Desenvolvimento ágil de software

Origem: Wikipédia, a enciclopédia livre.

Desenvolvimento ágil de *software* (em inglês: *Agile software development*) ou **Método ágil** é uma expressão que define um conjunto de metodologias utilizadas no desenvolvimento de *software*. As metodologias que fazem parte do conceito de desenvolvimento ágil, tal como qualquer metodologia de *software*, providencia uma estrutura conceitual para reger projetos de engenharia de software.

Existem inúmeros frameworks de processos para desenvolvimento de software. A maioria dos métodos ágeis tenta minimizar o risco pelo desenvolvimento do software em curtos períodos, chamados de iteração, os quais gastam tipicamente menos de uma semana a até quatro. Cada iteração é como um projeto de *software* em miniatura de seu próprio, e inclui todas as tarefas necessárias para implantar o mini-incremento da nova funcionalidade: planejamento, análise de requisitos, projeto, codificação, teste e documentação. Enquanto em um processo convencional, cada iteração não está necessariamente focada em adicionar um novo conjunto significativo de funcionalidades, um projeto de software ágil busca a capacidade de implantar uma nova versão do software ao fim de cada iteração, etapa a qual a equipe responsável reavalia as prioridades do projeto.

Métodos ágeis enfatizam comunicações em tempo real, preferencialmente cara a cara, a documentos escritos. A maioria dos componentes de um grupo ágil deve estar agrupada em uma sala. Isso inclui todas as pessoas necessárias para terminar o software: no mínimo, os programadores e seus *clientes* (clientes são as pessoas que definem o produto, eles podem ser os gerentes, analistas de negócio, ou realmente os clientes). Nesta sala devem também se encontrar os testadores, projectistas de iteração, redactores técnicos e gerentes.

Métodos ágeis também enfatizam o software funcional como uma medida primária de progresso. Combinado com a comunicação cara-a-cara, métodos ágeis produzem pouca documentação em relação a outros métodos, sendo este um dos pontos que podem ser considerados negativos. É recomendada a produção de documentação que realmente será útil.

Índice

Valores

Segundo a página *Agile Manifest*[1] - Manifesto ágil os valores relacionados ao Desenvolvimento ágil de software são:

- Indivíduos e iterações mais que processos e ferramentas;
- Software funcional mais que documentação abrangente;
- Colaboração do cliente mais que negociação de contratos;
- Responder a mudanças mais que seguir um plano

Ou seja, o item à esquerda sempre tem maior importância do que o item à direita

Princípios

Os princípios do desenvolvimento ágil[2] valorizam

- Garantir a satisfação do consumidor entregando rapidamente e continuamente software funcionais;
- Até mesmo mudanças tardias de escopo no projecto são bem-vindas para garantir a vantagem competitiva do cliente;
- Software funcionais são entregues frequentemente (semanas, ao invés de meses);
- Cooperação diária entre pessoas que entendem do 'negócio' e desenvolvedores;
- Projetos surgem através de indivíduos motivados, entre os quais existe relação de confiança.
- A maneira mais eficiente e efetiva de transmitir informações é conversar cara a cara;
- Software funcionais são a principal medida de progresso do projeto;
- Processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes para manter um ritmo constante indefinidamente.
- Design do software deve prezar pela excelência técnica;
- Simplicidade é essencial;
- As melhores arquiteturas, requisitos e projetos emergem de equipes auto-organizadas;
- Em intervalos regulares, a equipe reflete sobre como para tornar-se mais eficaz, então sintoniza e ajusta seu comportamento apropriadamente.

História

As definições modernas de desenvolvimento de software ágil evoluíram a partir da metade de 1990 como parte de uma reação contra métodos "pesados", caracterizados por uma pesada regulamentação, regimentação e micro gerenciamento usado o modelo em cascata para desenvolvimento. O processo originou-se da visão de que o modelo em cascata era burocrático, lento e contraditório a forma usual com que os engenheiros de software sempre realizaram trabalho com eficiência.

Uma visão que levou ao desenvolvimento de métodos ágeis e iterativos era retorno a prática de desenvolvimento vistas nos primórdios da história do desenvolvimento de software [1] (http://www2.umassd.edu/SWPI/xp/articles/r6047.pdf).

Inicialmente, métodos ágeis eram conhecidos como *métodos leves*. Em 2001, membros proeminentes da comunidade se reuniram em Snowbird e adotaram o nome *métodos ágeis*, tendo publicado o Manifesto ágil, documento que reúne os princípios e práticas desta metodologia de desenvolvimento. Mais tarde, algumas pessoas formaram a *Agile Alliance*, uma organização não lucrativa que promove o desenvolvimento ágil.

Os métodos ágeis iniciais—criado a priore em 2000— incluíam Scrum (1986), Crystal Clear, Programação extrema (1996), Adaptive Software Development, Feature Driven Development, and Dynamic Systems Development Method (1995).

Comparações com outros métodos

Métodos Ágeis são algumas vezes caracterizados como o oposto de metodologias *guiadas pelo planejamento* ou *disciplinadas*. Uma distinção mais acurada é dizer que os métodos existem em um contínuo do *adaptativo* até o *preditivo*.[3] Métodos ágeis existem do lado adaptativo deste contínuo.

Métodos adaptativos buscam a adaptação rápida a mudanças da realidade. Quando uma necessidade de um projeto muda, uma equipe adaptativa mudará também. Um time adaptativo terá dificuldade em descrever o que irá acontecer no futuro. O que acontecerá em uma data futura é um item de difícil predição para um método adaptativo. Uma equipe adaptativa pode relatar quais tarefas se iniciarão na próxima semana. Quando perguntado acerca de uma implantação que ocorrerá daqui a seis meses, uma equipe adaptativa deve ser capaz somente de relatar a instrução de missão para a implantação, ou uma expectativa de valor versus custo.

Métodos preditivos, em contraste, colocam o planejamento do futuro em detalhe. Uma equipe preditiva pode reportar exatamente quais aspectos e tarefas estão planejados para toda a linha do processo de desenvolvimento. Elas porém tem dificuldades de mudar de direção. O plano é tipicamente otimizado para o objetivo original e mudanças de direção podem causar a perda de todo o trabalho e determinar que seja feito tudo novamente. Equipes preditivas freqüentemente instituem um comitê de controle de mudança para assegurar que somente as mudanças mais importantes sejam consideradas.

Métodos ágeis têm muito em comum com técnicas de *Desenvolvimento rápido de aplicação* de 1980 como exposto por James Martin e outros.

Comparação com o desenvolvimento iterativo

A maioria dos métodos ágeis compartilha a ênfase no Desenvolvimento iterativo e incremental para a construção de versões implantadas do software em curtos períodos de tempo. Métodos ágeis diferem dos métodos iterativos porque seus períodos de tempo são medidos em semanas, ao invés de meses, e a realização é efetuada de uma maneira altamente colaborativa. estendendo-se a tudo.

Comparação com o modelo em cascata

O desenvolvimento ágil tem pouco em comum com o modelo em cascata. Na visão de alguns este modelo é desacreditado, apesar de ser um modelo de uso comum. O modelo em cascata é uma das metodologias com maior ênfase no planejamento, seguindo seus passos através da captura dos requisitos, análise, projeto, codificação e testes em uma sequência pré-planejada e restrita. O progresso é geralmente medido em termos de entrega de artefatos—especificação de requisitos, documentos de projeto, planos de teste, revisão do código, e outros. O modelo em cascata resulta em uma substancial integração e esforço de teste para alcançar o fim do ciclo de vida, um período que tipicamente se estende por vários meses ou anos. O tamanho e dificuldade deste esforço de integração e teste é uma das causas das falhas do projeto em cascata. Métodos ágeis, pelo contrário, produzem um desenvolvimento completo e teste de aspectos (mas um pequeno subconjunto do todo) num período de poucas semanas ou meses. Enfatiza a obtenção de pequenos pedaços de funcionalidades executáveis para agregar valor ao negócio cedo, e continuamente agregar novas funcionalidades através do ciclo de vida do projeto.

Algumas equipes ágeis usam o modelo em cascata em pequena escala, repetindo o ciclo de cascata inteiro em cada iteração. Outras equipes, mais especificamente as equipes de Programação extrema, trabalham com atividades simultaneamente.

Comparação com a "codificação cowboy"

A codificação cowboy, também chamada de Modelo Balbúrdia, é a ausência de metodologias de desenvolvimento de Software: os membros da equipe fazem o que eles sentem que é correto. Como os desenvolvedores que utilizam métodos ágeis freqüentemente reavaliam os planos, enfatizam a comunicação face a face e fazem o uso relativamente esparso de documentos, ocasionalmente levam as pessoas a confundirem isto com codificação *cowboy*. Equipes ágeis, contudo, seguem o processo definido (e freqüentemente de forma disciplinada e rigorosa).

Como em todas as metodologias, o conhecimento e a experiência dos usuários definem o grau de sucesso e/ou fracasso de cada atividade. Os controles mais rígidos e sistematizados aplicados em um processo implicam altos níveis de responsabilidade para os usuários. A degradação de procedimentos bem-intencionados e organizados pode levar as atividades a serem caracterizadas como codificação *cowboy*.

Aplicabilidade dos métodos ágeis

Embora os métodos ágeis apresentem diferenças entre suas práticas, eles compartilham inúmeras características em comum, incluindo o desenvolvimento iterativo, e um foco na comunicação interativa e na redução do esforço empregado em artefatos intermediários. (Cohen et al., 2004)[4] A aplicabilidade dos métodos ágeis em geral pode ser examinada de múltiplas perspectivas. Da perspectiva do produto, métodos ágeis são mais adequados quando os requisitos estão emergindo e mudando rapidamente, embora não exista um consenso completo neste ponto (Cohen et al., 2004).[4] De uma perspectiva organizacional, a aplicabilidade pode ser expressa examinando três dimensões chaves da organização: cultura, pessoal e comunicação. Em relação a estas áreas inúmeros fatores chave do sucesso podem ser identificados (Cohen et al., 2004):[4]

- A cultura da organização deve apoiar a negociação.
- As pessoas devem ser confiantes.
- Poucas pessoas, mas competentes.
- A organização deve promover as decisões que os desenvolvedores tomam.
- A Organização necessita ter um ambiente que facilite a rápida comunicação entre os membros.

O fator mais importante é provavelmente o tamanho do projeto (Cohen et al., 2004)..[4] Com o aumento do tamanho, a comunicação face a face se torna mais difícil. Portanto, métodos ágeis são mais adequados para projetos com pequenos times, com no máximo de 20 a 40 pessoas.

De forma a determinar a aplicabilidade de métodos ágeis específicos, uma análise mais sofisticada é necessária. O método dinâmico para o desenvolvimento de sistemas, por exemplo, provê o denominado 'filtro de aplicabilidade' para este propósito. Também, a família de métodos Crystal provê uma caracterização de quando selecionar o método para um projeto. A seleção é baseada no tamanho do projeto, criticidade e prioridade. Contudo, outros métodos ágeis não fornecem um instrumento explícito para definir sua aplicabilidade a um projeto.

Alguns métodos ágeis, como DSDM (Dynamic Systems Development Method) e FDM (Feature Driven Development), afirmam se aplicar a qualquer projeto de desenvolvimento ágil, sem importar suas características (Abrahamsonn et al., 2003).[5]

A comparação dos métodos ágeis irá revelar que eles suportam diferentes fases de um ciclo de vida do software em diferentes níveis. Estas características individuais dos métodos ágeis podem ser usadas como um critério de seleção de sua aplicabilidade.

Desenvolvimentos ágeis vêm sendo amplamente documentados (ver Experiências relatadas, abaixo, como também em Beck,[6] e Boehm & Turner[7]) como funcionando bem para equipes pequenas (< 10 desenvolvedores). O desenvolvimento ágil é particularmente adequado para equipes que têm que lidar com mudanças rápidas ou imprevisíveis nos requisitos.

A aplicabilidade do desenvolvimento ágil para os seguintes cenários é ainda uma questão aberta:

- esforços de desenvolvimento em larga escala (> 20 desenvolvedores), embora estratégias para maiores escalas tenham sido descritas.[8]
- esforços de desenvolvimento distribuído (equipes não co-alocadas). Estas estratégias tem sido descritas em *Bridging the Distance*[9] e *Using an Agile Software Process with Offshore Development*[10]
- esforços críticos de missão e vida.
- Companhias com uma cultura de comando e controle.

Barry Boehm e Richard Turner sugeriram que análise de risco pode ser usada para escolher entre métodos adaptativos ("ágeis") e preditivos ("dirigidos pelo planejamento").[7] Os autores sugerem que cada lado deste contínuo possui seu *ambiente ideal"*

Ambiente ideal para o desenvolvimento ágil:

- Baixa criticidade
- Desenvolvedores senior
- Mudanças frequente de requisitos
- Pequeno número de desenvolvedores
- Cultura que tem sucesso no caos.

Ambiente ideal para o desenvolvimento direcionado ao planejamento:

- Alta criticidade
- Desenvolvedores Junior
- Baixa mudança nos requisitos
- Grande número de desenvolvedores
- Cultura que procura a ordem.

Adaptabilidade dos métodos ágeis

Um método deve ser bastante flexível para permitir ajustes durante a execução do projeto. Há três problemas chaves relacionados ao tópico de adaptação dos métodos ágeis: **a aplicabilidade dos métodos ágeis** (no geral e no particular), e finalmente, o **suporte ao gerenciamento de projeto**.

Na Agile Culture (https://www.cronapp.io/pt-br/agile-culture-o-que-e-e-quais-as-vantagens-para-a-sua-empresa/), o profissional tem mais liberdade. Ele recebe um conjunto de métricas, objetivos e orientações para planejar o seu trabalho da melhor forma possível, agregando valor ao negócio e dando mais flexibilidade interna.

Métodos ágeis e o gerenciamento de projeto

Os métodos ágeis diferem largamente no que diz respeito a forma de serem gerenciados. Alguns métodos são suplementados com guias para direcionar o gerenciamento do projeto, mas nem todos são aplicáveis.[5]

PRINCE2™ tem sido considerado como um sistema de gerenciamento de projeto complementar e adequado.[11]

Uma característica comum dos processos ágeis é a capacidade de funcionar em ambientes muito exigentes que tem um grande número de incertezas e flutuações (mudanças) que podem vir de várias fontes como: equipe em processo de formação que ainda não trabalhou junto em outros projetos, requisitos voláteis, baixo conhecimento do domínio de negócio pela equipe, adoção de novas tecnologias, novas ferramentas, mudanças muito bruscas e rápidas no ambiente de negócios das empresas: novos concorrentes, novos produtos, novos modelos de negócio.

Sistemas de gerenciamento de projetos lineares e prescritivos, neste tipo de ambiente, falham em oferecer as características necessárias para responder de forma ágil as mudanças requeridas. Sua adoção pode incrementar desnecessariamente os riscos, o custo, o prazo e baixar a qualidade do produto gerado, desgastando a equipe e todos os envolvidos no processo.

A abordagem Scrum, para gestão de projetos ágeis, leva em consideração planejamento não linear, porém de maneira mais exaustiva e está focada em agregar valor para o cliente e em gerenciar os riscos, fornecendo um ambiente seguro. Pode ser utilizada na gestão do projeto aliada a uma metodologia de desenvolvimento como Programação Extrema, FDD, OpenUP, DSDM, Crystal ou outras.

Metodologia

- Programação extrema

Processo

- Scrum

Albert Joseph; Ercilia Chilaule; Francelino Itc(I2cv)

- Feature Driven Development (FDD)
- Dynamic Systems Development Method (DSDM)
- Adaptive Software Development
- Crystal
- Pragmatic Programming
- **Test Driven Development** (http://www.improveit.com.br/xp/praticas/tdd) (em português)

Críticas

O método de desenvolvimento ágil é algumas vezes criticado como codificação cowboy. O início da Programação extrema soava como controverso e dogmático, tal como a programação por pares e o projeto contínuo, tem sido alvo particular de críticos, tais como McBreen[12] e Boehm e Turner.[7] Contudo, muitas destas críticas têm sido vistas pelos defensores dos métodos ágeis como mal entendidos a respeito do desenvolvimento ágil.[13]

Em particular, a Programação extrema é revista e criticada por Matt Stephens' Extreme Programming Refactored.[14]

As críticas incluem

- falta de estrutura e documentação necessárias
- somente trabalhar com desenvolvedores de nível sênior
- incorpora de forma insuficiente o projeto de software
- requer a adoção de muita mudança cultural

- poder levar a maiores dificuldades nas negociações contratuais

Referências

1. «Manifesto for Agile Software Development» (http://www.agilemanifesto.org/). *www.agilemanifesto.org*. Consultado em 23 de outubro de 2015
2. «Principles behind the Agile Manifesto» (http://www.agilemanifesto.org/principles.html). *www.agilemanifesto.org*. Consultado em 23 de outubro de 2015
3. B. Boehm (2004). *Balancing Agility and Discipline: A Guide for the Perplexed* 2 ed. Boston,MA: Addison-Wesley. pp. 165–194. ISBN 0-321-18612-5
4. Cohen, D., Lindvall, M., & Costa, P. (2004). An introduction to agile methods. In *Advances in Computers* (pp. 1-66). New York: Elsevier Science.
5. Abrahamsson, P., Warsta, J., Siponen, M.T., & Ronkainen, J. (2003). New Directions on Agile Methods: A Comparative Analysis. *Proceedings of ICSE'03*, 244-254
6. K. Beck (1999). *Extreme Programming Explained: Embrace Change*. Boston, MA: Addison-Wesley. 157 páginas. ISBN 0-321-27865-8
7. B. Boehm (2004). *Balancing Agility and Discipline: A Guide for the Perplexed*. Boston, MA: Addison-Wesley. pp. 55–57. ISBN 0-321-18612-5
8. «Supersize Me» (http://www.sdmagazine.com/documents/s=10020/sdm0603g/0603g.html)
9. «Bridging the Distance» (http://www.sdmagazine.com/documents/s=7556/sdm0209i/sdm0209i.htm)
10. «Using an Agile Software Process with Offshore Development» (http://www.martinfowler.com/articles/agileOffshore.html)
11. Agile Alliance at http://agilealliancebeta.org/article/file/904/file.pdf (http://agilealliancebeta.org/article/file/904/file.pdf):
12. P. McBreen (2003). *Questioning Extreme Programming*. Boston, MA: Addison-Wesley. ISBN 0-201-84457-5
13. «sdmagazine» (http://www.sdmagazine.com/documents/s=1811/sdm0112h/0112h.htm)
14. «Extreme Programming Refactored» (http://www.softwarereality.com/ExtremeProgrammingRefactored.jsp)

Futuras leituras

- Fowler, Martin. *Is Design Dead?* (http://www.martinfowler.com/articles/designDead.html). Appeared in *Extreme Programming Explained*, G. Succi and M. Marchesi, ed., Addison-Wesley, Boston. 2001.
- Riehle, Dirk. *A Comparison of the Value Systems of Adaptive Software Development and Extreme Programming: How Methodologies May Learn From Each Other* (http://www.riehle.org/computer-science/research/2000/xp-2000.html). Appeared in *Extreme Programming Explained*, G. Succi and M. Marchesi, ed., Addison-Wesley, Boston. 2001.
- Tomek, Ivan. *What I Learned Teaching XP*. http://www.whysmalltalk.com/articles/tomek/teachingxp.htm (http://www.whysmalltalk.com/articles/tomek/teachingxp.htm)
- M. Stephens, D. Rosenberg. *Extreme Programming Refactored: The Case Against XP*. Apress L.P., Berkeley, California. 2003. (ISBN 1-59059-096-1)
- D. Rosenberg, M. Stephens. *Agile Development with ICONIX Process*. Apress L.P., Berkeley, California. 2005. (ISBN 1-59059-464-9)
- Beck, et. al., *Manifesto for Agile Software Development*. [2] (http://www.agilemanifesto.org/)
- Larman, Craig and Basili, Victor R. *Iterative and Incremental Development:A Brief History* IEEE Computer, June 2003 (http://www2.umassd.edu/SWPI/xp/articles/r6047.pdf)
- Abrahamsson, P., Warsta, J., Siponen, M.T., & Ronkainen, J. (2003). New Directions on Agile Methods: A Comparative Analysis. *Proceedings of ICSE'03*, 244-254.
- Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). Agile Software Development Methods: Review and Analysis. *VTT Publications 478*.
- Aydin, M.N., Harmsen, F., Slooten, K. v., & Stagwee, R. A. (2004). An Agile Information Systems Development Method in use. *Turk J Elec Engin, 12(2),* 127-138
- Aydin, M.N., Harmsen, F., Slooten van K., & Stegwee, R.A. (2005). On the Adaptation of An Agile Information Systems Development Method. *Journal of Database Management Special issue on Agile Analysis, Design, and Implementation, 16(4),* 20-24
- Cohen, D., Lindvall, M., & Costa, P. (2004). An introduction to agile methods. In *Advances in Computers* (pp. 1–66). New York: Elsevier Science.
- Karlstrom, D., & Runeson P. (2005). Combining agile methods with stage-gate project management. *IEEE Software, 22(3),* 43-49

Ligações externas

- Manifesto para Desenvolvimento Ágil de Software (http://www.agilemanifesto.org/) (em inglês)
- Agile Alliance Brasil (http://www.agilealliance.com.br) (em português)
- Cronapp - Freedom to Develop (https://www.cronapp.io/pt-br/cronapp/) (em português)