ALINHAMENTO DO RUP 7.0 AOS VALORES DO MOVIMENTO ÁGIL

RUP 7.0 ALIGNMENT WITH THE AGILE MANIFESTO VALUES

Sergio Akio Tanaka
Audare Engenharia de Software, E-mail: sergio.tanaka@audare.com.br

André Banki
AltoQi Tecnologia, E-mail: andre.banki@gmail.com

Resumo: O desenvolvimento ágil tem sido um assunto de interesse crescente na comunidade de desenvolvimento de software nos últimos anos. Ao comparar as metodologias ágeis, como o Extreme Programming (XP) com as prescritivas, muitos autores listam o Rational Unified Process (RUP) com estas últimas. Esses trabalhos atualmente disponíveis retratam a versão 2003 do RUP, com a abordagem já considerada "clássica" das Melhores Práticas. Este artigo mostra que o RUP é um processo em constante evolução e, nesta versão 7.0, englobou nos seus "Conceitos Chave" a maior parte dos valores defendidos pelo Manifesto Ágil perante a comunidade de desenvolvimento de software.

Palavras-chave: Engenharia de software; Movimento Ágil; Rational Unified Process

Abstratc. *Agile software development has been under increasing attention of the software development community in the recent years. Several authors list the Rational Unified Process (RUP) amongst the prescriptive methodologies when comparing them with the so-called "agile" methodologies, like Extreme Programming (XP). These currently available papers talk about RUP in its version 2003, using the classic approach of the Best Practices. This paper shows that RUP is a process under constant evolution and, in its version 7.0, incorporates in its "Key Concepts" most of the values professed by the Agile Manifesto to the software development community.*

Keywords**:** *Software engineering; Agile Manifesto; Rational Unified Proces*

1 INTRODUÇÃO

"É provável que a mudança mais relevante na forma de pensar o processo de desenvolvimento de software, nestes últimos anos, tenha sido o aparecimento do termo 'ágil'". (FOWLER, 2005). A prática do desenvolvimento iterativo, em substituição aos tradicionais modelos em "cascata", já pode ser considerada um consenso entre os autores modernos. Ainda assim, tem-se observado um debate que coloca de um lado as metodologias ditas "ágeis", como XP e Scrum, e de outro os processos ditos prescritivos. Dentre estes últimos, diversos autores destacam o RUP.

Segundo Runeson e Greberg (2004), nesse "boom" das metodologias ágeis, várias metodologias de *software* bem estabelecidas têm tentado também se apresentar como ágeis. O RUP está entre elas, tendo incluído já há alguns anos uma extensão baseada no XP para seu processo (POLLICE, 2001). Apesar disso, em geral, o RUP não é listado junto às demais metodologias ágeis. Este artigo procura abordar conceitualmente essa questão, verificando até que ponto os valores defendidos pelo movimento ágil diferem, ou não, da proposta do RUP.

2 CONTEXTUALIZAÇÃO

O desenvolvimento de software é uma atividade complexa. Essa complexidade corresponde à sobreposição das complexidades relativas aos seus diversos componentes: software, hardware, procedimentos, entre outros. Tentativas de lidar com essa complexidade envolvem a definição de um processo de desenvolvimento de software, o qual compreende todas as atividades necessárias para definir, desenvolver, testar e manter um produto de software. Alguns objetivos de um processo de desenvolvimento são: definir quais as atividades a serem executadas ao longo do projeto; quando, como e por quem tais atividades serão executadas; prover pontos de controle para verificar o andamento do desenvolvimento; e padronizar a forma de desenvolver software em uma organização (BEZERRA, 2007).

Não há processo correto ou incorreto; dependendo da sua aplicação, ambiente e objetivo, o uso de um processo específico pode ser vantajoso ou não. Um ponto importante a ressaltar é que cada autor e organização colocam e classificam processos e atividades de forma diferente, tornando difícil uma uniformidade completa (BONA, 2002). Ao avaliar uma metodologia exclusivamente sob o ponto de vista de outra, pode-se criticar um ou outro aspecto, mas, de forma geral, o objetivo de todos é o mesmo: produzir *software* de qualidade.

2.1 Rational Unified Process (clássico)

O Rational Unified Process® (também chamado de processo RUP®) é um processo de engenharia de software criado por Ivar Jacobson, Grady Booch e Jim Rumbaugh em 1998. O RUP evoluiu ao longo dos anos, em conjunto com a Unified Modeling Language (UML). Embora existam livros publicados sobre essa metodologia, o RUP em si é um produto comercial, desenvolvido pela Rational Software, hoje uma subsidiária da IBM.

O RUP oferece uma abordagem baseada em disciplinas para atribuir tarefas e responsabilidades dentro de uma organização de desenvolvimento. O ciclo de vida de *software* do RUP é dividido em quatro fases seqüenciais, cada uma concluída por um marco principal, ou seja, cada fase é basicamente um intervalo de tempo entre dois marcos principais. Cada fase pode ser dividida em um número planejado de iterações.

Além de ser um processo, o RUP é tratado como um produto, associado ao Rational Method Composer (RMC), o qual é desenvolvido e mantido pela IBM Rational. Assim como qualquer outra ferramenta de *software*, o RUP está em constante evolução. A primeira versão com o nome “Rational Unified Process” foi a versão 5.0, de 1998. A ela seguiram-se as versões 5.5 (1999), 2000 (2000), 2003 (2003) e 7.0 (2006). No momento da elaboração deste artigo, a versão corrente é a 7.0.1.

Praticamente toda a bibliografia disponível sobre o RUP, incluindo os trabalhos que o comparam com outras metodologias, utiliza os princípios presentes em sua versão 2003. Nessa abordagem “clássica”, as 4 fases são Iniciação, Elaboração, Construção e Transição. Um conceito muito importante dessa versão é o fato de que o RUP “mostra como é possível aplicar as melhores práticas de engenharia de *software* e usar ferramentas para automatizar o processo de engenharia de *software*”. Segundo Kruchten (2000), essas não são as “melhores práticas” por que é possível quantificar exatamente o seu valor, mas sim por que são comumente utilizadas por empresas de sucesso.

Pode-se dizer que o conceito das Melhores Práticas foi uma das principais formas de descrever a filosofia do RUP, até sua versão 2003, englobando:

a) Desenvolva Iterativamente;
b) Gerencie Requisitos;
c) Use Arquiteturas de Componentes;
d) Modele Visualmente (UML);
e) Verifique Qualidade Continuamente;
f) Gerencie Mudanças.

Na versão 7.0, as melhores práticas foram substituídas pelos "Princípios Chave para o Desenvolvimento Orientado a Negócios". Este ponto será tratado mais à frente.

2.2 Manifesto Ágil

Em 2001, movidos pela observação de que equipes de desenvolvimento de *software* nas mais diversas organizações estavam presas por processos cada vez mais burocráticos, um grupo de profissionais renomados reuniu-se para delinear os valores e princípios que permitiriam às equipes de desenvolvimento produzir rapidamente e responder às mudanças. Eles chamaram a si mesmos de Aliança Ágil. Trabalharam por dois dias para criar um conjunto de valores. O resultado foi o Manifesto da Aliança Ágil. (MARTIN, 2002).

Segundo o Manifesto Ágil, devem ser valorizados:

a) "indivíduos e interações mais do que processos e ferramentas";
b) "software funcional mais do que uma documentação completa";
c) "a colaboração do cliente mais do que a negociação de um contrato";
d) "responder às mudanças mais do que seguir um plano".

É importante salientar que o movimento ágil não é contrário à definição de um processo. Segundo Fowler e Highsmith (2001), os desenvolvedores ágeis adotam a modelagem, mas não apenas para arquivar diagramas em gavetas empoeiradas; adotam a documentação, mas não para produzir pilhas de papéis desatualizados e nunca consultados; e adotam o planejamento, mas reconhecendo os seus limites em um ambiente turbulento.

A publicação do Manifesto disparou um movimento na indústria de desenvolvimento de *software*, autodenominado "desenvolvimento ágil". Esses valores, por demonstrarem claramente os pontos mais importantes na visão de um grande segmento do setor, podem ser considerados uma referência para todas as demais metodologias.

2.3 Metodologias ágeis

Segundo Abrahamsson (2002), uma metodologia pode ser dita ágil quando efetua o desenvolvimento de software de forma incremental (liberação de pequenas versões, em iterações de curta duração), colaborativa (cliente e desenvolvedores trabalhando juntos, em constante comunicação), direta (o método em si é simples de aprender e modificar) e adaptativa (capaz de responder às mudanças até o último instante). Nesse conceito, inclui como metodologias ágeis *Extreme Programming* (XP), Scrum, *Crystal*, *Feature Driven*

Development (FDD), *Dynamic Systems Development Method* (DSDM), Open *Source Software Development* e, com certa ressalva, o próprio RUP.

Dentre as diversas metodologias ágeis, certamente a que mais se destaca no mercado é o XP. Desde o ano de 2000 é realizada anualmente a *Extreme Programming Conference*. Segundo Beck et al (2001), XP é uma metodologia leve, eficiente, flexível e de baixo risco para times pequenos e médios, que desenvolvem software com requisitos dinâmicos ou em constante mudança.

O XP segue um conjunto de valores, princípios e regras básicas que visam alcançar eficiência e efetividade no processo de desenvolvimento de software. Os valores são cinco: comunicação, simplicidade, *feedback*, coragem e respeito. Nestes valores, estão fundamentados alguns princípios básicos: *feedback* rápido, simplicidade assumida, mudanças incrementais, compreensão às mudanças e qualidade do trabalho (BECK, 2004)

3 PRINCÍPIOS CHAVE DO RUP 7.0

A maior parte dos trabalhos sobre o RUP, atualmente disponíveis no meio técnico, baseia-se em sua visão clássica, referente à versão 2003, com as 4 fases e 9 disciplinas. Os valores desse processo são normalmente conceituados com base nas "melhores práticas" para desenvolvimento de software. Por exemplo, os trabalhos de Booch, Martin e Newkirk (1998), Kruchten (2001), Menjoge (2003), Nonemacher (2003), Runeson (2004) e Martins e Silva (2004), dentre vários outros, comparam o RUP e outras metodologias, tomando por base a visão clássica do processo.

Todavia, com o lançamento de sua versão 7.0, o RUP apresenta-se como um processo ainda mais genérico e capaz de suportar o desenvolvimento ágil. Uma alteração fundamental é o foco na Arquitetura Orientada a Serviços (SOA), em cujo processo de entrega o ciclo de vida tradicional é substituído pelas novas fases "Análise de Transformação de Negócio", "Identificação", "Realização" e "Especificação". O trabalho de Arsanjani, Johnston e Smith (2005) é um ponto de partida para o entendimento dos conceitos do SOA associados ao RUP. O presente artigo não aborda esse assunto, atendo-se ao segundo ponto mais relevante nessa revisão do RUP: a atualização das "Melhores Práticas", relançadas como "Princípios Chave para Desenvolvimento Orientado para Negócios." (ARSANJANI, JOHNSTON, SMITH, 2005).

Conforme colocado no RUP 7.0 (IBM, 2006),

> as seis boas práticas testadas e aprovadas do Rational Unified Process foram a base para a evolução de ferramentas e processos da Rational por mais de uma década. Hoje em dia, como o desenvolvimento de software está se tornando uma capacidade de negócios principal, nossas boas práticas são o desenvolvimento dentro do contexto maior de desenvolvimento orientado a negócios. Os princípios a seguir rearticulam as boas práticas para o ciclo de vida mais amplo de sistemas de evolução de forma contínua, nos quais o elemento de evolução principal é o software.

Sucintamente, são definidos os seguintes princípios chave:

a) Adaptar o processo: lembra que é crítico dimensionar corretamente o processo de desenvolvimento para as necessidades do projeto, detalhando a quantidade de cerimônia, precisão e controle de acordo com uma variedade de fatores, incluindo o tamanho e a distribuição de equipes, a quantidade de restrições externamente impostas e a fase na qual o projeto se encontra;

b) Equilibrar as prioridades dos investidores: destaca a importância de equilibrar as necessidades frequentemente conflitantes dos investidores envolvidos, gerenciando os requisitos de forma efetiva, ao compreender e priorizar as necessidades de negócios e dos investidores;
c) Colaborar através de equipes: ressalta a importância de fomentar uma comunicação eficiente dentro do projeto, através de uma adequada organização das equipes, da criação de ambientes de colaboração e da motivação dos indivíduos;
d) Demonstrar valor iterativamente: explica o valor do desenvolvimento iterativo, assumindo como obrigações: fornecer o valor incremental para ativar o feedback inicial e contínuo, adaptar os planos, adotar e gerenciar alterações e conduzir os principais riscos inicialmente no ciclo de vida;
e) Elevar o nível de abstração: sugere elevar o nível de abstração, reduzindo a complexidade e a quantidade de documentação necessárias ao projeto através da reutilização, do uso de ferramentas de modelagem de alto nível e ao estabilizar a arquitetura inicialmente;
f) Focalizar continuamente na qualidade: enfatiza que, para obter qualidade, é preciso um acompanhamento por todo o ciclo de vida do processo, aproveitando as oportunidades fornecidas pela abordagem iterativa para medida e correção de problemas no projeto.

4 ANÁLISE DOS VALORES DO MANIFESTO ÁGIL

Para organizar uma comparação entre as metodologias ágeis e as tradicionais, um bom ponto de partida é o Manifesto Ágil, por estabelecer claramente os conceitos mais importantes dentro da metodologia ágil e por focar claramente áreas que diferem da visão tradicional. Com esse critério, Glass (2001) pontua tais metodologias com base em sua opinião acerca dos 4 valores e 12 princípios estabelecidos no Manifesto Ágil, citando 8 pontos para as metodologias ágeis contra 5 das tradicionais.

Dentre as “tradicionais”, Highsmith (2001) inclui o RUP. Defende-se aqui o ponto de que o RUP não pode ser realmente colocado dessa forma, por ser um processo fortemente adaptável às necessidades de cada projeto. Essa visão é defendida, entre outros, por Kruchten (2001), Pollice (2001) e Smith (2001).

Ainda assim, o RUP apresenta diferenças conceituais relevantes em relação ao estabelecido pelas metodologias ágeis. Embora todo o processo do RUP seja descrito como adaptável, um ponto razoável para comparação é considerar as melhores práticas como uma descrição desse processo. Dentro da diretriz “Adaptação do Processo”, o RUP 2003 (IBM, 2006) confirmava:

> o RUP incentiva a adaptação. Contudo, ela não é uma licença para ignorar o processo como um todo. Os fundamentos do RUP estão embutidos nas melhores práticas. Siga o espírito dessas melhores práticas ao adaptar as atividades e os artefatos para que atendam às suas necessidades.

Com o RUP 7.0, as melhores práticas tornaram-se um material de suporte para os princípios chave, sendo estes os responsáveis por descrever os principais pontos defendidos pelo RUP e fornecer uma descrição conceitual do processo.

A seguir, apresentam-se os quatro valores do Manifesto Ágil e comenta-se o que o RUP apresenta com relação aos mesmos tópicos.

4.1 "Indivíduos e interações valem mais do que processos e ferramentas"

Equipes de pessoas constroem sistemas de software e, para isso, precisam trabalhar juntas de forma efetiva – incluindo, mas não se limitando a, programadores, testadores, gerentes de projeto, analistas e clientes. Os fatores mais importantes a considerar são as pessoas e como estas trabalham juntas, pois, sem isso, os melhores processos e as melhores ferramentas não serão de nenhuma valia (AMBLER, 2006). Segundo Martin (2002), "Um bom processo não salvará o projeto se a equipe não dispuser de bons profissionais, mas um processo ruim pode fazer até o melhor dos profissionais ineficiente".

Embora esse seja um conceito facilmente aceito pelos dois lados, é uma crítica à ênfase demasiada no processo dentro de um ambiente de desenvolvimento de software. O RUP 2003, em sua diretriz "Adaptação do Processo", destacava que a finalidade de um projeto de software é gerar um produto. Por outro lado, afirmava que "um processo eficaz permite que o projeto gere um produto que atenda às necessidades de seus envolvidos, dentro do prazo e do orçamento." Pode-se dizer que o enfoque principal ainda era no processo de desenvolvimento em si, ainda que preocupando-se em torná-lo eficaz, através da adaptação e das melhores práticas.

Nesse aspecto, o RUP 7.0 (IBM, 2006) ampliou o foco do processo de desenvolvimento para o negócio como um todo, apresentando como princípio chave "Colaborar Através de Equipes", no qual afirma que "o software é produzido por pessoas talentosas e motivadas trabalhando de forma colaborativa". Nesse princípio, comenta explicitamente que a colaboração tem sido o principal foco das comunidades de desenvolvimento ágil, reconhecendo também a popularidade do conceito de equipes auto-gerenciadas, fortemente defendido pelas metodologias ágeis. O RUP 7.0 sugere prover as equipes de ambientes de efetiva colaboração, reduzindo a necessidade de reuniões e liberando os membros da equipe para atividades mais produtivas.

4.2 "Software funcional vale mais do que uma documentação completa"

Deve-se lembrar que o resultado final na construção de um software é o produto e o componente chave do produto é o programa em si. A documentação é importante, não há como dispensar o material de suporte ao usuário ou o levantamento dos requisitos, mas, ao longo dos anos, alguns setores tornaram os processos de desenvolvimento totalmente centrados na produção de documentos (GLASS, 2001).

A discussão a respeito da necessidade ou não de documentação em um projeto de software é bastante extensa. Dentre todas as metodologias ágeis, o XP é a que mais contribui para essa discussão, levando muitas vezes ao entendimento de "proposições extremas" como a ausência de documentação e de modelagem (HIGHSMITH, 2001). Por outro lado, Ambler (2007) aponta que esse é um dos três conceitos comumente incorretos sobre o XP: não é feita documentação no XP; não é feita modelagem no XP e; se é feita modelagem, a única opção é a UML. Segundo ele, a modelagem é parte do XP. As "estórias de usuário" e os cartões Classe-Responsabilidade-Colaborador (CRC) fazem parte do conjunto de ferramentas do XP, podendo ambos ser considerados modelos, no sentido de que são artefatos que têm o objetivo de representar conceitualmente o domínio do problema e a estrutura da aplicação.

Quanto à documentação, Jeffries (2001) afirma que apenas a necessidade de comunicar requisitos para um cliente externo à equipe justifica a elaboração de uma documentação, e essa deve ser requisitada pelo cliente da mesma forma que qualquer outro recurso necessário. Embora o RUP, da mesma forma, afirme que a finalidade de um projeto de software é gerar um produto, segue o caminho de listar os diversos documentos que podem ser necessários ao projeto, deixando para o usuário escolher o que for mais adequado.

O único ponto a ser destacado é o vínculo entre o RUP e a UML, dois conceitos que evoluíram de forma interligada desde seu princípio. O RUP 2003 tinha como uma das suas melhores práticas "Modele visualmente (UML)", afirmando claramente que "o RUP usa a Linguagem Unificada de Modelagem (UML), uma notação consistente que pode ser aplicada à engenharia de sistemas e à engenharia de negócios. A UML representa a convergência da melhor prática em modelagem de software pela indústria de tecnologia de objetos" (IBM, 2006).

Essa integração entre o RUP e a UML criava uma distinção com as metodologias ágeis como a Agile Modeling (AM), sobre a qual Ambler (2007) afirma que "embora a UML defina uma importante coleção de modelos, a realidade é que ela estreitou a faixa de discussão dentro da comunidade de modelagem".

O RUP 7.0, embora ainda aplique fortemente a UML, não está mais amarrado a esse conceito. Em seu princípio chave "Elevar o nível de abstração", sugere reduzir a complexidade através do uso de ferramentas, estruturas e linguagens de mais alto nível. Aconselha o uso de linguagens padrões como a UML, mas sem que seja a única opção. Ainda assim, é importante comentar que o RUP 7.0 destaca a necessidade de focalizar na arquitetura logo no início, como uma forma efetiva de gerenciar a complexidade do sistema. Essa é uma diferença conceitual em relação a metodologias ágeis como o XP, que confiam na refatoração contínua do código como uma forma de obter naturalmente a arquitetura mais adequada para o software.

4.3 "A colaboração do cliente vale mais do que a negociação de um contrato"

Estabelecer um contrato com o cliente é importante. O entendimento sobre os direitos e responsabilidades de cada parte envolvida formam a base desse contrato, mas um contrato não substitui uma comunicação eficiente. Desenvolvedores de sucesso trabalham próximos ao seu cliente, investindo tempo e esforço para descobrir o que eles realmente necessitam, e educam seus clientes nesse processo (AMBLER, 2006). Segundo Martin (2002), um contrato que especifica os requisitos, o cronograma e o custo de um projeto é fundamentalmente falho. Os melhores contratos são aqueles que definem a forma como o cliente e a equipe de desenvolvimento trabalharão juntos.

Esse princípio é aplicado de maneira bastante intensiva no XP, o qual tem como uma de suas 12 práticas fundamentais o "Cliente dedicado" (*on-site customer*). Embora o cliente não possua um papel efetivo no RUP, a colaboração do cliente é uma prática que pode ser absorvida com vantagens em qualquer metodologia. Smith (2001) confirma que muitas das atividades do RUP seriam beneficiadas, em termos de eficiência, pela presença do cliente como membro da equipe. Com isso, muitos artefatos intermediários (especialmente documentos) poderiam ser dispensados do processo.

A principal mensagem é a mesma que motiva a abordagem iterativa de desenvolvimento: não é possível fazer o estabelecimento prévio de requisitos em sistemas de software. Tanto o RUP como as metodologias ágeis baseiam-se em um processo iterativo de desenvolvimento, variando apenas quanto à forma de conduzi-lo. O RUP 2003 definia como melhor prática "Desenvolva iterativamente", afirmando que "os usuários mudarão de idéia durante o processo. Essa é a natureza humana. Forçar os usuários a aceitarem o sistema como o imaginaram originalmente está errado. Eles mudam de idéia porque o contexto está sendo alterado; eles aprendem mais sobre o ambiente e a tecnologia e enxergam uma demonstração intermediária do produto durante o seu desenvolvimento." (IBM, 2006).

Embora o RUP nunca tenha favorecido os contratos ou apoiado o congelamento de requisitos, na sua versão 7.0 o enfoque é mais claramente baseado nos mesmos princípios do Manifesto Ágil. Um de seus princípios chave é "Equilibrar as Prioridades dos Investidores". Segundo esse princípio, ao invés de enviar equipes programadas para atacar cada elemento em uma lista de requisitos, é necessário *compreender e priorizar as necessidades de negócios e dos investidores*. Isso significa que a necessidade da equipe envolve o cliente ou o representante de serviços no projeto para garantir que seja possível compreender quais são essas necessidades.

Embora a diferença seja sutil, a referência ao envolvimento direto do cliente no processo é feita e o enfoque dirige-se mais ao negócio do que exclusivamente ao projeto.

4.4 "Responder às mudanças vale mais do que seguir um plano"

"Mudança é uma realidade no desenvolvimento de software, uma realidade que seu processo deve refletir. Não há nada errado em ter um planejamento do projeto. De fato, eu ficaria preocupado com um projeto que não tivesse um. Todavia, um planejamento precisa ser maleável, com espaço para alterações de acordo com as mudanças da situação, ou rapidamente se tornará irrelevante" (AMBLER, 2006).

Novamente, este é um ponto em comum entre os processos iterativos de desenvolvimento, dentre os quais se situa o RUP. Todavia, o RUP aborda os requisitos como sendo variáveis ao mesmo tempo em que valoriza o planejamento do projeto. O processo iterativo defendido pelo RUP não ocorre de forma aleatória; o número, a duração e o objetivo das iterações são planejados. As tarefas e as responsabilidades dos participantes são definidas. São capturadas medidas de progresso objetivas.

A visão das metodologias ágeis é um pouco mais radical nesse sentido. Um dos doze princípios do Manifesto Ágil é "Acolher mudanças nos requisitos, mesmo na fase final de desenvolvimento". Segundo Glass (2001), essa visão deve ser tomada com cuidado, pois o processo deve estar preparado para aceitar a mudança, mas avaliações de impacto nos custos e cronogramas são necessárias. Quanto mais perto do final do projeto, mais importantes serão os problemas, independente da metodologia de desenvolvimento.

Um ponto interessante a ser observado está relacionado com um dos princípios do Manifesto Ágil: "nossa maior prioridade é satisfazer o cliente, através de entregas contínuas de software executável". Essa abordagem é um pouco diferente da adotada pelo RUP 2003, no qual o desenvolvimento iterativo focalizava o andamento mais adequado do projeto, com a redução

dos riscos mais importantes nas primeiras iterações. No RUP 7.0, ao invés de "desenvolver iterativamente", o princípio chave vai mais além e coloca "demonstrar o valor iterativamente". Em cada iteração, devem ser executados alguns requisitos, *design*, implementação e teste do aplicativo, produzindo assim um distribuível, que é a etapa mais próxima da solução final.

Nesse princípio, pode-se observar claramente a influência do Manifesto Ágil sobre o RUP 7.0. Ao invés de contar com a avaliação das especificações, como as especificações dos requisitos, modelos do *design* ou planos, sugere avaliar o quanto e como o código desenvolvido realmente funciona em cada iteração, testando resultados e demonstrações do código de trabalho aos *stakeholders*.

5 CONCLUSÕES

Segundo seus autores, o RUP é "um conjunto de práticas coletadas de engenharia de software que são continuamente aprimoradas, com regularidade, para refletirem alterações nas práticas do segmento de mercado". Nesse sentido, este artigo procurou ressaltar o fato de que o Manifesto Ágil, como um marco na difusão dos valores e princípios que norteiam o chamado "desenvolvimento ágil", alterou efetivamente a visão da comunidade técnica e, com ela, as práticas do mercado. A abordagem dada pelo RUP 7.0 em seus "Princípios Chave para o Desenvolvimento Baseado em Negócios" incorpora esses valores em definitivo à filosofia subjacente ao processo.

Embora a comunidade de desenvolvimento ainda não liste o RUP no conjunto de metodologias ágeis, isso se deve a uma deficiência no estabelecimento do conceito de "ágil". Conceitualmente, os valores defendidos pelo RUP coincidem quase que exatamente com os do Manifesto Ágil. Um motivo para isso, como apontado por Fowler (2005), é o fato de que o RUP não é um método, mas sim um *framework* para elaboração de processo de desenvolvimento de software. O primeiro passo para um projeto usando RUP é justamente definir o processo a ser efetivamente adotado.

Segundo Kruchten (2001, p. 27), "Agilidade, para uma empresa de desenvolvimento de software, é a habilidade de adaptar-se e reagir rápida e corretamente às mudanças no seu ambiente e nas necessidades impostas por esse ambiente. Um processo ágil é o que suporta prontamente esse grau de adaptabilidade". O RUP ressaltou a sua agilidade, com sua versão 7.0, ao adaptar a sua própria abordagem a uma nova cultura "ágil" presente no mercado.

REFERÊNCIAS

ABRAHAMSSON, Pekka et al. **Agile software development methods**: Review and analysis.VTT Publications 478. Oulu, Finland: VTT Publications, 2002.

AMBLER, Scott W. **Examining the Agile Manifesto**. 2006. Disponível em: <http://www.ambysoft.com/essays/agileManifesto.html>. Acesso em 06 ago. 2007.

_____. **Agile Modeling and Extreme Programming (XP)**. 2007. Disponível em: <http://www.agilemodeling.com/essays/agileModelingXP.htm>. Acesso em: 21 maio 2007.

ARSANJANI, Ali; JOHNSTON, Simon; SMITH, John. **The RUP Update for Service-Oriented Architecture**. Rational Software White Paper, 2005.

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

BECK, Kent et al. **Manifesto for Agile Software Development.** 2001. Disponível em: <http://www.agilemanifesto.org>. Acesso em: 21 maio 2007.

BEZERRA, Eduardo. **Princípios de Análise e Projeto de Sistemas com UML.** 2. ed. Rio de Janeiro: Campus, 2007.

BOOCH, Grady; MARTIN, Robert C.; NEWKIRK, James. The Process. In: __. **Object-Oriented Analysis and Design with Application.** London: Addison Wesley Longman, 1998. p. 93-108. (texto preliminar). Disponível em: <http://www.objectmentor.com/resources/articles/RUPvsXP.pdf>. Acesso em: 22 maio 2007.

BONA, Cristina. **Avaliação de Processos de Software**: Um Estudo de Caso em XP e ICONIX. 2002. Dissertação (Mestrado em Ciências da Computação). Universidade Federal de Santa Catarina, 2002.

FOWLER, Martin. **The New Methodology**. 2005. Disponível em: <http://www.martinfowler.com/articles/newMethodology.html>. Acesso em: 18/08/2007.

FOWLER, Martin; HIGHSMITH, Jim. **The Agile Manifesto**. Dr. Dobb´s Portal, ago., 2001. Disponível em <http://www.ddj.com/architect/184414755>. Acesso em: 15 ago. 2007.

GLASS, Robert L. Agile Versus Traditional: Make Love, Not War! **Cutter IT Journal**, v. 14, n. 12, dez., 2001. Disponível em: <www.emory.edu/BUSINESS/readings/MethodologyDebate.pdf>. Acesso em: 22 maio 2007.

HIGHSMITH, Jim. The Great Metodologies Debate: Opening Statement. **Cutter IT Journa**, v. 14, n. 12, dez., 2001. Disponível em: <http://www.emory.edu/BUSINESS/readings/MethodologyDebate.pdf>. Acesso em: 22 maio 2007.

IBM. **Rational Unified Process (software).** versão 7.0. USA: IBM Rational, 2006.

JEFFRIES, Ron. **Essential XP: Documentation**. 2001. Disponível em <http://www.xprogramming.com/xpmag/expDocumentationInXp.htm>. Acesso em 26 maio 2007.

KRUCHTEN, Philippe. **The Rational Unified Process**: an introduction. 2. ed. Reading, Massachusetts: Addison-Wesley, 2000.

_____. Agility with the RUP. **Cutter IT Journal**, v. 14, n. 12, dez. 2001. Disponível em: <http://www.emory.edu/BUSINESS/readings/MethodologyDebate.pdf>. Acesso: 22 maio 2007.

MARTIN, Robert C. **Agile Processes**. 2002. Disponível em: <http://www.objectmentor.com/resources/articles/agileProcess.pdf>. Acesso em 15 ago. 2007.

MARTINS, Paula V. & SILVA, Alberto R. Comparação de Metamodelos de Processos de Desenvolvimento de Software. *In*: Conferência para a Qualidade nas Tecnologias da Informação e Comunicações (QUATIC), 5, 2004. **Actas**… 2004. Disponível em: <http://berlin.inesc-id.pt/alb/static/papers/2004/pv-quatic2004.pdf>. Acesso em: 5 jun. 2007.

MENJOGE, Zehlam. **Software Development using the Knowledge Insight Approach**. North Carolina State, 2003. Thesis (Master of Science in Computer Science) North Carolina State University, 2003.

NONEMACHER, Marcos L. **Comparação e Avaliação entre o Processo RUP de Desenvolvimento de Software e a Metodologia Extreme Programming**. Florianópolis, 2003. Dissertação (Mestrado em Ciências da Computação) Universidade Federal de Santa Catarina, 2003.

POLLICE, Gary. **Using the IBM Rational Unified Process for Small Projects**: Expanding Upon eXtreme Programming. Rational Software White Paper, 2001. Disponível em <http://www3.software.ibm.com/ibmdl/pub/software/rational/web/whitepapers/2003/tp183.pdf>. Acesso em: 22 maio 2007.

RUNESON, Per; GREBERG, Peter. Extreme Programming and Rational Unified Process: Contrasts or Synonyms? *In*: Conference on Software Engineering Research and Practice in Sweden, 4, oct. 2004. **Proceedings…** Linköping University, 2004.

SMITH, John. **A Comparison of RUP® and XP**. Rational Software White Paper, 2001. Disponível em: <http://ftp.software.ibm.com/software/rational/web/whitepapers/2003/TP167.pdf>. Acesso em: 22 maio 2007.

Originais recebidos em: 24 jan. 2008.
Texto aprovado em: 29 fev. 2008.

SOBRE OS AUTORES

Sergio Akio Tanaka

Mestre em Ciência da Computação pela Universidade Federal do Rio Grande do Sul; Pós-graduado pela Universidade Estadual de Londrina nas áreas de Redes de Computadores e Banco de Dados e em Análise de Sistemas pela UniFil. Especialista em Gestão Empresarial pelo ISE – Instituto Superior de Ensino em convênio com o IESE de Barcelona. Graduado em Processamento de Dados. Engenheiro de Software/Arquiteto certificado pela IBM/Rational e Sales Professional em IBM Websphere. Ao Longo da sua carreira, foi Diretor de Novas Tecnologias da K2Solutions (empresa com 500 funcionários). Atuou como Gerente Geral da PLATIN/ADETEC e Consultor do Agente Softex/ADETEC no mercado Espanhol, Coordenou a área de TI do SENAC por 11 anos e foi Gerente da KAIZEN - DATABASE Marketing por 5 anos. Atualmente, é responsável pela Unidade IBM Rational, e também, consultor certificado pela IBM Rational e Websphere na AUDARE Engenharia de Software. Na área acadêmica, atua como professor e coordenador da UNIFIL - Universidade Filadélfia nos cursos de Pós-Graduação da área de Computação e Pesquisa; Professor Ad-hoc da Unopar e UEL em cursos de Pós-graduação. Possui expertise em Gerência de Projetos, Arquitetura de Software, UML, Construção de Frameworks e Componentes, Processo de Desenvolvimento de Software, Gerência de Requisitos, Gerência de Mudanças, Workflow, Modelagem Web e de Negócios. Tem fluência nos idiomas Inglês e Espanhol e forte experiência no mercado Internacional (Europa, Japão e América Latina). E-mail: sergio.tanaka@audare.com.br

André Banki

Engenheiro Civil pela Universidade Federal de Santa Catarina (UFSC), Mestre em Engenharia Civil (na área de Estruturas), também pela UFSC, e Especialista em Engenharia de Software pelo SENAI/Florianópolis, trabalha como Gerente de Desenvolvimento na empresa AltoQi Tecnologia em Informática Ltda., tendo sido responsável pela concepção e coordenação do desenvolvimento de diversos sistemas de renome nacional na área de Engenharia.
E-mail: andre.banki@gmail.com