UMA PROPOSTA DE MODELO DE PROCESSO DE DESENVOLVIMENTO DE SOFTWARE COM BASE EM CONCEITOS DE ENGENHARIA DE PRODUÇÃO

Gabriel Lara Baptista (UNINOVE)
gabriel.baptista@uninove.br
Jose Antonio Arantes Salles (UNINOVE)
salles@uninove.br

A engenharia de software é uma disciplina relativamente nova se comparada a outras engenharias existentes. Tal fato leva à dificuldade de acerto de custos, cumprimento de prazos e atendimento de funcionalidades. Visto este cenário, a práticca de utilização de técnicas de gestão existentes em outras engenharias é visualizada ao longo do histórico da engenharia de software. O presente trabalho propõe um modelo de processo de desenvolvimento de software que busca conceitos teóricos na engenharia de produção para solucionar questões de gestão de projeto de desenvolvimento de software, com grande ênfase no conceito de sistema de medição de desempenho. O resultado da pesquisa sugere um modelo que ainda depende de avaliação prática mas que esta alinhado a bons conceitos de desenvolvimento de software. O trabalho ainda apresenta pesquisas que podem ser construídas com base no modelo proposto.

Palavras-chaves: modelo de processo de software, sistema de medição de desempenho, técnicas de gestão, CMMI-DEV, MPS.BR

1. Introdução

O desenvolvimento de software pode ser considerado uma das mais novas engenharias existentes no mundo (DIJKSTRA, 1972). Tal situação leva a considerar que muitos problemas resolvidos em outras áreas ainda são dificuldades dessa indústria. Questões relacionadas a conseguir atingir prazos, custos e funcionalidades esperados são desafios mencionados no passado e ainda enfrentados nos dias de hoje (DIJKSTRA, 1972; HUMPHREY, 1995; KOSCIANSKI e SOARES, 2007; STANDISH GROUP, 2009; SOMMERVILLE, 2011).

Ao longo dos anos, uma série de modelos vem sendo apresentados na Engenharia de Software justamente para tentar resolver os problemas citados acima (ROYCE, 1970; BOEHM, 1988; BECK, 1999; KRUTCHEN, 2003; PRESSMAN, 2011; SOMMERVILLE, 2011). Também é sabido que alguns desses paradigmas utilizaram-se de técnicas já concebidas na Engenharia de Produção, como o desenvolvimento *Lean* (PETERSEN e WOHLIN, 2010; POPPENDIECK e POPPENDIECK, 2011) e a inspeção (FAGAN, 1986) para resolver questões da Engenharia de Software.

Partindo desse princípio, o presente artigo apresenta o modelo GMQA – Gestão, Medição, Qualidade e seus Artefatos. Esse modelo surge da iniciativa de observar conceitos existentes na Engenharia de Produção e vinculá-los às práticas existentes no desenvolvimento de software, tendo como foco a definição um modelo que abranja e integre os conceitos de um sistema de medição de desempenho conhecidos na Engenharia de Produção. O trabalho define modelos de processo de desenvolvimento de software como base teórica possível de adaptação para que as empresas de desenvolvimento possam criar os processos para produção do software adequados para cada realidade (PRESSMAN, 2011).

O artigo está dividido em cinco seções, sendo a primeira a introdução aqui apresentada. A segunda seção descreve conceitos utilizados para concepção do modelo. A terceira apresenta a metodologia utilizada na construção do modelo, que está descrito na quarta seção. Finalmente na quinta seção são apresentadas as considerações finais do trabalho, sugerindo um conjunto de trabalhos futuros.

2. Embasamento Teórico

2.1. Modelos de Processo de Desenvolvimento de Software

Inúmeras são as abordagens existentes para a produção de software. Desde o surgimento do termo Engenharia de Software, foram discutidas diversas maneiras para tentar obter o software com um nível de qualidade adequado (NATO, 1969).

Boehm (1988) afirma que o primeiro modelo conhecido para produção de software era chamado "Codifique / Corrija". Nesse modelo, questões como detalhamento de necessidades, análise, teste e manutenção eram levantadas após a codificação. A abordagem resultava em códigos mal escritos, necessidades de usuário não atingidas e alto custo de manutenção.

A abordagem que substituiu o Modelo Codifique / Corrija é chamada de Modelo Cascata (ROYCE, 1970). Sommerville (2011) comenta que esse modelo considera as atividades

fundamentais do processo de especificação, desenvolvimento, validação e evolução do software.

Boehm é um dos críticos do modelo Cascata e em seu trabalho que apresenta o modelo Espiral (BOEHM, 1988), processo que ficou conhecido pela sua capacidade de iteração e preocupação com a gestão de projetos (SOMMERVILLE, 2011).

Do conceito de iteração apresentado pelo modelo espiral, surgiram outros modelos. Técnicas como prototipação e entrega incremental foram desenvolvidas, utilizando-se justamente da abordagem iterativa (SOMMERVILLE, 2011). Outro processo muito conhecido que surgiu no início dos anos 2000 foi o *Rational Unified Process* – RUP (KRUTCHEN, 2003). Esse modelo foi baseado nas melhores práticas para desenvolvimento de software, dentre elas o próprio desenvolvimento iterativo.

Todos os modelos até aqui apresentados são considerados tradicionais na Engenharia de Software, dirigidos essencialmente ao planejamento. Após a publicação do Manifesto Ágil (BECK, 2001), inúmeras iniciativas que se preocupavam mais com questões relacionadas aos princípios defendidos de resposta à mudança, colaboração do cliente, software em funcionamento e indivíduos e interações surgiram para criticar a até então única Engenharia de Software. O *Extreme Programming* (BECK, 1999) e o *Scrum* (SCHWABER e SUTHERLAND, 2011) são exemplos de modelos ágeis considerados atualmente importantes no desenvolvimento de software, sendo que o primeiro foca a produção de software e o segundo se preocupa com a gestão de projetos de software.

2.2. Conceito sobre Sistema de Medição de Desempenho

Slack et. al. (2009) afirma que a medição de desempenho é um processo de quantificar a ação, no qual medição relaciona-se com o processo de quantificação e o desempenho é presumido como derivado de ações tomadas ao longo da produção.

Quanto à construção de um Sistema de Medição de Desempenho, El-Baz (2010) apresenta uma adaptação de Neely et. al. (1995) em relação aos nove passos típicos e necessários. A atividade inicia-se com a definição da missão da empresa. Em seguida, os objetivos estratégicos da empresa devem ser mapeados, utilizando-se da missão definida anteriormente como guia.

Neely et. al. (1995) afirma que se deve também desenvolver um entendimento de cada uma das áreas funcionais em relação aos objetivos estratégicos, determinando medições de desempenho globais para cada uma das áreas. A comunicação dos objetivos estratégicos deve ser executada para cada nível da empresa e a garantia de consistência desses objetivos com o critério de desempenho de cada nível deve ser realizada.

Por fim, deve haver a garantia de compatibilidade das medidas de desempenho de cada área e o SMD deve ser utilizado, sempre sendo reavaliado, em função das necessidades da empresa e o ambiente competitivo atual.

Entretanto, mesmo com passos pré-definidos para construção de um SMD, o alinhamento entre a estratégia e as operações da empresa pode ser considerado um dos fatores críticos para a sua implementação (ATKINSON, 2006; BHAGWAT e SHARMA, 2007; MACEDO, 2009).

No que se diz respeito à produção de software, Humphrey (1989) afirma que são quatro os principais papéis da medição – Controlar; Avaliar; Entender; e Prever.

Sabe-se, portanto, que um sistema de medição de desempenho adequado trará benefícios à corporação que, se alinhados com sua estratégia, auxiliarão no alcance das metas estratégicas (KAPLAN e NORTON, 1997).

Quando se fala de controle de software baseado em métricas, a utilização de uma base histórica se faz necessária. Humphrey (1995) apresenta em seu livro dedicado ao desenvolvimento pessoal de software a necessidade de armazenamento de um histórico, o que sugere tal prática como característica dos sistemas de medição de desempenho para projetos de software.

Pressman (2011) enfatiza também a importância de utilização métricas de software para entendimento e gestão de processos e produtos de software. Ele afirma que a única maneira racional de melhorar qualquer processo é medir atributos específicos do processo.

De forma contraditória ao apresentado até agora pela bibliografia levantada, que destaca o uso das métricas para gerenciamento de projetos de software, está uma pesquisa levantada pelo Ministério da Ciência e Tecnologia, que afirma que somente 39,6% das empresas pesquisadas medem o desempenho do processo de software de forma sistemática (MCT, 2009)

2.3. Normas e modelos de processos que apoiam a medição de desempenho

A Engenharia de Software possui um conjunto de normas e processo que apoiam a medição de desempenho. Dentre elas, pode-se destacar a norma ISO 12207 (ABNT, 1998), a norma ISO 15504 (ABNT, 2008), o modelo internacional CMMI-DEV – Capability Maturity Model Integration for Development (SEI, 2010) e o modelo nacional MPS.BR – Melhoria de Processo do Software Brasileiro (SOFTEX, 2011).

A NBR ISO/IEC 12207:1998 (ABNT, 1998) é a versão brasileira da norma que trata da questão de processos de ciclo de vida de software. Koscianski (2007) afirma que essa norma oferece uma estrutura para que uma organização defina os seus processos de produção de software, cobrindo todo seu ciclo de vida, de requisitos até a manutenção e retirada do produto. A norma requer em suas atividades que a empresa se preocupe em medir os processos de desenvolvimento de software, armazenando dados históricos dos projetos, de modo a entender os processos que está executando.

A norma ISO/IEC 15504:2008 (ABNT, 2008) trata da questão de avaliação do processo de desenvolvimento de software. Atualmente a norma possui sete partes, já traduzidas para o português. Salviano (2009) afirma que as partes oito e nove estão em desenvolvimento, sendo que existe um projeto atual para realizar a evolução da atual norma para uma série ISO/IEC 33000.

Anacleto (2005) afirma que a norma ISO 15504 presta-se a realização de avaliações dos processos com dois objetivos, melhorar e determinar a capacidade dos processos. Para tanto, ela define o conceito de nível de capacidade, que aponta o quão capaz é a empresa em relação à execução do processo avaliado. A norma apresenta seis níveis de capacidade, que vão de incompleto a otimizado, sendo que o seu quarto nível refere-se diretamente a empresas que tem a capacidade de prever resultados a partir de medições detalhadas e analisadas. A norma

ainda conta com uma área de processo dedicada à medição, considerada base para os modelos CMMI-DEV e MPS.BR (SEI, 2010; SOFTEX, 2011).

O CMMI-DEV é atualmente o framework para melhoria de processo de desenvolvimento de software mais conhecido e aceito mundialmente. O modelo atualmente encontra-se em sua versão 1.3 e é composto por 22 áreas de processo, que estão posicionadas em cinco níveis de maturidade ou seis níveis de capacidade, dependendo do tipo de representação utilizado. Ele está embasado nas propostas das normas ISO 9000, ISO/IEC 12207, ISO/IEC 15504, entre outras (SEI, 2010).

Já se faz obrigatório a partir do Nível 2 de maturidade do CMMI-DEV a aplicação de medição e análise nos projetos de software. Segundo o seu documento guia, a escolha das métricas de um projeto devem estar alinhadas com as necessidades da organização e do projeto. O modo de coleta das medições também deve ser definido, bem como a maneira como as métricas serão armazenadas.

O MPS.BR foi fruto de uma iniciativa brasileira ocorrida no final do ano de 2003, coordenada pela Associação para Produção da Excelência do Software Brasileiro (SOFTEX) e apoiada pelo Ministério de Ciência e Tecnologia (MCT), além de outras entidades governamentais e privadas (SOFTEX, 2011) com objetivo principal a melhoria de processo do software brasileiro, compatibilizando custos de implantação com o cenário brasileiro (SBC, 2010).

Nesse modelo, a área de processo de medição tem como propósito a coleta, o armazenamento, a análise e o relato dos dados relativos a produtos e processos da organização, também apoiando os objetivos da empresa (SOFTEX, 2011). O modelo deixa claro ainda que o foco principal da medição seja o de apoiar a tomada de decisão em relação aos projetos e processos, sendo as métricas criadas a partir de necessidades estratégicas de informação da organização. Ele sugere ainda a utilização de um método de medição como, por exemplo, o GQM (Goal-Question-Metric), idealizado por Solingen e Berghout (1999).

3. Metodologia

A construção do modelo foi realizada com base em um levantamento bibliográfico estruturado e sistemático sobre modelos de processo de desenvolvimento de software e sistemas de medição de desempenho.

Desse levantamento, uma análise crítica das metodologias de desenvolvimento de software e sistemas de medição de desempenho foi realizada, criando um mapeamento entre técnicas, normas e modelos que poderiam ser utilizados. Essa etapa serviu de base para a escolha dos princípios do modelo conceitual, associando os conceitos de Processo de Desenvolvimento de Software e Sistemas de Medição de Desempenho.

Em seguida, o modelo conceitual foi formulado. Tal modelo busca alinhar os conceitos necessários para a construção de um software ao sistema de medição de desempenho escolhido. O modelo proposto já é fruto de duas rodadas de apresentação em empresas que desenvolvem software, com o objetivo de detectar a percepção e avaliação de profissionais da área em relação à aplicabilidade do modelo proposto (SEVERINO, 2007).

4. Apresentação do Modelo

4.1. Ciclo de Vida do Projeto de Software

A Figura 1 apresenta o ciclo de vida do projeto de software considerado pelo modelo. Esse ciclo possui dois conceitos já conhecidos na Engenharia de Produção. O primeiro está relacionado com a contabilização de medições relativas aos estoques (M1, M2, M3 e M4), originário do Lean (POPPENDIECK e POPPENDIECK, 2007; PETERSEN e WOHLIN, 2010; SCHWABER e SUTHERLAND, 2011).

Outro fator ligado à Engenharia de Produção tem relação com a tomada de decisão em alguns pontos estratégicos do processo. Essa avaliação foi definida com base nos conceitos de stage-gates, oriundos do desenvolvimento de produtos (COOPER, 1993; KARLSTRÖM e RUNESON, 2006; COOPER, 2007).

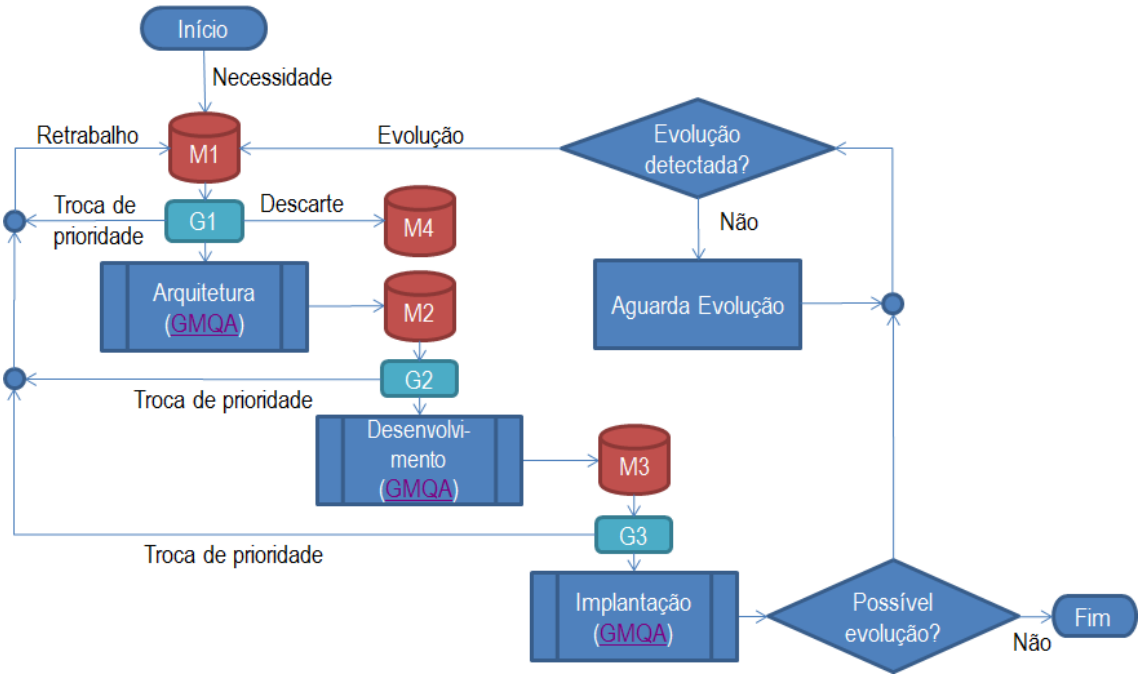

Figura 1 – Ciclo de vida do projeto de software para o modelo GMQA

Apesar de o modelo apresentar um conjunto de tarefas em sequência, não é correto imaginá-lo como um desenvolvimento em cascata. Na verdade, ele é apenas a representação de atividades que podem até estar acontecendo paralelamente em um ambiente de desenvolvimento, ao longo da vida de um software, garantindo a sua fluidez, decisões difusas, foco e flexibilidade, conforme mencionado pelo modelo defendido de Cooper (1993) para desenvolvimento de produto.

O que ele na verdade representa são macro etapas elementares existentes em um processo de desenvolvimento de software, com os pontos mínimos de medição exigidos (M1, M2, M3 e M4) e os portões de decisão (G1, G2 e G3).

O modelo acredita 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 com certeza não valeu nem um centavo do investimento. Portanto, as três macro etapas

apresentadas deveriam ser consideradas em qualquer projeto de software – Arquitetura, Desenvolvimento e Implantação.

4.2. Princípios do Modelo

O modelo acredita 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 2. Dentro desse ciclo, três fases podem ser definidas: Planejamento, Execução e Entrega. Estas etapas vão de encontro com conceitos básicos do gerenciamento de projetos (PMI, 2004).

Figura 2 – Ciclo de Gestão de Projetos de Software GMQA

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, de acordo com o portão G1. Com base em conceitos ágeis, espera-se que cada ciclo seja planejado de tal maneira que sejam entregues partes do software ao final de um curto espaço de tempo, de no máximo um mês (SCHWABER e SUTHERLAND, 2011).

As macro etapas 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 no meio para atender a demandas oriundas de novas necessidades. Daí o motivo de se ter um ciclo curto (SCHWABER e SUTHERLAND, 2011).

4.3. Pontos mínimos de medição exigidos

O modelo apresenta quatro pontos de medição exigidos, chamados de M1, M2, M3 e M4. O conceito por trás destes pontos tem relação com a avaliação de estoques, base dos princípios Lean (PETERSEN e WOHLIN, 2010). Deve-se, portanto, considerar cada um desses marcos um estoque do ciclo de vida de desenvolvimento do software. São eles:

M1 – Estoque de necessidades a serem avaliadas. Pode ser medido por quantidade de requisitos de usuário pendentes.

M2 – Estoque de software arquitetado: Esse estoque possui documentação essencial para construção do software, tais como requisitos de sistema definidos, custo para desenvolvimento, e até modificações na arquitetura planejada. Entretanto, o desenvolvimento dessa arquitetura ainda não foi executado por conta de falta de recursos ou por falta de aprovação de custo. Pode ser medido pela somatória da quantidade de requisitos sistêmicos aprovados com os requisitos pendentes de aprovação.

M3 – Estoque de software construído: Nesse terceiro estoque encontra-se software considerado pronto para instalação, mas que não foi ainda entregue ao seu usuário final, seja por falta de treinamento ou falta de recursos para instalação. Pode ser medido pela soma da quantidade de requisitos pendentes de implantação, classificados como exemplificado.

M4 – Estoque de software descartado: No quarto estoque encontra-se software que foi inicialmente considerado válido, mas que por mudanças internas ou externas ao projeto, não pôde ser implementado por conta de não mais fazer parte da realidade do sistema.

Poppendieck e Poppendieck (2007) afirmam que o estoque em desenvolvimento de software equivale a trabalhos parcialmente acabados. Um software parcialmente acabado tem todos os males de um estoque na produção: se perde, fica obsoleto, esconde problemas de qualidade e estagna dinheiro. Sendo assim, a correta e contínua priorização de quais itens do estoque M1 serão produzidos irá interferir diretamente na eficiência do processo. Essa decisão está relacionada ao portão G1.

Os conceitos Lean deixam claro que o aumento dos volumes em estoque simboliza desperdício e, por consequência, prejuízo (POPPENDIECK e POPPENDIECK, 2007; PETERSEN e WOHLIN, 2010; SCHWABER e SUTHERLAND, 2011). É interessante, portanto, o monitoramento de cada um desses estoques de modo a garantir uma estratégia correta de execução de tarefas. Espera-se que, com a medição desses pontos, os envolvidos no processo de desenvolvimento tenham conhecimento de quais serão os pontos que devem ser medidos para evoluir o processo.

4.4. Portões de decisão

Após a execução de um ciclo de gestão, chega o momento de tomar decisões no projeto. Por tal razão, os portões de decisão irão usufruir de todo o material até o momento gerado. O modelo prevê inicialmente três portões.

Cada um desses pontos poderá iniciar uma nova macro etapa do projeto ou realimentar o estoque de necessidades por conta de modificações no trabalho que foi gerado ao final do ciclo de gestão. Abaixo, um conjunto de preocupações existentes em cada um dos portões de decisão:

G1 – O portão 1 é aquele responsável pela priorização das atividades que irão começar a ser arquitetadas. Ele deve estar aderente às necessidades dos clientes e da organização, de tal maneira que não se desperdice tempo com o início de atividades de arquitetura que não são primordiais para o andamento do projeto. A priorização das atividades pode ser considerada peça chave para o sucesso na execução de qualquer projeto. A mão de obra atualmente em qualquer ambiente corporativo é escassa, o que sugere que a priorização do que realmente

precisa ser desenvolvido gerará menor nível de estoques intermediários (SCHWABER e SUTHERLAND, 2011).

G2 – Ao chegar ao portão 2 deverão ser observados os estoques M1, M2 e M3 de tal maneira a tomar a decisão por Desenvolver o que está em M2, mover necessidades de M2 para M1 ou simplesmente deixar de executar a macro atividade de Desenvolvimento para atender a outros estoques, com maior demanda.

G3 – Ao chegar ao portão 3 deverão ser observados os estoques M1, M2 e M3 de tal maneira a decidir por Implantar o que está em M3, mover necessidades de M3 para M1 ou simplesmente deixar de executar a macro atividade de Implantação para atender a outros estoques, com maior demanda.

4.5. Análise de Aderência do Modelo e outras Abordagens

O modelo apresentado é fruto do estudo de uma série de diferentes abordagens, oriundas do levantamento bibliográfico realizado. A Tabela 1 mostra cada uma dessas abordagens, bem como a sua referência base.

Abordagem	Referência
Melhores práticas no desenvolvimento de software	(BOOCH, 1998)
Princípios Lean para desenvolvimento de software	(POPPENDIECK e POPPENDIECK, 2011)
Passos para desenvolver um SMD	(NEELY et. al, 1995)
Papéis da Medição	(HUMPHREY, 1989)
CMMI-DEV	(SEI, 2010)
MPS.BR	(SOFTEX, 2011)
SCRUM	(SHWABER e SUTHERLAND, 2011)
ISO 15939	(ISO, 2007)
Stage-Gates	(COOPER, 2007)

Tabela 1 – Abordagens base para concepção do modelo

Com base no levantamento das principais atividades solicitadas por esse conjunto de abordagens existentes para produção do software, a Figura 3 apresenta uma visão do quão próximo o modelo se encontra das diversas abordagens utilizadas para sua concepção.

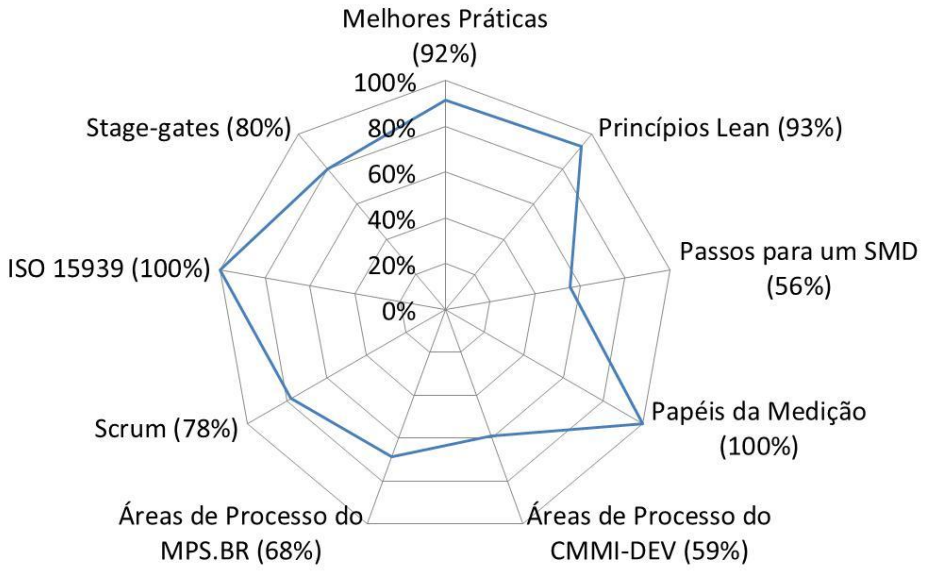

Figura 3 – Aderência do modelo a outras abordagens

5. Considerações finais

Considerando como objetivo principal do trabalho a proposta de um modelo de processo de desenvolvimento de software com base em conceitos de Engenharia de Produção, pode-se afirmar que o modelo apresentado consegue atingir essa meta.

Também é possível afirmar que a verificação de conceitos já existentes na Engenharia de Produção pode ser feita com o intuito de adaptar tais técnicas na Engenharia de Software, desde que observadas as variações existentes em um projeto nessa indústria.

Ressalta-se que o modelo ainda não foi adaptado para processos em empresas, sendo apenas testado conceitualmente. Essa deve ser considerada a principal limitação do trabalho. A aplicação em alguns ambientes poderia ser tema para estudos futuros, mas mesmo assim não seria possível afirmar que tal modelo resolveria os problemas mencionados na introdução do trabalho em sua completude.

Outro ponto que pode ser discutido em um trabalho futuro está relacionado a quais métricas e artefatos devem ou poderiam ser gerados ao longo do processo de desenvolvimento de software.

Referências

ABNT (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS). Tecnologia da informação – *Processos de ciclo de vida de software*. NBR ISO/IEC 12207, 1998.

ABNT (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS). Tecnologia da informação – *Avaliação de processo*. NBR ISO/IEC 15504, 2008.

ANACLETO, A.; WANGENHEIM, C.; SALVIANO, C. *Um método de avaliação de processos de software em micro e pequenas empresas*. SBQS - Simpósio Brasileiro de Qualidade de Software. Porto Alegre, 2005.

ATKINSON, H., *Strategy implementation: a role for the balanced scorecard?* Management Decision v.44 n. 10, 2006.

BECK, K. *Embracing Change with Extreme Programming.* IEEE Computer, 1999.

BECK, K. et. al. *Manifesto for Agile Software Development.* Utah, 2001. Acessado eletronicamente em http://www.agilemanifesto.org/.

BHAGWAT, R. e SHARMA, M. K., *Performance measurement of supply chain management*: A balanced scorecard approach. Computer & Industrial Engineering. Elsevier, 2007.

BOEHM, B. *A Spiral Model of Software Development and Enhancement.* IEEE Computer, 1988.

BOOCH, G. *Leaving Kansas.* IEEE Software 15(1), 1998.

COOPER, R. G. *Winning at new products, accelerating the process from idea to launch.* Reading, M. A., Perseus Books, 1993.

COOPER, R. G. *Doing it Right: Winning with new products.* Innovation Framework Technologies, 2007.

DIJKSTRA, E. W. *The Humble Programmer*. EWD 340. ACM 15(10): 859-866, 1972.

EL-BAZ, M. A. *Fuzzy performance measurement of a supply chain in manufacturing companies*. Expert Systems with Applications, 2010.

FAGAN, M. *Advances in Software Inspections.* IEEE Transactions on Software Engineering, 1986.

HUMPHREY, W. S. *Managing the software process.* Addison-Wesley, 1989.

HUMPHREY, W. *A discipline for software engineering.* SEI series in software engineering. Addison-Wesley, 1995.

ISO/IEC 15939-2:2007 *Systems and software engineering* – Measurement process.

KAPLAN, R. S., NORTON, D. P. *A Estratégia em Ação: Balanced Scorecard.* Rio de Janeiro: Campus, 1997.

KARLSTRÖM, D., RUNESON, P. *Integrating agile software development into stage-gate managed product Development*. Empirical Software Engineering, 2006.

KOSCIANSKI, A; SOARES, M. S. *Qualidade de software: Aprenda as metodologias e técnicas mais modernas para desenvolvimento de software.* São Paulo: Novatec, 1995.

KRUTCHEN, P. *The Rational Unified Process* – An Introduction. Addison-Wesley, 2003.

MACEDO, M. A. S. et al., *Desempenho de agências bancárias no Brasil*: aplicando análise envoltória de dados (DEA) a indicadores relacionados às perspectivas do BSC. Revista Economia e Gestão v.19 n. 19, 2009.

MCT (MINISTÉRIO DA CIÊNCIA E TECNOLOGIA) *Pesquisa de Qualidade no Setor de Software Brasileiro 2009*. Brasília, 2009.

NATO Science Committee Software Engineering *Report on a Conference sponsored by the NATO Science Committee*, Garmisch, Germany, 1969.

NEELY, A.; GREGORY, M. J.; PLATTS, K. *Performance measurement system design: A literature review and research agenda.* International Journal of Operations & Production Management, 1995.

NEELY, A; ADAMS, C; KENNERLEY, M. *The Performance Prism - The Scorecard for Measuring and ManagingBusiness Success.* Financial Times Prentice Hall, 2002.

PETERSEN, K., WOHLIN, C. *Software process improvement throught the Lean Measurement (SPI-LEAM) method*. The Journal of Systems and Software, 2010.

PMI (PROJECT MANAGEMENT INSTITUE) *Um Guia do Conjunto de Conhecimentos em Gerenciamento de Projetos (PMBOK)*, 3ª Edição, 2004.

POPPENDIECK, M., POPPENDIECK, T. *Implementando o desenvolvimento Lean de Software: do conceito ao dinheiro*. Porto Alegre: Bookman, 2011.

PRESSMAN, R. S. *Engenharia de Software*: Uma abordagem profissional. Porto Alegre: AMGH, 2011.

ROYCE, W. W. *Managing the Development of Large Software Systems*. WESCON, 1970.

SALVIANO, C. *Projetos da CE-21-007-10 e Novidades da ISO/IEC 15504 (SPICE)*. São Paulo, 2009.

SCHWABER, K., SUTHERLAND, J. *The Definitive Guide to Scrum*: The rules of the game. Scrum.org, 2011.

SEVERINO, A. J. *Metodologia do trabalho científico.* 23. ed. São Paulo: Cortez, 2007.

SEI (SOFTWARE ENGINEERING INSTITUTE) *CMMI® for Development*, Version 1.3. Pittsburgh, 2010.

SLACK, N. et. al. *Administração da Produção*. 3ª ed. São Paulo: Atlas, 2009.

SOMMERVILLE, I. *Engenharia de Software*. 9ª ed. São Paulo: Pearson Prentice Hall, 2011.

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

SOFTEX *MPS.BR – Melhoria de Processo do Software Brasileiro*: Guia Geral, 2011.

STANDISH GROUP INTERNATIONAL. *CHAOS Summary 2009 Report*. Boston, 2009.