O Manifesto Ágil

Facilitar mudanças é mais efetivo do que tentar preveni-las. Aprender a confiar nas suas habilidades para responder a eventos imprevisíveis é mais importante do que confiar nas suas habilidades de planejamento contra desastres.

Nós últimos 12-18 meses, várias publicações – *Software Development*, *IEEE Software*, *Cutter IT Journal*, *Software Testing*, *Quality Engineering* e mesmo *The Economist*, publicaram artigos nos quais Martin Fowler chamou de a Nova Metodologia (www.martinfowler.com/articles/newMethodology.html), refletindo o crescente interesse nestes novos enfoques no desenvolvimento de software (*Extreme Programming*, *Crystal Methodologies*, *SCRUM*, *Adaptive Software Development*, *Feature-Driven Development*, *Dynamic Systems Development Methodology* entre outros). Adicionalmente a essas metodologias batizadas, várias organizações desenvolveram suas próprias abordagens "*lights*" para desenvolverem software.

Formação da Aliança Ágil

De 11 a 13 de Fevereiro de 2001, na *The Lodge* da estação de esqui *Snowbird* nas montanhas Wasatch em Utah, EUA, 17 pessoas se encontraram para conversar, esquiar, relaxar e tentar achar um denominador comum. O que surgiu foi a Aliança do Desenvolvimento Ágil de Software. Como seria difícil outra reunião com estes anarquistas organizacionais, o que resultou desta reunião foi o simbólico – Manifesto do Desenvolvimento Ágil de Software – assinado por todos participantes. Embora o Manifesto apresente algo de específico, muitos membros da Aliança tratam os temas com mais profundidade. Ao final de dois dias de reunião, o mentor da *Extreme Programming* Bob Martin brincou que ele estava quase fazendo uma declaração "melosa". Pois, tingido com humor, o sentimento de Bob foi compartilhado pelo grupo – todos nós gostamos de trabalhar com pessoas com as quais compartilhamos objetivos e valores baseados no mútuo respeito e confiança, que promovem a colaboração, cujo foco está nos modelos das organizações e que produzem tipos de comunidades de profissionais nas quais gostaríamos de trabalhar.

O movimento da metodologia ágil não é anti-metodologia; de fato, muitos de nós queremos restaurar a credibilidade desta palavra. Também queremos restaurar um equilíbrio. Nós abraçamos a modelagem mas não meramente arquivar alguns diagramas num empoeirado repositório corporativo. Nós abraçamos a documentação mas não o gasto de resmas de papel em nunca mantidos e raramente utilizados tomos. Nós planejamos mas reconhecemos os limites do planejamento em ambientes turbulentos. Aqueles que se distinguem como proponentes de XP, SCRUM ou qualquer outra metodologia ágil como "*hackers*" ignoram ambos a metodologia e a definição original do termo (um *hacker* foi originalmente definido como um programador que gosta de resolver problemas complexos de programação, ao invés daquele que pratica desenvolvimento *ad hoc* ou destruição).

Anteriormente, Alistair Cockburn identificou o desagrado geral da palavra *light*: "Eu não me importo que as metodologias sejam chamadas de leve em peso, mas eu não tenho certeza que quero ser referido como um "peso-leve" participando de uma reunião de metodologistas de pouco peso. Isso se parece com um monte de magrelos e fracos de idéia tentando se lembrar que dia é hoje". Então nossa primeira tarefa foi encontrar um adjetivo com o qual poderíamos conviver. Agora nossos processos são "ágeis", mesmo que algum de nós esteja um pouco capenga.

O resultado desta reunião (e a frenética interação seguida *online*) foi o Manifesto Ágil. Enquanto o propósito e os princípios do Manifesto foram desenvolvidos pelo grupo todo, nós (Jim e Martin, ambos autores do Manifesto) adicionamos, para este artigo, nossas interpretações e explicações.

O Manifesto Ágil: Propósito

"Nós estamos descobrindo maneiras melhores de se desenvolver software, desenvolvendo e ajudando outras pessoas a desenvolver. Nós valorizamos:

- Indivíduos e interações ao invés de processos e ferramentas
- Software operante ao invés de documentações completas
- Colaboração do cliente ao invés de negociações contratuais
- Responder à mudanças ao invés de seguir um planejamento"

Esta declaração tem um número fascinante de aspectos, não apenas aqueles que as 17 pessoas estavam tentando concordar. Primeiro a palavra *descobrindo*. Embora este grupo fosse composto por experientes e reconhecidos "gurus" desenvolvedores, a palavra *descobrindo* foi escolhida para assegurar (ou assustar) a audiência que os membros da Aliança não têm todas as respostas e não adotam a teoria da bala de prata. Segundo, a palavra *desenvolvendo* indica que os membros de fato praticam estes métodos nos seus trabalhos. Ken Schwaber (um proponente do SCRUM) falou dos seus dias de vendedor de ferramentas para automatizar metodologias "pesadas". Impressionado com a capacidade de resposta da empresa de Ken, Jeff Sutherland o perguntou quais das metodologias "pesadas" ele utilizava no seu desenvolvimento interno. "Eu ainda lembro da expressão do Jeff", recorda Ken, "quando eu disse – 'Nenhuma, se utilizássemos alguma delas, estaríamos fora do mercado!'" Terceiro, este grupo é para ajudar e não para ditar. Os membros da Aliança querem ajudar outras pessoas através dos métodos ágeis e ampliar seus próprios conhecimentos aprendendo com o quais tentam ajudar.

As declarações dos valores têm um formato: em cada uma, o primeiro segmento indica a preferência, enquanto o último descreve um item que, embora importante, é de prioridade menor. Esta distinção está no coração da agilidade, mas simplesmente pedir às pessoas para listarem o que é valorizado não revela as diferenças essenciais. Roy Singham, chefe do Martin na ThoughtWorks, colocou bem quando disse que é a periferia dos casos, as escolhas difíceis, que o interessa. "De fato, nós valorizados o planejamento, documentação completa, processos e ferramentas. Isso é fácil de falar. O difícil é perguntar 'o que você valoriza *mais* ?'"

A Aliança reconhece a importância dos processos e ferramentas, reconhecendo adicionalmente que a interação entre indivíduos capacitados tem ainda maior importância. Da mesma forma, documentação completa não é necessariamente ruim, mas o foco primário deve permanecer no produto final – a entrega de software operante. Entretanto, toda equipe de projeto precisa determinar por si só qual documentação é absolutamente essencial.

Negociação contratual, sendo o acordo para um projeto interno ou um contrato legal, não é uma prática ruim, apenas insuficiente. Contratos e acordos podem apresentar condições de fronteira nas quais as partes podem trabalhar, mas somente através de contínua colaboração é que a equipe consegue entender e entregar o que o cliente deseja.

Ninguém discute que seguir um planejamento é uma boa idéia – certo ? Bem, sim e não. No mundo turbulento dos negócios e tecnologia, seguir escrupulosamente um plano pode ter conseqüências desastrosas, mesmo se o plano for executado fielmente. Portanto, um plano cuidadosamente elaborado, pode se tornar perigoso se ele impedir mudanças. Nós examinamos vários projetos de sucesso e apenas alguns, se é que teve algum, entregaram o que foi planejado inicialmente, ainda que eles tiveram sucesso devido à equipe de desenvolvimento que respondia sempre e sempre às mudanças externas.

O Manifesto Ágil: Princípios

Nossa prioridade mais alta é satisfazer o cliente através de entregas contínuas e antecipadas de software válido.

Numa recente workshop, um gerente de desenvolvimento de software questionou a abordagem das funcionalidades e estórias no planejamento de ciclos iterativos. “Mas os documentos de especificação de requisitos e arquitetura não são importantes ?” ele perguntou. “Sim”. Respondeu Jim. “Eles *são* importantes, mas nós devemos entender que o cliente não se importa com documentos, diagramas UML ou integração com o legado. Clientes se importam se você está ou não entregando software operante para ele a cada ciclo de implantação – alguns pedaços da funcionalidade dos negócios que provam para ele que o aplicativo em evolução serve às suas necessidades”.

Implementar o princípio “valor para o cliente” é uma das atividades “mais fáceis de dizer do que de fazer”. O gerenciamento tradicional de projeto assume que cumprir um plano é igual ao sucesso do projeto que é igual a demonstrar valor ao cliente. A volatilidade associada aos projetos de hoje em dia exige que o valor do cliente seja re-avaliado freqüentemente, e ir de encontro ao plano do projeto original pode não ter muito impacto no sucesso do projeto.

Mudanças nos requisitos são bem-vindas, mesmo as que chegam tarde no desenvolvimento. Processos ágeis asseguram a mudança como uma vantagem competitiva do cliente.

O crescimento da imprevisibilidade do futuro é um dos aspectos mais desafiadores da nova economia. Turbulências – nos negócios e na tecnologia – causam mudanças, que

podem ser vistas tanto quanto ameaças a serem evitadas ou como oportunidades a serem abraçadas.
Ao invés de resistir às mudanças, o enfoque ágil se vira para acomodá-las da maneira mais fácil e eficiente possível, enquanto mantendo ciência de suas conseqüências. Embora a maioria das pessoas concorde que feedback é importante, elas geralmente ignoram o fato que o resultado do aceite de um feedback é uma mudança. Metodologias ágeis asseguram este resultado pois seus proponentes entendem que facilitar mudanças é mais efetivo do que tentar evitá-las.

Entregar software produtivo freqüentemente, de algumas semanas a alguns meses, de preferência os tempos mais curtos.

Por muitos anos, os gurus dos processos vêm dizendo para todo mundo utilizar um estilo incremental e iterativo de desenvolvimento de software, através de múltiplas entregas com funcionalidades crescentes. Enquanto esta prática vem crescendo em uso, ela ainda não é predominante, entretanto, é essencial em projetos ágeis. Além do que, nós trabalhamos duro para reduzir o tempo dos ciclos de entrega.
Lembrar que entrega não é o mesmo que *release*[1]. O pessoal do comercial pode ter razões válidas para não colocar código em produção a cada par de semanas. Temos vistos projetos que não chegam até uma *release* por mais de anos. Mas isso não os isentam de um rápido ciclo interno de entregas que permite que todos avaliem e aprendam com o produto em crescimento.

Pessoal de negócio e desenvolvedores trabalham juntos diariamente durante o projeto.

Muitos caras querem comprar software da mesma maneira que compram carros. Eles têm uma lista de características em mente, negociam o preço e pagam por aquilo que pediram. Este modelo simples de compra é chamativo, mas para a maioria dos projetos de software, não funciona. Assim, desenvolvedores ágeis respondem com uma mudança radical no nosso conceito do processo de requisitos.
De início, nós não esperamos um conjunto detalhado de requisitos para ser assinado no início do projeto; ao invés, nós temos uma visão de alto nível dos requisitos que é sujeita a freqüentes mudanças. Claramente, isso não é suficiente para projetar nem codificar, então a lacuna é preenchida com interações freqüentes entre o pessoal de negócio e os desenvolvedores. A freqüência deste contato geralmente surpreende as pessoas. Nós colocamos “diário” a princípio para enfatizar que o contínuo comprometimento do cliente seja parte, e de fato divida a responsabilidade, do projeto de software.

Criar projetos em torno de indivíduos motivados, proporcionar o ambiente e suporte que eles necessitam e confiar que eles farão o serviço.

Implante todas as ferramentas, tecnologias e processos que você quiser, até o nosso processo ágil, mas no final, são as pessoas que fazem a diferenças entre sucesso e falha. Nós percebemos, entretanto, que quanto mais duro a gente trabalhava em idéias de processos, o melhor que podíamos esperar era um efeito de segunda ordem no projeto. Então é importante maximizar o fator humano de primeira ordem.

Para muitas pessoas, confiança é a coisa mais difícil de se passar. Decisões devem ser tomadas pelas pessoas que mais conhecem a situação. Isto significa que os gerentes devem confiar à suas equipes as decisões das coisas que eles são pagos para conhecer.

O método mais eficiente e efetivo para transmitir informações entre e para a equipe de desenvolvimento é conversão cara a cara.

Inevitavelmente, quando discutimos metodologias ágeis, surge o tópico da documentação. Nossos oponentes muitas vezes aparecem furiosos, zombando da nossa "falta" de documentação. Isso é suficiente para nos fazer gritar, "o caso *não* é documentação – é *entendimento*!" Sim, documentos físicos têm peso e substância, mas a medida real do sucesso é abstrata. Será que as pessoas envolvidas ganham o entendimento que precisam ? Muitos de nós somos escritores, mas independente de nossos prêmios e vendas de livros, nós sabemos que escrever é difícil e é um meio ineficiente de comunicação. Nós o utilizamos pois temos que, porém muitas equipes de projeto podem e devem utilizar uma técnica mais direta de comunicação.

"Conhecimento tácito não pode ser transferido extraindo-o da cabeça das pessoas para o papel", escreveu Nancy Dixon em *Common Knowledge* (Hardvard Business School Press, 2000). "Conhecimento tácito pode ser transferido movendo ao redor as pessoas que o detém. A razão é que o conhecimento tácito não é somente fatos mas relacionamentos entre fatos – isto é, a maneira que pessoas podem combinar certos fatos para lidarem com uma situação específica". Então a distinção entre ágil e metodologias centradas em documentos não é documentação extensiva versus sem documentação; mas sim um conceito distinto que mistura a documentação e conversação requerida para obter o entendimento.

Software produtivo é a medida primária do progresso.

Freqüentemente, nós vemos equipes de projetos que não percebem que estão em perigo até um pouco antes da entrega. Eles fizeram os requisitos no prazo, o projeto no prazo e talvez até os códigos no prazo, mas os testes e a integração levaram muito mais tempo do que eles imaginaram. Nós somos a favor primariamente do desenvolvimento iterativo pois ele fornece marcos[2] que não podem ser burlados, os quais transmitem uma medida precisa do progresso e profundo entendimento dos riscos envolvidos num dado projeto. Como Chet Hendrickson, co-autor de *Extreme Programming Installed* (Addison-Wesley, 2000), lembra, "Se um projeto vai falhar, eu prefiro saber depois de um mês do que depois de 15".

"Software produtivo é a medida do progresso porque não há outra forma de se capturar as sutilezas dos requisitos: Documentos e diagramas são muito abstratos para permitir que o usuário 'saiam cantando os pneus'", disse Dave Thomas, co-autor do *The Pragmatic Programmer* (Addison-Wesley, 1999).

Processos ágeis promovem um desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.

Nossa indústria é caracterizada por longas noites e finais de semana, durante os quais as pessoas tentam desfazer os erros dos planejamentos irresponsáveis. Ironicamente, estas longas horas na verdade não levam a grande produtividade. Martin e Kent Beck sempre

se lembram de empresas que gastam o dia inteiro removendo os erros feitos na noite anterior.
Agilidade depende de pessoas que estão alertas e criativas, e conseguem manter esta atenção e criatividade durante todo o projeto de desenvolvimento de software. Desenvolvimento sustentável significa encontrar um ritmo de trabalho (40 ou tantas horas por semana) que a equipe consiga sustentar durante todo tempo e permanecer saudável.

Atenção contínua à excelência técnica e boa solução[3] melhoram a agilidade.

Muitas pessoas quando olham para o desenvolvimento ágil vêem traços do esforço "rápido e sujo" RAD (*Rapid Application Development*) da última década. Mas, enquanto o desenvolvimento ágil é similar ao RAD em termos de velocidade de flexibilidade, há uma grande diferença quando surge a clareza técnica. Abordagens ágeis enfatizam a qualidade da solução porque solução de qualidade é essencial para manter a agilidade.
Um dos aspectos capciosos, no entanto, é o fato de que os processos ágeis assumem e encorajam a alteração dos requisitos enquanto o código vai sendo escrito. Tal qual o projeto não pode puramente ser uma atividade antecipada a ser completada antes da construção. Ao invés, o ato de projetar é uma atividade contínua que é realizada durante todo o ciclo do projeto. Cada e toda iteração terão o trabalho de projetar.
A diferença é que os processos ágeis enfatizam os diferentes estilos de projeto. FDD [*Feature-Driven Development*] tem um passo explícito no início de cada iteração no qual a solução é executada, usualmente graficamente com UML. XP deposita grande ênfase na re-fabricação para permitir que o projeto evolua na medida que o desenvolvimento prossegue. Mas todos estes processos pedem emprestados uns para os outros: FDD usa a re-fabricação quando os desenvolvedores re-visitam decisões anteriores de projeto, e XP encoraja pequenas sessões de projeto antes da codificação. Em todos os casos, a solução é continuamente melhorada durante o projeto.

Simplicidade – a arte de maximizar a quantidade de trabalho não feita – é essencial.

Qualquer tarefa de desenvolvimento de software pode ser abordada por uma variedade de métodos. No projeto ágil, é particularmente importante utilizar abordagens simples, pois elas são mais fáceis de mudar. É mais fácil acrescentar alguma coisa num processo muito simples do que tirar alguma coisa de um processo muito complicado. Por isso, há um grande gosto pelo minimalismo em todos os métodos ágeis. Inclua somente aquilo que todos precisam ao invés daquilo que alguém precisa, para facilitar a inclusão de alguma coisa quando uma equipe específica precisar.
"Simples, de princípios e propósitos claros criam comportamentos complexos e inteligentes", disse Dee Hock, ex-*CEO* da Visa International. "Regras complexas e regulamentos criam comportamentos simples e estúpidos". Nenhuma metodologia pode atender a toda complexidade dos modernos projetos de software. Dar às pessoas um conjunto simples de regas e encorajá-las na criatividade produzirá um resultado muito melhor do que impor regras e regulamentos complexos.

As melhores arquiteturas, requisitos e projetos emergem de equipes auto-organizadas.

Ao contrário do que você tem ouvido, forma não quer dizer função. Forma quer dizer falha. "A forma de fazer as coisas está sempre sujeita à mudança em resposta às imperfeições reais ou percebidas, elas deixam de funcionar devidamente", escreveu Henry Petroski, professor de engenharia civil e autor do *The Evolution of Useful Things* (Vintage Books, 1994). Stuart Brand escreve que a idéia "forma quer dizer função" vem enganando arquitetos na crença que eles podem prever como os prédios serão de fato utilizados.

As visões de Petroski são similares a um dos dois pontos deste princípio – que os melhores projetos (arquiteturas e requisitos) emergem do desenvolvimento iterativo e uso ao invés de planos antecipados. O segundo ponto do princípio é que propriedades emergentes (*emergência*, propriedade chave dos sistemas complexos, aproximadamente traduzida por inovação e criatividade em organizações humanas) são melhores quando geradas a partir de equipes auto-organizadas com alta interatividade e poucas regras de processo.

Em intervalos regulares, a equipe reflete sobre como se tornar mais efetiva e então sintoniza e ajusta seu comportamento de forma apropriada.

Métodos ágeis não são alguma coisa que você pega e segue religiosamente. Você pode começar com um daqueles processos [citados acima], mas nós todos reconhecemos que não podemos aparecer com o processo certo para toda situação. Assim, qualquer equipe ágil deve refinar e refletir durante o caminho, constantemente melhorando suas práticas para as situações locais.

Jim tem trabalhado com consultorias para desenvolver a *Adaptive Software Development* – combinação da metodologia *Extreme Programming*. A primeira equipe que a utilizou, a modificou de imediato. Martin tem trabalhado com um grande número de equipes na ThoughtWorks para customizar[4] as práticas da *Extreme Programming* às várias situações dos projetos. Confiar nas pessoas, acreditar nas capacidades individuais e na interação do grupo são chaves para o sucesso que se estendem em equipes confiáveis que monitoram e melhoram seus próprios processos de desenvolvimento.

Rumo ao Futuro Ágil

As respostas ao Manifesto Ágil têm sido gratificantes. Muitos e-mails expressam sentimentos tais como, "Meu gerente de produto colocou o Manifesto na sua parede". Muitos dos colegas do Martin na ThoughtWorks deram uma passada para dizer o quanto eles compartilhavam dos valores.

Uma questão levantada de imediato foi se a Aliança é ou não a precursora da, conforme disse um participante, Metodologia Peso-Leve Unificada [*Unified Lightweight Methodology*]. Absolutamente não! Enquanto o grupo acredita que um conjunto de propósitos e princípios comuns beneficiará os usuários de metodologias ágeis, nós somos igualmente convictos que a variedade e diversidade de práticas são necessárias. Quando isso vai para as metodologias, cada projeto é distinto e cada equipe de projeto é distinta – não há uma solução tamanho único.

E o futuro ? Podemos confiantemente dizer que não sabemos. Agilidade está mais para confiança na habilidade das pessoas de responderem a eventos imprevisíveis do que na confiança da habilidade delas de previsão. Nós também sabemos que o relacionamento pessoal formado em nossa colaboração, de longe, importa mas do que o documento que produzimos. Uma coisa é clara: estamos apenas começando.

O Manifesto para o Desenvolvimento Ágil de Software
Dezessete anarquistas concordam:

Nós estamos descobrindo maneiras melhores de se desenvolver software, desenvolvendo e ajudando outras pessoas a desenvolver. Nós valorizamos:

- Indivíduos e interações ao invés de processos e ferramentas
- Software operante ao invés de documentações completas
- Colaboração do cliente ao invés de negociações contratuais
- Responder à mudanças ao invés de seguir um planejamento

Isto é, enquanto nós valorizamos os itens da direita, nós valorizamos mais os itens da esquerda. Nós seguimos os seguintes princípios:

- Nossa prioridade mais alta é satisfazer o cliente através de entregas contínuas e antecipadas de software válido.
- Mudanças nos requisitos são bem-vindas, mesmo as que chegam tarde no desenvolvimento. Processos ágeis asseguram a mudança como uma vantagem competitiva do cliente.
- Entregar software produtivo freqüentemente, de algumas semanas a alguns meses, de preferência os tempos mais curtos.
- Pessoal de negócio e desenvolvedores trabalham juntos diariamente durante o projeto.
- Criar projetos em torno de indivíduos motivados, proporcionar o ambiente e suporte que eles necessitam e confiar que eles farão o serviço.
- O método mais eficiente e efetivo para transmitir informações entre e para a equipe de desenvolvimento é conversão cara a cara.
- Software produtivo é a medida primária do progresso.
- Processos ágeis promovem um desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.
- Atenção contínua à excelência técnica e boa solução melhoram a agilidade.
- Simplicidade – a arte de maximizar a quantidade de trabalho não feita – é essencial.
- As melhores arquiteturas, requisitos e projetos emergem de equipes auto-organizadas.
- Em intervalos regulares, a equipe reflete sobre como se tornar mais efetiva e então sintoniza e ajusta seu comportamento de forma apropriada.

Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas
www.agilealliance.org

Agosto de 2001

Notas da Tradução:

1 *deliver* foi traduzido com entrega porém optou-se por deixar *release* no original

2 no original *milestones*

3 *project* foi traduzido como projeto no sentido de "atividade em andamento" e *design*, embora também projeto, foi traduzido como solução

4 no original *tailor*

João Rotta Neto
joaorotta@hotmail.com

Junho de 2002