DOI: 10.5935/1809-2667.20130035

Artigo Original

Uma análise comparativa entre as metodologias de desenvolvimento de software: Rational Unified Process e Extreme Programming

A Comparative Analysis of Two Software Development Methodologies: Rational Unified Process and Extreme Programming

Marcelo Rafael Borth*
Henrique Yoshikazu Shishido**

Mediante a grande exigência do mercado por inovação, produtividade, qualidade e desempenho dos sistemas computacionais, criaram-se as metodologias de desenvolvimento de software. A partir do uso de uma metodologia de software é possível reduzir o custo, o risco e o tempo do desenvolvimento de um projeto e, ainda, aumentar a qualidade do produto final. Este artigo realiza uma comparação entre duas metodologias de desenvolvimento: o *Rational Unified Process* e o *Extreme Programming*. A comparação realizada mostra as principais semelhanças e contrastes entre as abordagens, destacando e comentando suas características predominantes.

Palavras-chave: Metodologias de Desenvolvimento de Software. Rational Unified Process. Extreme Programming.

Software development methodologies were created to meet the great market demand for innovation, productivity, quality and performance. With the use of a methodology, it is possible to reduce the cost, the risk, the development time, and even increase the quality of the final product. This article compares two of these development methodologies: the Rational Unified Process and the Extreme Programming. The comparison shows the main differences and similarities between the two approaches, and highlights and comments some of their predominant features.

Keywords: Software Development Dethodology. Rational Unified Process. Extreme Programming.

Introdução

As empresas de desenvolvimento de software enfrentam desafios e muita competitividade nos dias atuais. De forma geral, isso está relacionado à volatilidade dos requisitos de software (BECK, 2000), (BOEHM; TURNER, 2005), (PIKKARAINEN et al., 2008). Assim, torna-se necessário criar um planejamento estratégico para fornecer

* Doutorando em Ciências Ambientais e Sustentabilidade Agropecuária na Universidade Católica Dom Bosco - UCDB, mestre em Ciência da Computação pela Universidade Estadual de Maringá - UEM, especialista em Tecnologia Java pela UNIPAN, e graduado em Sistemas de Informação pela UNIPAR. Professor do Instituto Federal de Educação, Ciência e Tecnologia do Mato Grosso do Sul - IFMS (Ponta Porá) – MS/Brasil

** Mestre em Ciência da Computação da Universidade Estadual de Maringá. Possui graduação em Tecnologia em Análise e Desenvolvimento de Sistemas pela Universidade Tecnológica Federal do Paraná. Docente da Universidade Tecnológica Federal do Paraná atuando nos cursos de Engenharia da Computação e Tecnologia em Análise e Desenvolvimento de Sistemas – Paraná/Brasil

às equipes de desenvolvimento e aos clientes um diferencial, uma vez que o mercado impõe exigências ambiciosas, gerando, desse modo, a necessidade de adaptações de processos para melhorar o produto final. Além disso, com o passar do tempo foi possível perceber que empresas se tornaram dependentes dos sistemas de informação, aumentando a exigência das empresas de desenvolvimento de software. Dessa forma, tornou-se evidente o elevado custo de desenvolvimento pela alta complexidade do software, dificultando a manutenção (SOMMERVILLE, 1995) e a estimativa de custo do produto desenvolvido. Por essa razão, são adotadas as metodologias de desenvolvimento de software, as quais estruturam os processos de desenvolvimento de um projeto de software. Tais metodologias foram criadas para modelar e facilitar o entendimento do software pelo cliente e pela própria empresa desenvolvedora (PRESSMAN, 2001), cujo objetivo é aumentar a qualidade, fazer entregas frequentes do produto e, atenderàs expectativas do cliente dentro do prazo firmado. As metodologias mais comuns adotadas nas empresas são: *Extreme Programming (XP)*, *Scrum*, *Feature Driven Development (FDD)*, *Crystal*, *Ratinal Unified Process (RUP)* e *Dynamic Systems Development Method (DSDM)*.

Este trabalho aborda algumas vantagens e contrastes da utilização de metodologias em projetos de software, focando no RUP (metodologia rigorosa) e no XP (metodologia leve). Esses métodos permitem dividir o gerenciamento e o desenvolvimento do software em ciclos para reduzir os custos e os riscos em todas as fases do projeto, até mesmo quando surge a necessidade de alterar os requisitos ou funcionalidades do sistema pelo cliente durante a fase de desenvolvimento. Este estudo faz uma revisão de literatura e analisa ambas metodologias, comparando-as de acordo com suas principais características a fim mostrar ao leitor em quais aspectos uma é melhor que outra e, também, apresentando alguns casos de sucessos dessas metodologias ágeis.

Este artigo está organizado da seguinte forma: inicialmente é apresentada a fundamentação teórica sobre as metodologias ágeis no processo de desenvolvimento de software. Em seguida, são apresentadas algumas características sobre as metodologias RUP e XP, respectivamente. Na sequência, é realizada uma comparação entre ambas as metodologias, apresentando alguns casos de sucesso. E, por fim, apresentadas as considerações finais.

Metodologias Ágeis

As metodologias ágeis têm uma grande importância diante da comunidade de desenvolvimento de software devido ao sucesso gerado em inúmeras companhias ao redor do mundo. Os métodos ágeis se tornaram populares quando surgiram novas abordagens de desenvolvimento na tentativa de adotar processos capazes de se adaptarem às mudanças (BECK, 2000). Segundo Abrahamsson (2002), uma metodologia pode ser considerada ágil quando provê o desenvolvimento de software de forma incremental,

colaborativa, direta e adaptativa. Essas metodologias aplicam uma coleção de práticas, guiadas por princípios e valores que devem ser utilizadas por profissionais no dia-a-dia.

Os métodos tradicionais ou clássicos abordam um conjunto de atividades pré-definidas. Cada etapa (ciclo) do processo possui uma documentação específica e apresenta uma sequência a ser seguida. Normalmente, o trabalho se inicia com um levantamento de requisitos, seguido pelo projeto de alto-nível, implementação, validação e finalizando com a manutenção (SOMMERVILLE, 1995). Uma dificuldade que os modelos clássicos enfrentam, e acreditamos que seja uma das mais graves, é a possível alteração no projeto em seu estágio mais avançado, pois se não for bem especificado em sua fase inicial, ele pode sofrer um grande aumento em seu custo e cronograma devido às alterações de requisitos.

Para documentar e favorecer novos meios no desenvolvimento de software, um grupo de 17 metodologistas formaram a "Aliança do Desenvolvimento Ágil de Software" e definiram princípios nos quais os métodos ágeis deveriam se adaptar. Conforme afirmam BECK et al. (2001), os princípios são: a) entregar o software operacional em várias etapas, sempre priorizando um curto prazo; b) a prioridade é satisfazer o cliente, mediante entregas rápidas e contínuas; c) mudanças de requisitos devem ser sempre bem-vindas, mesmo quando solicitadas em uma fase mais avançada do projeto; d) a equipe de projeto e os desenvolvedores devem trabalhar juntos durante todo o projeto; e) manter os participantes motivados durante o desenvolvimento do projeto. Dar-lhes um ambiente adequado, agradável e, confiar na capacidade de cada um; f) a transmissão da informação para a equipe de desenvolvimento deve ser por meio de conversas pessoais ("cara-a-cara"); g) ter o software funcionando é a principal medida de progresso; h) as pessoas envolvidas no projeto devem ser capazes de manter um ritmo constante; i) simplicidade é essencial; j) as melhores arquiteturas, requisitos e projetos emergem de equipes organizadas; e k) reflexão periódica da equipe sobre como se tornar mais eficaz e, em seguida, ajustar as sugestões conforme a análise.

Uma pesquisa realizada por Ambler (2008) exibiu um resultado referente à adoção de metodologias ágeis nas organizações. A pesquisa realizada com 642 empresas mostrou que o percentual de adoção da metodologia após a realização do teste foi de 69%. A Tabela 1 exibe alguns resultados obtidos pelas empresas mediante o uso temporário de uma metodologia ágil. Conforme pode ser observado, os fatores de produtividade, qualidade e satisfação foram bastante favoráveis ao utilizar uma metodologia. Além disso, após a realização da pesquisa, várias empresas continuaram com o uso no dia a dia.

Tabela 1. Resultados da adoção de uma metodologia ágil

Fator	Melhorou	Não mudou	Piorou
Produtividade	82%	13%	5%
Qualidade	77%	14%	9%
Satisfação	78%	15%	7%
Custo	37%	40%	23%

Fonte: AMBLER (2008)

Em suma, os métodos ágeis procuram fornecer alguns benefícios na sua adoção, tais como: reduzir os custos, obter padronização e organização, melhorar a produtividade, reduzir o tempo de desenvolvimento (ANDERSON, 2003), aumentar a qualidade do software e a satisfação dos clientes (BOEHN e TURNER, 2003). Além disso, elas também podem fornecer respostas rápidas nas mudanças de requisitos de software e, aumentar o contato dos *stakeholders* com o projeto (KARLSTRÖM e RUNESON, 2006) (PIKKARAINEN et al., 2008). As metodologias RUP e XP são abordadas, respectivamente, nos tópicos seguintes.

Rational Unified Process

O *Rational Unified Process* é um *framework* de processo de desenvolvimento de software iterativo. Esse *framework* foi criado pela *Rational Software Corporation* e adquirido posteriormente pela IBM®. Tal metodologia é utilizada na orientação a objetos, unificando os processos com notações UML[1]. Por si só, essa metodologia pode ser considerada um produto de software, pois sua metodologia é sustentada por diversas ferramentas desenvolvidas pela IBM®. O RUP – devido à exigência de utilização de vários artefatos em todas as fases do projeto – é comumente adotado por grandes equipes de desenvolvimento ou que estejam espalhadas geograficamente (KRUCHTEN, 2000). Para viabilizar isso, a metodologia possui 30 papéis distintos, os quais estão agrupados em 9 disciplinas, as quais permitem atribuir tarefas e responsabilidades dentro de uma organização, sendo elas:

• Modelagem de negócio: estuda os problemas atuais que há na organização (cliente). Essa atividade tenta esclarecer a real necessidade do cliente para não ser desenvolvido algo errado ou desnecessário;

• Requisitos de software: traduz as necessidades dos *stakeholders* em artefatos;

• Análise e *design*: especifica a arquitetura dos requisitos de software;

• Implementação: desenvolvimento do sistema;

• Teste: fase de teste do software para verificar se está conforme o esperado. Nessa fase é documento as falhas e os problemas encontrados e fornecido um *feedback* a equipe de desenvolvimento do sistema;

• Implantação: distribui, instala e realiza os testes no cliente. É realizada a importação de dados do sistema antigo (se houver) e realizado treinamentos com os usuários;

• Gerência de implantação e mudança: foco na rastreabilidade de versões do projeto a partir de ferramentas como, o SVN[2] e o CVS[3];

• Gerência de projetos: gerencia os riscos, custos e viabiliza o projeto para entregá-lo dentro do prazo e conforme as expectativas do cliente; e

[1] Sigla de Unified Modeling Language. É uma linguagem para modelagem de software.
[2] *Subversion*, também conhecido como SVN. É um sistema de controle de versão de software. Disponível em: http://subversion.tigris.org/
[3] CVS ou *Concurrent Version System* (Sistema de Versões Concorrentes) é um sistema *open-source* para controlar as versões de um software. Disponível em: http://savannah.nongnu.org/projects/cvs/

• Suporte: fornece o suporte do sistema.

Além disso, o RUP é um *framework* de processos considerado pesado (*heavyweight*), pois exige várias documentações em todas as fases do desenvolvimento de um software. Entretanto, torna-se uma metodologia ágil na medida em que ela é adaptada para outras realidades, customizando ou removendo as técnicas que a equipe não utiliza, diminuindo a formalização e minimizando a documentação (IBM, 2005). Ele determina seu ciclo de vida em quatro fases que constituem o ciclo de vida da construção de uma versão do software, sendo elas: (i) início ou concepção - é determinado o escopo da versão do projeto e como será o produto final; (ii) elaboração - o projeto começa a se tornar real, pois é realizada uma análise do problema e criada a arquitetura do projeto; (iii) construção - implementação do sistema e; (iv) transição - entrega do produto ao cliente. Embora Brooks (1987) afirme que é impossível especificar um software totalmente antes de sua implementação, o RUP sugere que a fase de concepção (i) seja realizada dessa maneira para ter uma visão geral da versão do software.

Na fase inicial ou concepção, alguns pontos importantes são avaliados, tais como: viabilidade, avaliação de risco, elaboração de cronograma e diagrama de casos de uso. A fase de elaboração é bem crítica, pois o desenvolvimento realizado nessa etapa será considerado nas etapas seguintes. Na fase de construção, dependendo do tamanho do projeto é recomendado que ele seja dividido em partes, uma vez que pequenos problemas podem ser resolvidos com soluções mais simples. Dessa forma, haverá maior flexibilidade no gerenciamento das iterações e artefatos. E, por fim, a fase de transição inclui treinamentos com os usuários, manutenção do sistema, validação, testes finais com o cliente visando receber o *feedback* do usuário final.

Extreme Programming

A *Extreme Programming* é uma metodologia leve (*lightweight*), eficiente, flexível, previsível, científica, possui um baixo risco (BECK, 2000) e é direcionada para equipes pequenas e médias. Ela permite o desenvolvimento de softwares a partir de requisitos vagos e que mudam frequentemente. Um dos objetivos do XP é diminuir o custo com as possíveis mudanças no software, sendo que o código realizado em cada etapa é um indicador de progresso do projeto. Um ponto bastante comentado da especificação do XP é não considerar documentação, processos, ferramentas, planejamento, mas sim indicar um valor secundário que esse conjunto possui diante das iterações, tais como: o bom funcionamento do software e a iteração com o cliente. A principal preocupação do XP é com a entrega rápida do software e a satisfação do cliente, tendo como premissas cumprir seus custos e prazos. Para isso, o XP possui alguns papéis fundamentais (BECK, 2000):

• Programador: desenvolvedor do sistema (escreve os códigos-fonte);

• Instrutor: responsável pelo processo como um todo, realizando treinamentos

e instruindo os membros da equipe. Esse papel é exercido por uma pessoa com ampla experiência e conhecimento na metodologia;

• Fiscal: responsável por fornecer o *feedback* dos processos, realizando estimativas e avaliando as metas para alcançar os objetivos conforme os recursos e o tempo especificado;

• Cliente: responsável por descrever as prioridades no projeto e suas user stories. O Cliente também faz parte da equipe, com o objetivo de sanar dúvidas em qualquer fase do projeto;

• Testador: ajuda o cliente a realizar testes funcionais regularmente;

• Consultor: membro externo que auxilia a equipe com assuntos técnicos; e

• Gerente de projetos: pessoa responsável pelo projeto, estando sempre em comunicação com a equipe. Esse papel tem por função resolver os problemas e tomar as decisões do projeto.

O XP se baseia em cinco valores (BECK, 2000) (BECK, 2004) (WELLS, 2009): (i) comunicação - para que um projeto tenha sucesso e qualidade é necessária muita comunicação "cara-a-cara" entre o provedor de serviço e o cliente; (ii) *feedback* - as decisões devem ser ágeis e decisivas, sendo que todos os integrantes do projeto devem estar cientes do que está acontecendo; (iii) coragem - otimizações de códigos devem ser feitas sem criar novos *bugs* no sistema; (iv) simplicidade - redução do código-fonte com funções e procedimentos desnecessários. Trabalhar com simplicidade acelera o processo de produção dos requisitos do projeto, pois o que os clientes precisam, geralmente, é mais simples do que o analista descreve e; (v) respeito – entregar o sistema para o cliente no prazo determinado e conforme o solicitado.

Nessa metodologia, as iterações devem ser curtas entre os participantes, fornecendo constantes atualizações no sistema para o cliente, lançando, assim, várias versões (*releases*) frequentemente. O objetivo das iterações rápidas é acelerar o processo de *feedback* e *tomar* como referência os comentários da versão atual para a próxima etapa. Por esse motivo, o XP requer que os profissionais tenham um perfil mais maduro e experiente, além de precisar de profissionais comunicativos e com facilidade de relacionamento, pois os analistas e programadores fazem contatos bem frequentes com os clientes.

Semelhanças e Contrastes entre as Metodologias RUP e XP

Comparar ambas as metodologias necessitou de um estudo empírico diante da literatura atual sobre desenvolvimento de software ágil. Embora essas metodologias possuam várias diferenças, acreditamos que a mais relevante é referente ao seu direcionamento, pois o RUP é orientado a processos e o XP a pessoas. Outra diferença significativa que muitas vezes torna-se motivo de escolha entre metodologias é que o RUP conta com softwares comerciais pagos mantido por uma empresa e o XP é totalmente gratuito mantido por uma comunidade de voluntários.

Assim, projetos de baixo custo podem usufruir dos benefícios do XP, caso contrário seria inviabilizado pelo preço das licenças de software. Por outro lado, semelhanças importantes foram detectadas, tais como: são orientadas ao cliente, possuem seu funcionamento baseado em iterações e utilizam papéis. Essa seção apresenta uma análise comparativa (divididas em 6 categorias) a partir das principais características de ambas as metodologias.

Comunicação

Uma das principais diferenças entre essas metodologias é referente a comunicação entre os membros da equipe, pois, enquanto que o RUP é baseado totalmente em artefatos, utilizando-os para gerenciar as etapas de um projeto e possibilitando que as comunicações sejam rastreadas, o XP possui uma característica oposta, uma vez que toda sua comunicação é baseada por conversas diretas ("cara-a-cara"). O RUP recomenda que seja gasto mais tempo com a diagramação e a modelagem visual, enquanto que o XP recomenda pouco tempo nessas atividades e com um nível menor de formalismo. Dessa forma, ao analisar o formato da comunicação entre as metodologias, é possível observar que o RUP tem todas as fases preparadas para trabalhar com equipes presenciais e, principalmente, distribuídas. No XP, por sua vez, é preferível que toda comunicação seja realizada presencialmente, o que dificulta o desenvolvimento de software com equipes geograficamente distribuídas. Além disso, no XP há pouca utilização de artefatos, pois o foco dessa metodologia está em suas *user stories* (descrições textuais a respeito das características e funcionalidades do sistema), técnica mais simples que a utilizada pelo RUP.

Na tentativa de suprir a pouca documentação de um projeto, o XP orienta que os seus desenvolvedores fiquem próximos uns dos outros, pois na medida em que precisarem, terão acesso rápido e direto aos membros mais experientes (MARTIN, 2003) e, até mesmo, com o próprio cliente. Logo, isso dispensaria o trabalho de buscar nos documentos as respostas de possíveis dúvidas. Embora essa estratégia acelere o desenvolvimento de software, essa característica pode aumentar o risco de um projeto, pois o conhecimento fica armazenado apenas nos códigos-fontes e nas pessoas da equipe (propriedade intelectual) – analistas e desenvolvedores – e, dessa forma, a experiência da equipe pode diminuir na medida em que a rotatividade das pessoas experientes se torne frequente.

Softwares de Apoio

O RUP especifica e recomenda várias ferramentas (pagas) desenvolvidas pela própria IBM® que podem ser utilizadas em um projeto. Essas ferramentas podem

ser encontradas pelo pacote de software chamado "*IBM Rational Rose Modeler*". Ela foi designada especialmente para analistas e arquitetos de software. Essa é uma opção para maximizar as habilidades de design e arquitetura de software, pois é um ambiente em que o usuário pode obter resultado de modelagem mais rápido que o convencional e, com suporte baseado em padrões de projetos com possibilidade de reuso de componentes. O XP, por sua vez, não especifica nenhuma ferramenta para ser utilizado no desenvolvimento do projeto, deixando livre para a equipe decidir o que achar mais viável. Nesse caso, é possível utilizar ferramentas livres como, o *Ant*[4], o *SVN* e o *JUnit*[5].

Papéis e Responsabilidades (Equipe)

Ao colocarmos lado a lado as atividades e os papéis de uma equipe, pôde ser observado que o RUP fornece uma divisão de tarefas específica, enquanto que no XP, as tarefas e os papéis têm um caráter mais abrangente, ou seja, possui características genéricas e sem responsabilidade específica para cada atividade. Além disso, muitas das atividades do RUP poderiam ser beneficiadas se houvesse a presença do cliente como um papel efetivo na equipe, como acontece no XP, pois em vez do analista/desenvolvedor buscar as informações na documentação, ele poderia consultar diretamente o cliente. Assim, vários documentos e artefatos poderiam ser reduzidos do processo e, consequentemente, aumentaria a eficiência.

Tratamento do Código-Fonte

Para tratar o código-fonte o RUP divide um problema grande em vários subproblemas. Dessa forma, a ideia é dividir um módulo do sistema em vários submódulos. Assim, cada submódulo terá um coordenador que fará o acompanhamento para que no final do desenvolvimento tudo se integre da melhor forma possível. Essa estratégia é interessante, visto que problemas menores possuem soluções mais simplificadas. O XP, por sua vez, realiza esse tratamento de modo coletivo, possibilitando que qualquer programador modifique o código-fonte na medida em que os "*bugs*" são encontrados ou, visualizarem uma maneira de otimizar o código.

Certificação

Para certificar o conhecimento dos profissionais perante a metodologia existem

[4] É uma ferramenta utilizada para automatizar o processo de construção do software. Disponível em: <http://ant.apache.org/>.
[5] É um *framework* que auxilia na criação e execução de testes unitários de classes Java. Disponível em: <http://www.junit.org/>.

as certificações, que tem por finalidade atestar o produto, o serviço, o sistema e o próprio membro da equipe. Nessa última, é possível verificar se o profissional está em conformidade com as normas, diretrizes, requisitos e regulamentos da metodologia. Para isso, apenas o RUP disponibiliza um meio de validar as habilidades do profissional diante da metodologia. Existem várias provas que podem certificar o conhecimento na metodologia e na arquitetura RUP, entretanto, a prova *Rational Unified Process v7.0 é uma das mais indicadas para quem utiliza a metodologia. Essa é a certificação base para quem utiliza o RUP e é ideal para os profissionais que desejam* ter um diferencial nessa carreira. Ao passar na prova de certificação, o profissional se diferencia dos demais que trabalham com a metodologia e recebe o título de "*IBM Certified Solution Designer – Rational Unified Process v7.0*".

Esforço e Tempo

A alocação de tempo das pessoas da equipe em ambas as metodologias diferem, uma vez que o RUP requer diversas documentações e artefatos e o XP após suas *user stories,* foca apenas no desenvolvimento do produto.

Por fim, é criado a Tabela 2 que exibe uma comparação de algumas características usadas em ambas as metodologias. Embora a Tabela 2 mostre que o XP não seja adequado para equipes de grande porte, essa metodologia pode ser adaptada e utilizada de forma satisfatória. A mesma situação ocorre com a metodologia RUP para equipes pequenas. Essas são metodologias customizáveis e, assim, elas podem ser modeladas para qualquer tamanho de equipe e projeto. As metodologias RUP e XP não obrigam que a organização siga estritamente suas regras, podendo, em certas situações, adaptar e adequar mediante as necessidades da empresa. De acordo com a comparação da Tabela 2, podemos identificar que ambas as abordagens possuem características positivas e negativas. Enquanto que o RUP tem seu foco voltado a projetos complexos com equipes grandes e distribuídas, e um alto controle com vários artefatos em sua documentação, o XP é direcionado para projetos de pequeno e médio porte, em virtude das suas características, agilidade e pouca especificação documental. Assim, a documentação e a preocupação formal do XP são descartadas e subentendidas como desnecessárias com o objetivo de fornecer mais atenção ao cliente. Por outro lado, o RUP é considerado automaticamente um processo de desenvolvimento iterativo, entretanto, é possível que a equipe utilize essa metodologia e faça uma iteração completa de modelagem sem que o programador codifique uma linha de código, deixando, assim, de ser totalmente iterativa.

Tabela 2. Comparação entre o RUP e o XP

Característica da metodologia	RUP	XP
Possui prova de certificação	X	
Possui ferramentas pré-definidas para a utilização da equipe	X	
Possui desenvolvimento iterativo e incremental	X	X
Visa a qualidade do código-fonte		X
Faz a integração contínua dos componentes		X
É baseada em artefatos	X	
É baseada na comunicação com o cliente		X
Direcionada à arquitetura e modelagem do software	X	
Possui o controle sobre os riscos e os custos do projeto	X	X
Possui o gerenciamento de mudanças do projeto	X	
Utiliza vários diagramas UML para a documentação do software	X	
Utiliza os principais diagramas UML para documentar o software	X	X
O cliente faz parte da equipe de software		X
O desenvolvedor possui contato com o cliente		X
Indicado para projetos complexos	X	
Indicado para equipes grandes e distribuídas geograficamente	X	

Em várias situações é difícil escolher a metodologia ideal para utilizar em um projeto ou organização, pois algumas delas como, o RUP, possuem formalidades excessivas de artefatos em cada fase do projeto (LARMAN, 2004). Ter formalidade é bom para a equipe na tentativa de reduzir falhas, porém é preciso saber quando utilizá-las – sendo breve e objetivo – para não tornar os processos burocráticos demais e dependentes de artefatos desnecessários (MARTIN, 2003).

Independente se a metodologia escolhida para o desenvolvimento de software possui muitos critérios, a adoção de uma metodologia visa aumentar a qualidade do software, reduzindo os possíveis riscos do projeto e gerenciando melhor as iterações ocorrentes dentro do escopo do desenvolvimento do produto. Então, qual metodologia deve ser escolhida para ser utilizada em um projeto de pequeno e grande porte? Na opinião dos autores deste artigo, antes de realizar qualquer decisão que irá afetar a equipe inteira do início ao término do projeto é necessário explorar cada metodologia profundamente e verificar qual a metodologia em que o projeto melhor se encaixa, uma vez que existem vários fatores para analisar como, a natureza do projeto, equipe, organização, cultura (COCKBURN, 2002), etc. Além disso, se o gerente de projetos estiver procurando uma metodologia que seja compatível com modelos que atendem padrões de qualidade como, o *Capability Maturity Model* (CMM), estará no rumo certo com ambas as metodologias, pois com algumas adaptações o RUP reúne quase todos os requisitos para alcançar o CMM nível 2 e 3 (MANZONI et al., 2003) e o XP também é compatível (PAULK, 2001).

Estudos de Caso de Sucesso

Nessa seção são apresentados alguns estudos de caso pressentes na literatura. No primeiro, Bezerra et al. (2012) analisam se a metodologia XP pode ser utilizada de maneira eficiente e eficaz nos processos gerenciais de uma empresa. Para tanto,

os autores realizaram um estudo em uma empresa de tecnologia que empregava em todos os seus setores a metodologia XP. Foi observada que essa metodologia pode ser aplicada em empresas, sobretudo, do ramo de tecnologia por possuir características mais liberais, criativas e menos tradicionais. Entre outros benefícios, percebeu-se que, por ser um método simples e econômico, o resultado foi em melhorias no clima organizacional, relacionamento interpessoal, *feedback*, comunicação e na disseminação do conhecimento dentro da empresa.

Em Dias et al. (2009), foi analisada a possibilidade de utilizar e adotar a metodologia XP no desenvolvimento de software livre por comunidades de usuários distribuídas geograficamente. Como é normal o desenvolvimento de software por equipes distribuídas e frequente o surgimento de comunidades de software livre trabalhando de forma distribuída é necessário que o produto final desenvolvido coletivamente seja de qualidade. Assim, o objetivo desse trabalho é fornecer informações para essas comunidades a fim tornar o software mais maturo, absorvendo as vantagens que o XP pode oferecer. Nesse trabalho, os autores avaliam e mapeiam o perfil desses desenvolvedores em relação a sua experiência prática e opinião sobre a possibilidade de adoção total ou parcial do XP para guiar o desenvolvimento dos projetos de software livre. Trabalhando ainda com desenvolvimento distribuído de software, Dos Santos et al. (2010) apresentam os resultados da experiência de uma fábrica de Software baseada no processo de desenvolvimento distribuído de software embasada em uma metodologia ágil.

Em um dos raros trabalhos dedicados à análise de metodologias ágeis em instituições públicas brasileiras, Melo et al. (2010) realizam um estudo de caso em uma instituição pública de grande porte, o Banco Central do Brasil. Esse trabalho teve por objetivo analisar empiricamente a adoção de métodos ágeis e seu impacto no aprendizado, qualidade do código-fonte, produtividade e satisfação do cliente. Além disso, os autores detalham o contexto organizacional que motivou e apoiou o processo de adoção, descreve dois projetos pilotos e discute os resultados observados, sob perspectivas técnicas e gerenciais de um projeto de implantação de 18 meses. Os resultados obtidos foram satisfatórios e decisivos para encorajar novas experiências com métodos ágeis dentro da organização (MELO et al., 2010).

Treccani e Souza (2010) apresentaram resultados de um estudo empírico com o objetivo de saber como são aplicadas as metodologias ágeis no processo de desenvolvimento e manutenção de sistemas, em especial na atividade de refatoração. WU et al. (2010) apresentam um *framework* para uso em projetos que foi resultado de um estudo de ambas as metodologias, RUP e XP. Shen et al. (2012) faz uma revisão sistemática sobre a aplicação de métodos ágeis no desenvolvimento de software embarcado. Por fim, Dybå et al. (2008) apresentam uma revisão sistemática sobre as metodologias de software ágil e apontam que diversos trabalhos sobre métodos ágeis existem na literatura. Contudo, identificam que pouco se sabe sobre como esses métodos são realizados na prática e quais efeitos geram, deixando em aberto possíveis trabalhos futuros nessa área.

Conclusão

Neste trabalho apresentamos uma comparação entre duas metodologias de desenvolvimento de software, mostrando seus pontos positivos e negativos e, ao final, é apresentado alguns estudos de caso existentes na literatura. De forma geral, o RUP é uma boa alternativa para projetos complexos e/ou que possuem equipes geograficamente distribuídas, pois o formato de comunicação da metodologia favorece as iterações entre as equipes em cada ciclo do projeto. No RUP, as diretrizes referentes a especificação do tamanho do projeto não estão totalmente claras. Todavia, é possível afirmar que essa metodologia faz uma divisão muito bem definida das atividades que cada membro da equipe deve realizar, especificando os artefatos que devem ser usados para construir o produto final. Além disso, ele possui inúmeros artefatos produzidos durante cada fase do projeto que, em algumas delas, podem ser consideradas irrelevantes devido ao tamanho do projeto. Logo, utilizar essa metodologia por completo não torna a metodologia ágil, pois fornece uma documentação detalhada para cada fase do projeto e exige bastante dos profissionais por abranger o processo de produção de software como um todo. Dessa forma, para o RUP se tornar um processo ágil é necessário que seja adotado apenas alguns dos artefatos disponibilizados pela metodologia. Entretanto, uma pergunta que fica em aberto é: se o RUP for adotado e, processos, documentos e artefatos forem customizados ou removidos, o resultado ainda será RUP?

De fato, o XP apresenta uma metodologia bem menos burocrática. Essa é uma boa alternativa de adoção para pequenos e médios projetos quando a ênfase não é descrever documentação em cada etapa do software, mas sim fundamentar-se diretamente com o cliente por meio da comunicação oral. Além disso, essa metodologia é recomendada para novos projetos onde o cliente não sabe exatamente o que quer, sendo que, nesse caso, o cliente pode mudar de opinião frequentemente durante o desenvolvimento do projeto. Assim, com o constante *feedback* recebido dos *stakeholders*, será possível adaptar as eventuais solicitações de mudanças do sistema. Entretanto, mesmo o foco do XP sendo para projetos de pequeno e médio porte, ele pode se adequar a projetos complexos, grandes e distribuídos. Contudo, caso isso ocorra, é preciso cuidar ao atribuir papéis e responsabilidades aos membros da equipe, pois eles não são totalmente específicos e detalhados.

Por fim, muitas vezes o fato das equipes adotarem apenas alguns artefatos e aplicarem a metodologia de forma diferente não significa que está errado. Para isso, é importante conhecer o que há de melhor entre as metodologias a fim de que a adoção de uma delas atenda as expectativas do projeto.

Referências

ABRAHAMSSON, P. et al. *Agile Software Development Methods:* Review and Analysis. Espoo: VTT Publications. Technical Report, 2002.

AMBLER, S. W. Has *Agile Peaked?* Scott Crunches the Numbers to Find Out, 2008. Disponível em: <http://www.ddj.com/architect/207600615> Acesso em: abr. 2013.

ANDERSON, D. J. *Agile Management for Software Engineering, Applying the Theory and Constraints for Business Results*. Upper Saddle River, NJ: Prentice Hall, 2003.

BECK, K. *Extreme Programming Explained*. Boston, Massachusetts: Addison-Wesley Longman, 2000.

BECK, K. *Extreme Programming Explained:* Embrace Change. 2. ed. Reading, Massachusetts: Addison-Wesley, 2004.

BECK, K. et al. *Manifesto for Agile Software Development*. 2001. Disponível em: <http://agilemanifesto.org/> Acesso em: abr. 2013.

BEZERRA, W. et al. Utilização da Metodologia Ágil Extreme Programming (XP) como Ferramenta de Gestão: Um Estudo de Caso numa Empresa do Ramo de Tecnologia e Serviços. *Revista Connexio*, p. 41-56, 2012.

BOEHM, B. et al. *Management Challenges to Implement Agile Processes in Traditional Development Organizations*. IEEE Software, 2005.

BROOKS, F. No Silver Bullet: Essence and Accidents of Software Engineering. Proc. IFIP, IEEE CS Press, pp. 1069-1076; reprinted in *IEEE Computer*, p. 10-19, Apr. 1987.

COCKBURN, A. *Agile Software Development*. MA: Addison Wesley Longman Inc., 2002.

DIAS, T. M. R. et al. Adoção da Metodologia Extreme Programming para Construção de Sofware. In: SIMPÓSIO BRASILEIRO DE COMPONENTES, ARQUITETURAS E REUTILIZAÇÃO DE SOFTWARE (SBCARS), 2009. p. 10-23.

DOS SANTOS, A. C. C. et al. Experiência Acadêmica de uma Fábrica de Software utilizando Scrum no Desenvolvimento de Software. In: WORKSHOP BRASILEIRO DE MÉTODOS ÁGEIS (WBMA), 2010, Porto Alegre. p. 86-98.

DYBÅ, T. et al. Empirical studies of agile software development: A systematic review. *Information and Software Technology*, v. 50, n. 9-10, p. 833–859, 2008.

IBM. *Using RUP to Manage Small Projects and Teams*, 2005. Disponível em: <http://www.ibm.com/developerworks/rational/library/jul05/kohrell/index.html> Acesso em: abr. 2013.

KARLSTRÖM, D. et al. Integrating Agile Software Development into Stage-Gate Managed Product Development. *Empirical Software Engineering*, v. 11, n. 2, p.203–225, 2006.

KRUCHTEN, P. *The Rational Unified Process:* An Introdution. Massachusetts: Addison Wesley, 2000.

LARMAN, C. *Applying UML and Patterns:* An Introduction to Object-Oriented Analysis and Design and Iterative Development. 3 ed. NY: Prentice Hall, 2004.

MANZONI, L. V. et al. Identifying extensions required by RUP (Rational Unified Process) to comply with CMM (Capability Maturity Model) levels 2 and 3. *IEEE Transactions on Software Engineering*, v. 29 , n. 2, p. 181-92, Feb. 2003.

MARTIN, R. *Agile Software Development:* Principles, Patterns, and Practices. NY:

Pearson Education, 2003.

MELO, C. O. et al. Adoção de métodos ágeis em uma instituição pública de grande porte - um estudo de caso. In: WORKSHOP BRASILEIRO DE MÉTODOS ÁGEIS (WBMA), 2010, Porto Alegre. v. 24, p. 112-125.

PAULK, N. C. Extreme programming from a CMM perspective. *IEEE Software*, v. 18, n. 6, p. 19-26, Nov/Dec. 2001.

PIKKARAINEN, M. et al. The Impact of Agile Practices on Communication in Software Development. *Empirical Software Engineering*, 2008.

PRESSMAN, R. S. *Software Engineering:* A Practitioner's Approach. 5 ed. New York: McGraw-Hill, 2001.

SHEN, M. et al. Applying agile methods to embedded software development: A systematic review. In: SOFTWARE ENGINEERING FOR EMBEDDED SYSTEMS (SEES), 2., INTERNATIONAL WORKSHOP ON IEEE, 2012.

SOMMERVILLE, I. *Software Engineering*. 5ª ed. NY: Addison-Wesley, 1995.

TRECCANI, P. J. F. et al. Utilização de Metodologias Ágeis no Desenvolvimento de Software: Resultados de um Estudo Empírico. In: EXPERIMENTAL SOFTWARE ENGINEERING LATIN AMERICAN WORKSHOP (ESELAW), 7., 2010. *Proceedings*… p. 50-59.

WELLS, D. *Extreme Programming*. A Gentle Introduction, 2009. Disponível em: <http://www.extremeprogramming.org> Acesso em: abr. 2013.

Wu, X. et al. The Research on Necessity and Plan for Using Extreme Programming in Rational Unified Process. In: COMPUTATIONAL INTELLIGENCE AND SOFTWARE ENGINEERING (CISE), 2010 INTERNATIONAL CONFERENCE ON IEEE, 2010.

Artigo recebido em: 21 maio 2013
Aceito para publicação em: 16 dez. 2013