiro," em *XVIII SIMPÓSIO DE ENGENHARIA DE PRODUÇÃO*, Bauru, 2010.

Gabriel Lara Baptista é Professor dos Cursos de Graduação em Informática na Universidade Nove de Julho – UNINOVE. Graduado em Ciência da Computação e Mestre em Engenharia de Produção pela Universidade Nove de Julho - UNINOVE. E-mail: gabriel.baptista@uninove.br.

Rosangela Maria Vanalle é Professora do Programa de Pós-Graduação em Engenharia de Produção na Universidade Nove de Julho - UNINOVE. Graduada em Engenharia de Produção Química pela Universidade Federal de São Carlos, Doutora em Engenharia Mecânica pela Universidade de São Paulo/EESC e pós-doutorado pela Universidad Complutense de Madrid. E-mail: rvanalle@uninove.br

José Antonio Arantes Salles é Professor do Programa de Pós-Graduação em Engenharia de Produção na Universidade Nove de Julho - UNINOVE. Graduado em Engenharia de Produção pela Universidade de São Paulo, Doutorado em Administração de Empresas pela Fundação Getúlio Vargas e pós-doutorado pela Universidad Complutense de Madrid, E-mail: salles@uninove.br.