UNINOVE

Manifesto ágil e as metodologias ágeis.

O PRESENTE TÓPICO TEM POR OBJETIVO VIABILIZAR O CONTATO DO ALUNO COM OS CONCEITOS E COM A APLICAÇÃO DOS MÉTODOS DE DESENVOLVIMENTO ÁGIL DE SOFTWARE, A FIM DA CONSTRUÇÃO DE COMPETÊNCIAS INERENTES AO TEMA EM QUESTÃO.

AUTOR(A): PROF. ANDRE RONALDO RIVAS

O Manifesto Ágil

Durante as décadas de 80 e 90 era frequente a opinião de que o processo de construção de software deveria seguir um rígido planejamento delimitado por fases específicas, devidamente auditadas e documentadas. Este cenário, mostrava-se aderente ao propósito do exercício dos métodos de análise vigentes e até então sustentados pelas ferramentas do momento, a maioria destas conhecidas como "ferramentas CASE".

Esta compreensão do contexto de desenvolvimento de software foi fortemente influenciada pela comunidade mais representativa neste tipo de atividade durante aquele período, faço referência à comunidade responsável por desenvolver softwares para grandes empresas ou setores com certa ênfase em pesquisas como é o caso das áreas de Engenharia, Saúde, Governo e Militar. Há de se considerar que este panorama contemplava softwares mantidos por décadas e as equipes eram mais estáveis quanto aos seus vínculos trabalhistas.

Estas abordagens de desenvolvimento cuja orientação ao planejamento é mais enfática, atribui um maior peso às atividades de controle e documentação para as alterações nos sistemas, a justificativa para tanto é o número de pessoas diferentes que lidam com a sustentação do sistema que por vezes é caracterizado como sendo de missão crítica, ou seja, suporta algum processo de negócio importante da empresa.

Ponderando acerca das características das grandes instituições, não há de modo mais imediato grandes problemas nesta forte orientação ao planejamento; no entanto, considerando empresas mais modestas em seu porte, ao utilizar similar abordagem, o tempo a ser gasto para descrever como um software deverá ser desenvolvido poderá ser maior que o tempo relacionado às atividades de implementação e testes. Se levarmos em conta ainda as alterações de requisitos que certamente ocorrerão neste cenário, o desconforto com o peso de todo o trabalho torna-se inevitável. Em 2001, endereçar estes desafios para a produção de software era um propósito amplamente discutido em eventos pelo mundo, culminando na consolidação do chamado Manifesto Ágil por um grupo de 17 especialistas em processos de software, representando os métodos ágeis de maior interesse na ocasião.

OS 12 PRINCíPIOS DO MANIFESTO ÁGIL:

1. Maior prioridade na satisfação do cliente por meio da entrega antecipada e contínua de um valioso software.
2. As mudanças de requisitos são bem-vindas, mesmo próximas do final do desenvolvimento. Processos ágeis utilizam a mudança como uma vantagem competitiva do cliente.
3. Entregar software funcionando com frequência, a partir de algumas semanas a alguns meses, com preferência para os prazos mais curtos.
4. Colaboradores/usuários e desenvolvedores trabalham juntos diariamente durante todo o projeto.
5. Construir projetos em torno de indivíduos motivados. Proporcione o ambiente e apoio que precisam e confie neles para fazer o trabalho.
6. O método mais eficiente e eficaz de transmitir informações para e dentro de uma equipe de desenvolvimento é a conversa cara a cara.
7. Software funcionando é o principal parâmetro de progresso.
8. Processos ágeis promovem o desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante.
9. Atenção contínua a excelência técnica e um bom projeto aumentam a agilidade.
10. Simplicidade - a arte de maximizar a quantidade de trabalho não realizado é essencial.
11. As melhores arquiteturas, requisitos e projetos emergem de equipes auto-organizadas.
12. Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz, então sintoniza e ajusta seu comportamento de modo adequado.

Os 12 princípios do Manifesto Ágil podem ser traduzidos nos valores a seguir, adotados por todas as metodologias ágeis:

- Indivíduos e interações em vez de processos e ferramentas.
- Software funcionando em vez de documentação abrangente.
- Colaboração do cliente em vez de negociação de contratos.
- Responder a mudanças em vez de seguir um plano.

A alcunha ágil relacionada ao desenvolvimento de software tornou-se evidente pela primeira vez em um artigo publicado durante os anos 90 em que métodos "ágeis" eram recomendados para a construção de softwares no Japão (AOYAMA, 1998; MILLER, 2001). Na ocasião, os japoneses também faziam parte do grupo de interesse de discussão do tema, muito em razão das pressões competitivas presentes em seu contexto corporativo, sendo assim, buscavam a redução do tempo para levar um produto ao mercado, cabe afirmar que esta é uma das características mais importantes a ser tratada para agilizar a execução dos processos de software. Atualmente, muito do que se pratica no mundo em cultura ágil tem origem Japão, os princípios da produção Lean (enxuta) por exemplo, estes adaptados para diversos métodos ágeis (vide o Scrum).

Algumas das principais Metodologias Ágeis encontram-se a seguir:

Extreme Programming (XP)

A Programação Extrema (Extreme Programming) ou XP foi criada por Kent Beck em 1996 enquanto ele era gerente de projetos na Chrysler, divisão Comprehensive Compensation System - Sistema de Compensação Abrangente da Chrysler - (C3), onde desenvolvia um projeto de longa duração com o objetivo de reescrever a aplicação de folha de pagamento da Chrysler.

É considerada uma metodologia leve, eficiente, de baixo risco, flexível, previsível, científica e simples de desenvolver software. XP é considerada uma disciplina de desenvolvimento de software, pois para se considerar como um praticante de XP, deve-se seguir suas diretrizes, tais como escrever testes de software.

O Quadro Resumo da Metodologia XP expõe as principais características do modelo.

PRINCÍPIO OU PRÁTICA	DESCRIÇÃO
Planejamento incremental	Os requisitos são registrados em cartões de estória e as estórias a serem incluídas são determinadas pelo tempo disponível e sua prioridade relativa. Os desenvolvedores quebram essas estórias em Tarefas do Desenvolvimento.
Pequenas versões	Inicialmente, desenvolve-se um conjunto mínimo, útil, que já agrega valor aos negócios. Depois, frequentemente se lançam versões do sistema que gradualmente adicionam funcionalidades à primeira versão.
Projeto de *software* simples	Um projeto é definido de modo a atender aos requisitos atuais e nada mais.
Testar primeiro	Sistemas que automatizam o teste de unidades são utilizados para registrar os testes que serão exercidos em uma nova funcionalidade, antes que a funcionalidade em si seja implementada.
Refatoração	A refatoração de código é uma atividade que todos os desenvolvedores deverão executar, de forma contínua. Isso mantém o código simples e sustentável.
Programação em pares	Os desenvolvedores trabalham em pares, um verificando o trabalho do outro e proporcionando apoio para realizar o trabalho com qualidade.
Propriedade coletiva	Os pares de desenvolvedores trabalham em todas as áreas do sistema, de modo a não formar ilhas de especialização. Todos os desenvolvedores devem assumir a responsabilidade por todo o código e qualquer desenvolvedor pode alterar qualquer código.
Integração contínua	Assim que o trabalho em uma tarefa for concluído, ele é integrado ao todo do sistema. Depois de tal integração, todos os testes de unidade são refeitos.
Ritmo sustentável	Trabalha-se exatamente a quantidade de horas normais. Grandes quantidades de horas extras não são considerados aceitáveis, o efeito líquido é muitas vezes a redução da qualidade do código e da produtividade a médio prazo.
Cliente no local	Um representante do usuário final do sistema (o cliente) deve estar disponível o tempo todo com as equipes XP. Em um processo de programação extrema, o cliente é um membro da equipe de desenvolvimento e é responsável por trazer os requisitos do sistema para a equipe de implementação.

Legenda: QUADRO RESUMO - METODOLOGIA EXTREME PROGRAMMING (XP)

O Extreme Programming (XP), criado por Kent Beck e Ward Cunningha, em 1996, orienta-se para práticas semelhantes ao Scrum, utilizando-se de ciclos contínuos de trabalho e interação com o cliente para gerar um produto que possua a maior utilidade possível ao cliente.

AMORIM, H.; MAURÍCIO, R.G. TAYLORIZAçãO E AUTO-TAYLORIZAçãO DO TRABALHO: AS METODOLOGIAS áGEIS NA INDúSTRIA DE SOFTWARE. SECULO XXI, SANTA MARIA, V. 8, N. 2, 2018.

Scrum

O Scrum é uma abordagem empírica, que utiliza princípios da teoria de controle de processos com viés da indústria, adaptando-os para o desenvolvimento de software, resultando assim em um propósito que abrange ideias de flexibilidade, adaptabilidade, além de produtividade. O Scrum não se limita a nenhuma técnica específica para desenvolvimento de software considerando a implementação, concentrando-se sobretudo no modo como a equipe de desenvolvimento se comporta durante a produção do software, utilizando dos princípios da flexibilidade, transparência e inspeção. De um modo geral, o Scrum é uma metodologia ágil, mas seu interesse maior volta-se a gerenciar o desenvolvimento iterativo em vez de concentrar esforços nas abordagens técnicas específicas para a Engenharia de Software.

O principal conceito do Scrum é que o desenvolvimento de sistemas envolve muitas variáveis ambientais e técnicas (por exemplo, requisitos, unidades de tempo, recursos, tecnologia) e provavelmente sofrerão mudanças durante a execução do processo. Isso torna o processo de desenvolvimento imprevisível e complexo, requerendo flexibilidade do processo de desenvolvimento do sistema para ser capaz de responder a essas mudanças.

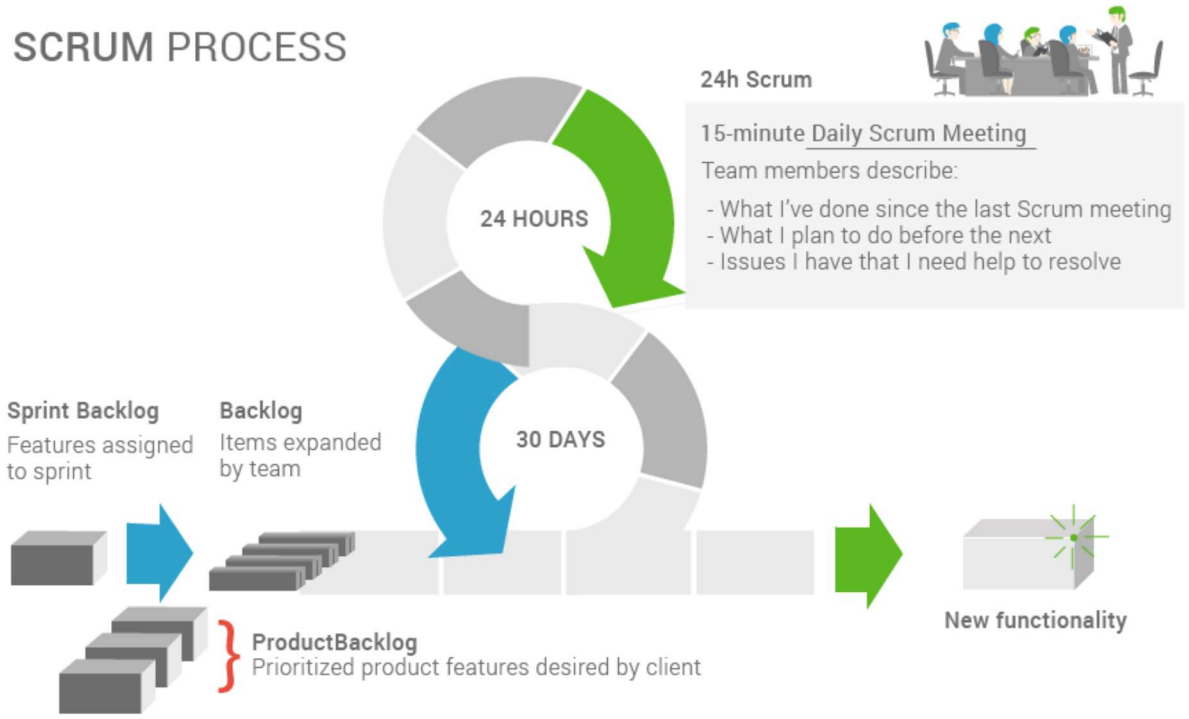
Um processo de desenvolvimento Scrum é marcado pela execução de três fases: planejamento, desenvolvimento e encerramento.

- No planejamento, se estabelecem os objetivos gerais para o projeto e projeta-se a arquitetura de software do sistema.
- O desenvolvimento é marcado pela execução de uma série de ciclos chamada Sprint. Cada ciclo desenvolve um incremento do sistema.
- A fase de encerramento termina o projeto, cria uma documentação completa e avalia as lições aprendidas com o projeto.
- Dentro de um ciclo Sprint, uma Scrum Sprint é uma unidade de planejamento em que o trabalho a ser feito é avaliado, os recursos são selecionados para o desenvolvimento, e o software é implementado. No final de um Sprint, a funcionalidade concluída é entregue às partes interessadas.
- Sprints são de comprimento fixo, e duram normalmente de 2 a 4 semanas. Eles correspondem ao desenvolvimento de uma versão do sistema em XP.
- O ponto de partida para o planejamento é o Product Backlog, que é a lista de trabalho a ser realizada no projeto. O cliente está intimamente envolvido neste processo e pode introduzir novos requisitos ou tarefas no início de cada Sprint.
- A fase de seleção de um Sprint envolve toda a equipe do projeto, que trabalha com o cliente para selecionar os recursos e funcionalidades a serem desenvolvidas durante o Sprint.
- Havendo um acordo, a equipe organiza-se para desenvolver o software. Reuniões diárias curtas envolvendo todos os membros da equipe são realizadas para avaliar o progresso e, se necessário,

priorizar o trabalho. Durante esta fase, a equipe está isolada do cliente e da organização, com todas as comunicações canalizadas através do Scrum Master.

- O papel do Scrum Master é proteger a equipe de desenvolvimento de distrações externas.
- No final do Sprint o trabalho é revisado e apresentado às partes interessadas. O próximo ciclo de Sprint começa na sequência.

A figura a seguir, ilustra esta dinâmica do Scrum.



Legenda: DINâMICA SCRUM

Crystal

A metodologia Crystal é, na realidade, uma família de metodologias que foram desenvolvidas em meados da década de 1990. As metodologias Crystal são consideradas e descritas como "metodologias leves".

As metodologias Crystal estão focadas em:

- pessoas;
- interação;
- comunidade;
- habilidades;
- talentos; e
- comunicações.

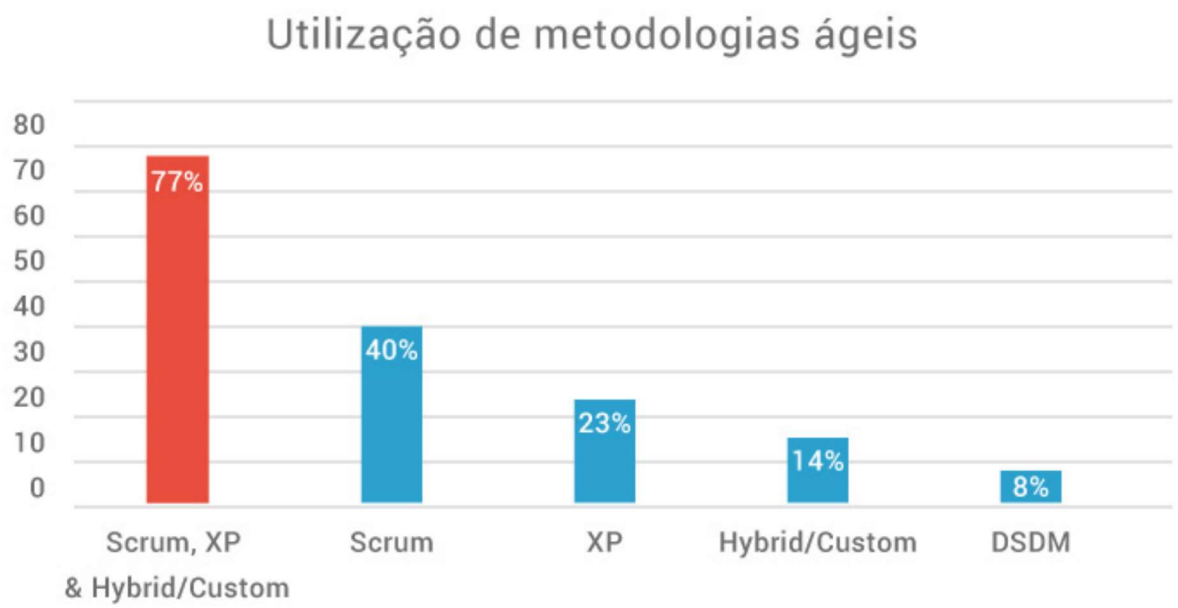
A ideia por trás das metodologias Crystal é que as equipes envolvidas no desenvolvimento de software normalmente têm conjuntos de habilidades e talentos variados e, assim, o elemento processo em si não é um fator importante.

Uma vez que as equipes podem executar tarefas semelhantes de formas diferentes, a família Crystal de metodologias é muito tolerante nesse sentido, o que faz com que na família Crystal seja uma das metodologias mais fáceis de aplicar. Além disso, ela possui metodologias que podem ser aplicadas em diversos tipos de projetos, de acordo com seu tamanho e criticidade, utilizando cores diferentes para designar o ?peso? da metodologia a usar. Quanto mais escura a cor, mais adequada é a metodologia para projetos críticos e de longa duração.

Desenvolvimento de Sistemas Dinâmicos - Dynamic Systems Development Method (DSDM)

A DSDM é uma das metodologias ágeis de desenvolvimento de software. Esta metodologia visa desenvolver uma aplicação com a qualidade desejada sem ultrapassar limites de tempo e orçamento. Para conseguir estes objetivos, a DSDM se concentra na interação com o cliente e o usuário final, entrega de protótipos frequentes, equipes de desenvolvimento autônomas e testes realizados durante todo o processo, baseando-se na definição de prioridades de requisitos fornecida pelo cliente.

O gráfico a seguir viabiliza uma ideia do quanto estes Métodos Ágeis são adotados nas empresas.



Legenda: UTILIZAçãO DAS METODOLOGIAS ÁGEIS

Além do Scrum, o Extreme Programming - XP e a Metodologia de Desenvolvimento de Sistemas Dinâmicos (DSDM) são os Métodos Ágeis encontrados com mais frequência em fábricas de software.

ROSES, L.K.; WINDMÖLLER, A.; DO CARMO, E.A. FAVORABILITY CONDITIONS IN THE ADOPTION OF AGILE METHOD PRACTICES FOR SOFTWARE DEVELOPMENT IN A PUBLIC BANKING/ FAVORABILIDADE NA ADOçãO DE PRáTICAS DE MéTODOS áGEIS NO DESENVOLVIMENTO DE SOFTWARE EM UM BANCO PUBLICO. JOURNAL OF INFORMATION SYSTEMS AND TECHNOLOGY MANAGEMENT

ATIVIDADE

De acordo com o Manifesto Ágil, o que é valorizado?

A. Responder às mudanças para satisfazer as necessidades do cliente.

B. Processos e ferramentas bem documentos acima de indivíduos e interações entre eles.

C. Negociação de contratos acima da colaboração com o cliente.

Os princípios ágeis de um modo geral priorizam a colaboração com o cliente em vez de negociação de contratos. Há outras opções mais adequadas para assinalar.

ATIVIDADE

Qual destas é uma afirmação do Manifesto Ágil?

A. Valorizamos a negociação de contratos acima da colaboração do cliente.

B. Valorizamos seguir um plano acima da reação à mudança.

C. Valorizamos processos e ferramentas acima dos indivíduos e da interação.

D. Valorizamos um software funcional acima de uma documentação abrangente.

ATIVIDADE

Qual dos seguintes NÃO é um princípio do Manifesto Ágil?

A. Software funcionando, entregue com frequência.

B. Cooperação próxima e diária entre usuários de negócio e desenvolvedores.

C. Reuniões diárias com a equipe para analisar os progressos e levantar impedimentos.

D. Projetos construídos em torno de indivíduos motivados, os quais precisam ganhar confiança.

REFERÊNCIA

1. ABBAS, N.; GRAVELL, A. M.; WILLS, G. B. Historical Roots of Agile Methods: Where Did - Agile Thinking - Come From - In: Agile Processes in Software Engineering and Extreme Programming. [s.l.] Springer, 2008. p. 94?103.
   AMORIM, H.; MAURÍCIO, R.G. Taylorização e auto-taylorização do trabalho: as metodologias ágeis na indústria de software. Seculo XXI, Santa Maria, v. 8, n. 2, 2018. Disponível em: https://search.proquest.com/docview/2163363358/abstract/ABB1B81F97894B0CPQ/7?accountid=43603 . Acesso em: 12 junho 2020.
2. COCKBURN, A.; WILLIAMS, L. Agile software development: It?s about feedback and change. In: Computer, v. 36, no 6, p. 39-43, 2003.
3. MANIFESTO, ÁGIL. Disponível em:< http://www. manifestoagil. com. br/>. Acesso em: 11 junho 2020.
4. PONTES, Reni Elisa da Silva; NETO, João Souza. Contratação do desenvolvimento ágil de software na Administração Pública Federal: riscos e ações mitigadoras. Revista do Serviço Publico, Brasilia, v. 66, n. 1, 2015. Disponível em: https://search.proquest.com/docview/1916489555/abstract/D9A36383589140E9PQ/4?accountid=43603 . Acesso em: 15 maio 2020.
5. PRESSMAN, R. S. Engenharia de Software: Uma abordagem profissional. 7. ed. São Paulo: McGraw Hill, 2011.
6. ROSES, L.K.; WINDMÖLLER, A.; DO CARMO, E.A. Favorability conditions in the adoption of agile method practices for software development in a public banking/ Favorabilidade na adoção de práticas de métodos ágeis no desenvolvimento de software em um banco publico. Journal of Information Systems

and Technology Management : JISTEM. São Paulo, v. 13, n. 3, setembro 2016. Disponível em: https://search.proquest.com/docview/1879601477/D9A36383589140E9PQ/3?accountid=43603 . Acesso em: 12 junho 2020.

7. SOMMERVILLE, I. Engenharia de Software. 9. ed. São Paulo: Pearson, 2011.