



Engenharia de Software II

Aula 03 – Desenvolvimento Ágil



Desenvolvimento Ágil

- Em 2001, Kent Beck e outros dezesseis renomados desenvolvedores, autores e consultores da área de software (batizados de “Agile Alliance”) assinaram o “Manifesto de Desenvolvimento Ágil de Software”, que se inicia da seguinte maneira:
 - Indivíduos e interações acima de processos de ferramentas
 - Software operacional acima de documentação completa
 - Colaboração dos clientes acima de negociação contratual
 - Respostas a mudanças acima de seguir um plano

Panorama

PANORAMA

O que é? A engenharia de software ágil combina filosofia com um conjunto de princípios de desenvolvimento. A filosofia defende a satisfação do cliente e a entrega de incremental prévio; equipes de projeto pequenas e altamente motivadas; métodos informais; artefato de engenharia de software mínimos e, acima de tudo, simplicidade no desenvolvimento geral. Os princípios de desenvolvimento priorizam a entrega mais que análise e projeto (embora essas atividades não sejam desencorajadas); também priorizam a comunicação ativa e contínua entre desenvolvedores e clientes.

Quem realiza? Os engenheiros de software e outros envolvidos no projeto (gerentes, clientes, usuários finais) trabalham conjuntamente em uma equipe ágil — uma equipe que se auto-organiza e que controla seu próprio destino. Uma equipe ágil acelera a comunicação e a colaboração entre todos os participantes (que estão a seu serviço).

Por que é importante? O moderno ambiente dos sistemas e dos produtos da área é acelerado e está em constante mudança. A engenharia de software ágil constitui uma razoável alternativa para a engenharia convencional voltada para certas

classes de software e para certos tipos de projetos, e tem se mostrado capaz de entregar sistemas corretos rapidamente.

Quais são as etapas envolvidas? O desenvolvimento ágil poderia ser mais bem denominado “engenharia de software flexível”. As atividades metodológicas básicas permanecem: comunicação, planejamento, modelagem, construção e emprego. Entretanto, estas se transformam em um conjunto de tarefas mínimas que impulsiona a equipe para o desenvolvimento e para a entrega (pode-se levantar a questão de que isso é feito em detrimento da análise do problema e do projeto de soluções).

Qual é o artefato? Tanto o cliente como o engenheiro têm o mesmo parecer: o único artefato realmente importante consiste em um “incremento de software” operacional que seja entregue, adequadamente, na data combinada.

Como garantir que o trabalho foi realizado corretamente? Se a equipe ágil concordar que o processo funciona e essa equipe produz incrementos de software passíveis de entrega e que satisfaçam o cliente, então, o trabalho está correto.



O que é Agilidade?

- No contexto da engenharia de software, o que é agilidade? Ivar Jacobson apresenta uma útil discussão:
- “Atualmente, agilidade tornou-se a palavra da moda quando se descreve um moderno processo de software. Todo mundo é ágil. Uma equipe ágil é aquela rápida e capaz de responder apropriadamente a mudanças. Mudanças têm muito a ver com o desenvolvimento de software. Mudanças no software está sendo criado, mudanças nos membros da equipe, mudanças devido a novas tecnologias, mudanças de todos os tipos que poderão ter um impacto no produto que está em construção ou no projeto que cria o produto.”



Continuando...

- ▶ Segundo Jacobson, a penetração da mudança é o principal condutor para a agilidade.
- ▶ A agilidade pode ser aplicada a qualquer processo de software. Entretanto, para obtê-la, é essencial que seja projetado para que a equipe possa adaptar e alinhar (racionalizar) tarefas; possa conduzir o planejamento compreendendo a fluidez de uma abordagem do desenvolvimento ágil.
- ▶ Sempre enfatizar a estratégia de entrega incremental, conseguindo entregar ao cliente, o mais rapidamente possível, o software operacional para o tipo de produto e ambiente operacional.

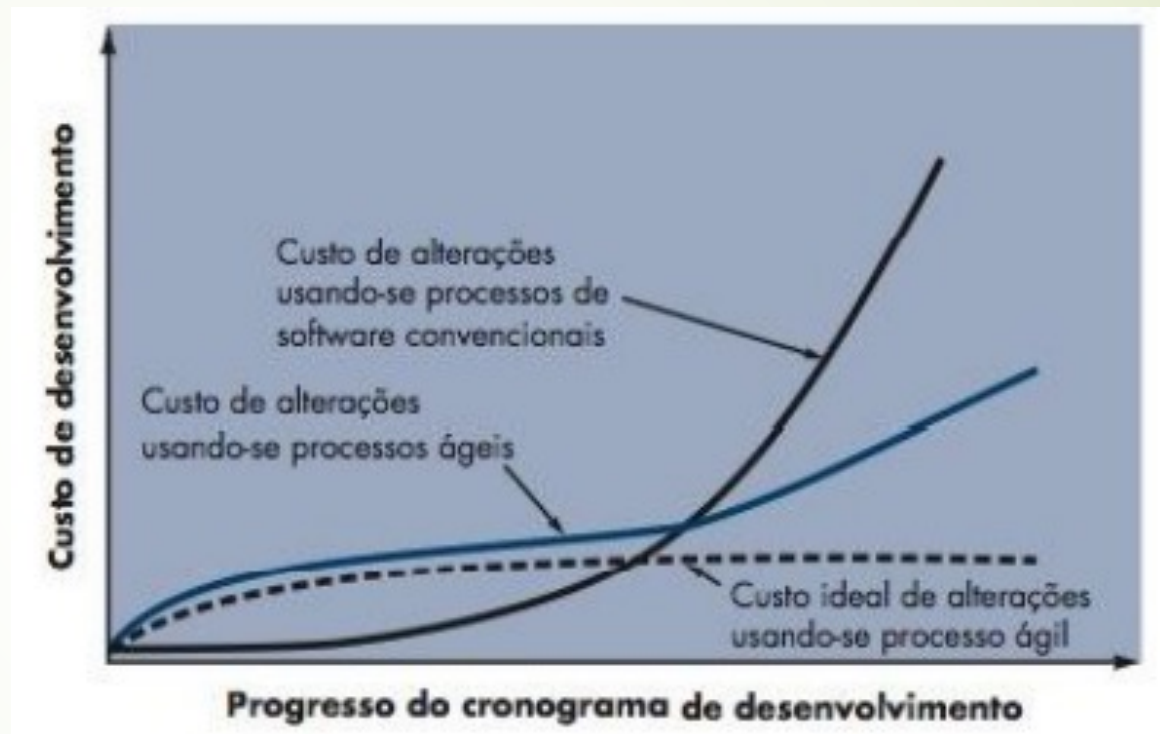


Agilidade e Custo das Mudanças

- A sabedoria convencional em desenvolvimento de software afirma que os custos de mudanças aumentam de forma não linear conforme o projeto avança.
- É relativamente fácil assimilar uma mudança quando uma equipe de software está juntando requisitos. Pode-se ter de alterar um detalhamento do uso, ampliar uma lista de funções ou editar uma especificação por escrito.
- Os custos crescem rapidamente e não serão triviais o tempo e custos necessários para assegurar que a mudança seja feita sem efeitos colaterais inesperados.

Custos de Alteração

- Os defensores da agilidade, argumentam que um processo ágil bem elaborado “achata” o custo da curva de mudança.
- Há evidências que sugerem que se pode alcançar redução significativa nos custos de alterações, embora haja um debate contínuo sobre qual o nível em que a curva de custos torna-se “achatada”.



O que é Processo Ágil?

- Dado esses três preceitos, surge uma importante questão: como criar um processo capaz de administrar e imprevisibilidade?
- A resposta, conforme já observado, consiste na adaptabilidade de processo. Portanto, um processo ágil deve ser adaptável.

1. É difícil afirmar antecipadamente quais requisitos de software irão persistir e quais sofrerão alterações. É igualmente difícil prever de que maneira as prioridades do cliente sofrerão alterações conforme o projeto avança.
2. Para muitos tipos de software, o projeto e a construção são "interconduzidos". Ou seja, ambas as atividades devem ser realizadas em sequência (uma atrás da outra), para que os modelos de projeto sejam provados conforme sejam criados. É difícil prever quanto de trabalho de projeto será necessário antes que a sua construção (desenvolvimento) seja implementada para avaliar o projeto.
3. Análise, projeto, construção (desenvolvimento) e testes não são tão previsíveis (do ponto de vista de planejamento) quanto gostaríamos que fosse.



Princípios da Agilidade

- A maior prioridade é satisfazer o cliente por meio de entrega adiantada e contínua de software valioso.
- Acolha bem os pedidos de alterações, mesmo atrasados no desenvolvimento. Os processos ágeis se aproveitam das mudanças como uma vantagem competitiva na relação com o cliente.
- Entregue software em funcionamento frequentemente, de algumas semanas par alguns meses, dando preferência a intervalos mais curtos.
- O pessoal comercial e os desenvolvedores devem trabalhar em conjunto diariamente ao longo de todo o projeto.
- Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e apoio necessários e confie neles para ter o trabalho feito.
- O método mais efetivo de transmitir informações para e dentro de uma equipe de desenvolvimento é uma conversa aberta, de forma presencial.
- Software em um funcionamento é a principal medida do progresso.



Continuando...

- Os processos ágeis promovem desenvolvimento sustentável.
- Atenção contínua para com a excelência técnicas e para com bons projetos aumenta a agilidade.
- Simplicidade – a arte de maximizar o volume de trabalho não efetuado, é essencial.
- As melhores arquiteturas, requisitos e projetos emergem de equipes que se auto-organizam.
- A intervalos regulares, a equipe se avalia para ver como torna-se mais eficiente, então sintoniza e ajusta seu comportamento de acordo.



A Política do Desenvolvimento Ágil

- Há debates consideráveis sobre os benefícios e a aplicabilidade do desenvolvimento de software ágil em contraposição a processos de engenharia de software mais convencionais.
- Ninguém é contra a agilidade. A verdade questão é: Qual a melhor maneira de atingi-la? Igualmente importante, como desenvolver software que atenda às necessidades atuais dos clientes e que apresente características de qualidade que permitirão que seja estendido e ampliado para responder às necessidades dos clientes no longo prazo?
- Conclusão: pode-se ganhar muito considerando o que há de melhor nas duas escolas e praticamente nada denegando uma ou outra abordagem.



Fatores Humanos

- Os defensores do desenvolvimento de software ágil se esmeram para enfatizar a importância dos fatores humanos.
- **Competência:** no contexto do desenvolvimento ágil, a competência abrange talento inato, habilidades específicas relacionadas a software e conhecimento generalizado do processo que a equipe escolheu para aplicar.
- **Foco comum:** embora os membros de uma equipe ágil possam realizar diferentes tarefas e tragam diferentes habilidades para o projeto, todos devem estar focados em um único objetivo – entregar um incremento de software funcionando ao cliente, dentro do prazo prometido.



Continuando...

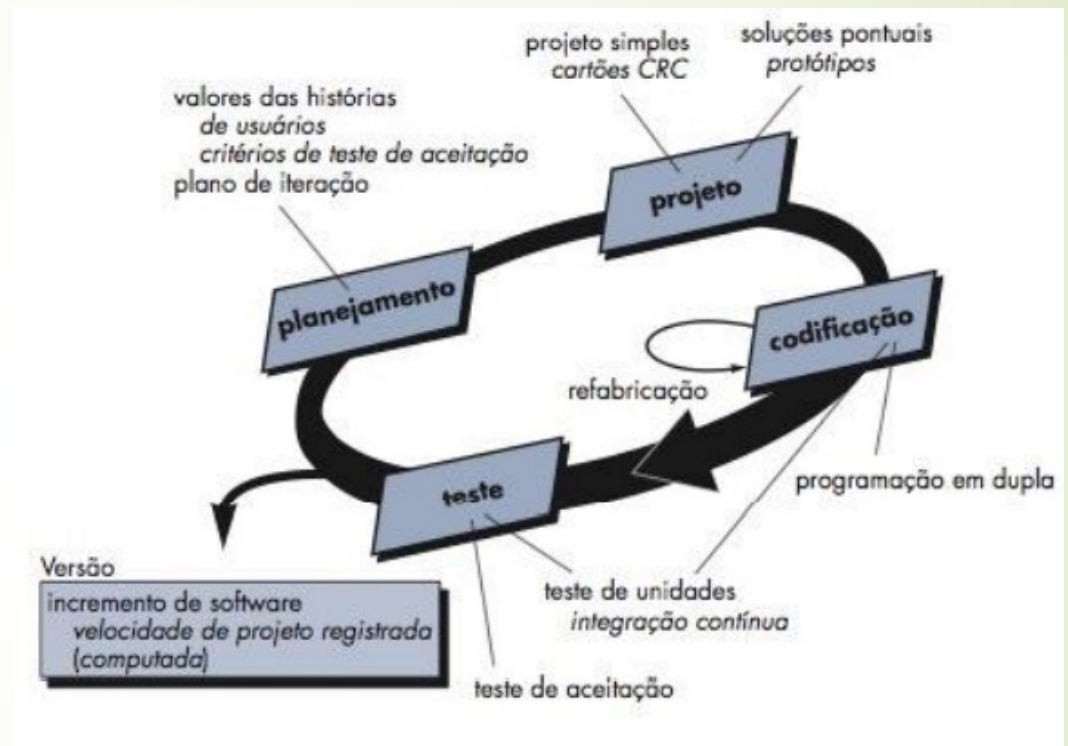
- **Colaboração:** trata de avaliação, análise e uso de informações comunicadas à equipe de software; criar informações que ajudarão todos os envolvidos a compreender o trabalho da equipe e a construir informações que forneçam valor do negócio para o cliente.
- **Tomada de Decisão:** qualquer boa equipe deve ter liberdade para controlar seu próprio destino.
- **Habilidade de solução de problemas confusos:** os gerentes de software devem reconhecer que a equipe ágil terá que lidar continuamente com a ambiguidade e que será continuamente atingida por mudanças
- **Confiança mútua e respeito:** uma equipe consistente demonstra a confiança e o respeito necessário para torna-la forte.
- **Auto-organização:** implica três fatores: (1) a equipe ágil se organiza para o trabalho ser feito, (2) a equipe organiza o processo para melhor se adequar ao seu ambiente, (3) a equipe organiza o cronograma de trabalho para melhor cumprir a entrega do incremento de software.



Extreme Programming - XP

- ▶ Para ilustrar um processo ágil de forma um pouco mais detalhada, segue uma visão geral de XP, abordagem mais amplamente utilizada para desenvolvimento de software ágil.
- ▶ Um conjunto de cinco valores que estabelecem as bases para todo trabalho realizado como parte da XP – comunicação, simplicidade, feedback, coragem e respeito.

O Processo XP



O Debate XP

- Todos os novos métodos e modelos de processo estimulam debates úteis e, em algumas casos, debates acalorados. A XP provocou ambos.

- *Volatilidade de requisitos.* Pelo fato de o cliente ser um membro ativo da equipe XP, alterações de requisitos são solicitadas informalmente. Como consequência, o escopo do projeto pode mudar e trabalhos anteriores podem ter de vir a ser alterados, a fim de acomodar as necessidades de então. Seus defensores argumentam que isso acontece independentemente do processo aplicado e que a XP oferece mecanismos para controlar o surgimento incontrolado de novos escopos.
- *Necessidades conflitantes de clientes.* Projetos em quantidade possuem múltiplos clientes, cada um com seu próprio conjunto de necessidades. Na XP, a própria equipe tem a tarefa de assimilar as necessidades de diferentes clientes, um trabalho que pode estar além de seu escopo de autoridade.
- *Os requisitos são levantados informalmente.* Histórias de usuários e testes de aceitação são a única manifestação explícita de requisitos da XP. Seus críticos argumentam que, frequentemente, torna-se necessário um modelo ou especificação mais formal para assegurar que omissões, inconsistências e erros sejam descobertos antes que o sistema seja construído. Seus defensores rebatem dizendo que a natureza mutante de requisitos torna tais modelos e especificações obsoletos praticamente logo depois de terem sido desenvolvidos.
- *Falta de projeto formal.* A XP tira a ênfase da necessidade do projeto de arquitetura e, em muitos casos, sugere que todos os tipos de projetos devam ser relativamente informais. Seus críticos argumentam que em sistemas complexos deve-se enfatizar a elaboração do projeto para assegurar que a estrutura geral do software apresentará qualidade e facilidade de manutenção. Já os defensores da XP sugerem que a natureza incremental do processo XP limita a complexidade (a simplicidade é um valor fundamental) e, consequentemente, reduz a necessidade de um projeto extenso.

Casa Segura – Software Ágil

CASA SEGURA



Considerando o desenvolvimento de software ágil

Cena: de Doug Miller.

Atores: Doug Miller, gerente de engenharia de software; Jamie Lazar, membro da equipe de software; Vinod Raman, membro da equipe de software.

Conversa:

(Batendo à porta, Jamie e Vinod adentram à sala de Doug)

Jamie: Doug, você tem um minuto?

Doug: Com certeza, Jamie, o que há?

Jamie: Estivemos pensando a respeito da discussão sobre processos, de ontem... Sabe, que processo vamos escolher para este novo projeto CasaSegura.

Doug: E?

Vinod: Eu estava conversando com um amigo de uma outra empresa e ele me falou sobre a Extreme Programming. É um modelo de processo ágil... Já ouviu falar?

Doug: Sim, algumas coisas boas, outras ruins.

Jamie: Bem, pareceu muito bom para nós. Permite que se desenvolva software realmente rápido, usa algo chamado programação em dupla para fazer checagens de qualidade em tempo real... É bem legal, eu acho.

Doug: Realmente, apresenta um monte de ideias muito boas. Gosto do conceito de programação em dupla, por exemplo, e da ideia de que os envolvidos devam fazer parte da equipe.

Jamie: O quê? Quer dizer que o pessoal de marketing trabalhará conosco na equipe de projeto?

Doug (confirmando com a cabeça): Eles são envolvidos, não são?

Jamie: Jesus... Eles solicitarão alterações a cada cinco minutos.

Vinod: Não necessariamente. Meu amigo me disse que existem formas de se "abarcara" as mudanças durante um projeto XP.

Doug: Portanto, meus amigos, vocês acham que deveríamos usar a XP?

Jamie: Definitivamente, vale considerar.

Doug: Eu concordo. E mesmo que optássemos por um modelo incremental como nossa abordagem, não há nenhuma razão para não podermos incorporar muito do que a XP tem a oferecer.

Vinod: Doug, mas antes você disse "algumas coisas boas, outras ruins". O que são as "coisas ruins"?

Doug: O que não me agrada é a maneira pela qual a XP dá menos importância à análise e ao projeto... Dizem algo como: a codificação resume a ação para construir um software.

(Os membros da equipe se entreolham e sorriem.)

Doug: Então vocês concordam com a abordagem XP?

Jamie (falando por ambos): Escrever código é o que fazemos, chefe!

Doug (rindo): É verdade, mas eu gostaria de vê-lo perdendo um pouco menos de tempo codificando para depois recodificar e dedicando um pouco mais de tempo analisando o que precisa ser feito e projetando uma solução que funcione.

Vinod: Talvez possamos ter as duas coisas, agilidade com um pouco de disciplina.

Doug: Acho que sim, Vinod. Na realidade, tenho certeza disso.



Outros Modelos de Processos Ágeis

- Desenvolvimento de software adaptativo (ASD).
- Scrum.
- Método de desenvolvimento de sistemas dinâmicos.
- Crystal.
- Desenvolvimento dirigido a funcionalidades (FDD).
- Desenvolvimento de software enxuto (LSD).
- Modelagem ágil (AM).
- Processo unificado ágil (AUP).

Desenvolvimento de Software Adaptativo (ASD)

O ASD como uma técnica de construção de software e sistemas complexos.



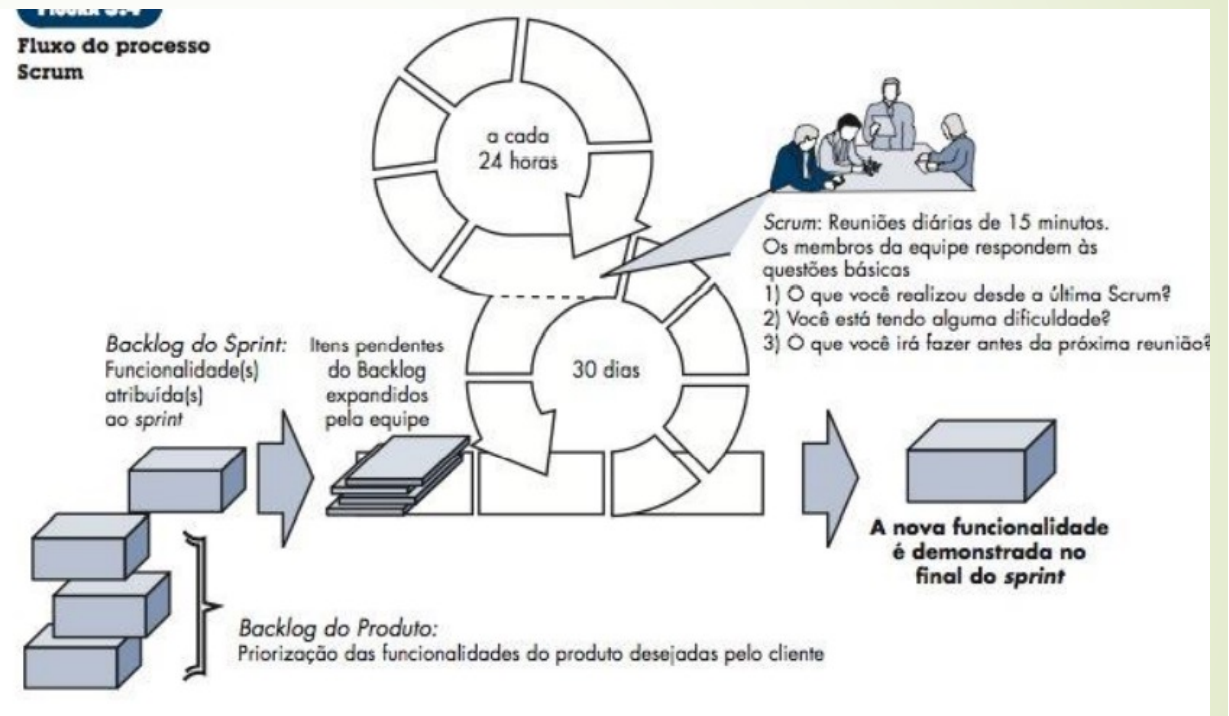


Scrum

- É um método de desenvolvimento ágil de software concebido por Jeff Sutherland e sua equipe de desenvolvimento no início dos anos 1990.
- Os princípios do Scrum são consistentes com o manifesto ágil e são usados para orientar as atividades de desenvolvimento dentro de um processo que incorpora as seguintes atividades estruturais: requisitos, análise, projeto, evolução e entrega. Em cada atividade metodológica, chamada **Sprint**.

Sprint - Scrum

- Reuniões Scrum – são reuniões curtas (tipicamente 15 minutos), realizadas diariamente pela equipe Scrum.
- Demos – entrega de incremento de software ao cliente para que a funcionalidade implementada possa ser demonstrada e avaliada pelo cliente.





Método de Desenvolvimento de Sistemas Dinâmicos (DSDM)

- O método DSDM é uma abordagem de desenvolvimento de software ágil que “oferece uma metodologia para construir e manter sistemas que atendem restrições de prazo apertado através do uso da prototipagem incremental em um ambiente de projeto controlado”
- O DSDM é um processo de software iterativo em cada interação segue a regra dos 80%. Ou seja, somente o trabalho suficiente é requisitado pra cada incremento, para facilitar o movimento para o próximo incremento.

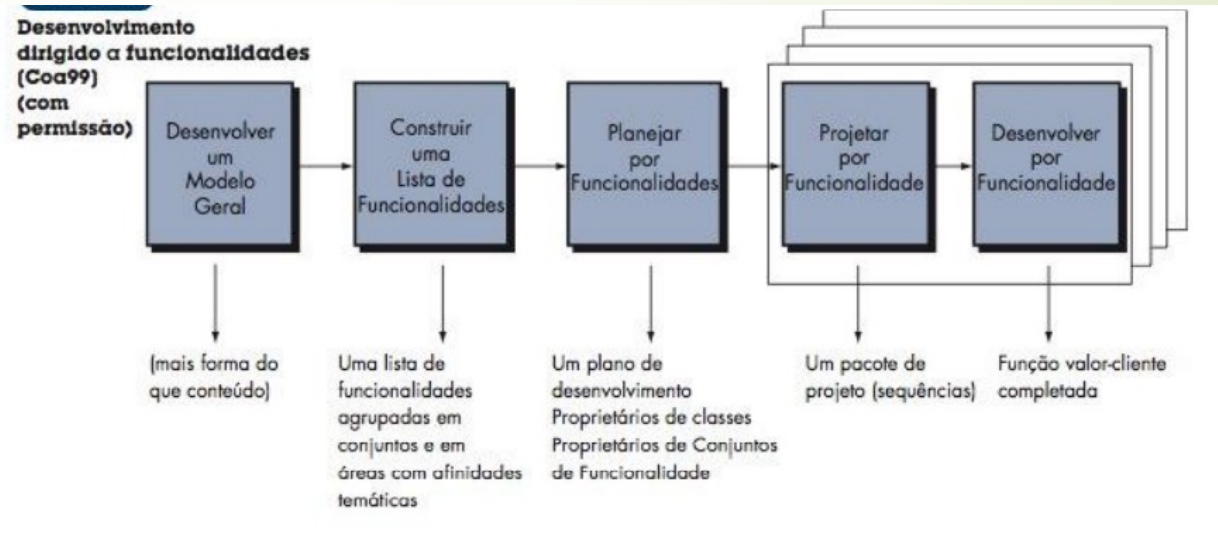



Crystal

- Elaborar uma abordagem de desenvolvimento de software que priorizasse a adaptabilidade caracterizado como um “jogo de invenção e comunicação cooperativo e com recursos limitados, tendo como primeiro objetivo entregar software útil em funcionamento e como segundo objetivo preparar-se para o jogo seguinte”

Desenvolvimento Dirigido a Funcionalidades (FDD)

- O FDD adota uma filosofia que enfatiza a colaboração entre pessoas da equipe, gerencia problemas e complexidade de projetos utilizando a decomposição baseada em funcionalidades, seguidas pela integração dos incrementos de software.





Desenvolvimento de Software Enxuto (LSD)

- O LSD adaptou os princípios da fabricação enxuta para o mundo da engenharia de software. Os princípios enxutos que inspiram o processo LSD podem ser sintetizados em eliminar desperdício, incorporar qualidade, criar conhecimento, adiar compromissos, entregar rápido, respeitar as pessoas e otimizar o todo.



Modelagem Ágil (AM)

Modele com um objetivo. O desenvolvedor que utilizar o AM deve ter um objetivo antes de criar o modelo (por exemplo, comunicar informações ao cliente ou ajudar a compreender melhor algum aspecto do software). Uma vez identificado o objetivo, ficará mais óbvio o tipo de notação a ser utilizado e o nível de detalhamento necessário.

Use modelos múltiplos. Há muitos modelos e notações diferentes que podem ser usados para descrever software. Somente um subconjunto é essencial para a maioria dos projetos. AM sugere que, para propiciar o insight necessário, cada modelo deve apresentar um aspecto diferente do sistema e somente aqueles que valorizem esses modelos para a audiência pretendida devem ser usados.

Viajar leve. Conforme o trabalho de engenharia de software prossegue, conserve apenas aqueles modelos que terão valor no longo prazo e despache o restante. Todo produto de trabalho mantido deve sofrer manutenção à medida que as mudanças ocorram. Isso representa trabalho que retarda a equipe. Ambler [Amb02a] observa que "Toda vez que se opta por manter um modelo, troca-se a agilidade pela conveniência de ter aquela informação acessível para a equipe de uma forma abstrata (já que, potencialmente, aumenta a comunicação dentro da equipe, assim como com os envolvidos no projeto)".

Conteúdo é mais importante do que a representação. A modelagem deve transmitir informação para sua audiência pretendida. Um modelo sintaticamente perfeito que transmite pouco conteúdo útil não possui tanto valor como aquele com notações falhas que, no entanto, fornece conteúdo valioso para seu público-alvo.

Tenha conhecimento, domínio dos modelos e das ferramentas que for utilizar. Compreenda os pontos fortes e fracos de cada modelo e ferramenta usada para criá-lo.

Adapte localmente. A abordagem de modelagem deve ser adaptada às necessidades da equipe ágil.



Processo Unificado Ágil (AUP)

- O AUP adota uma filosofia “serial para o que é amplo” e “iterativa para o que é particular”.
- Cada interação AUP dirige-se para as seguintes atividades: Modelagem, Implementação, Teste, Aplicação e Configuração e gerenciamento de projeto.



Conjunto de Ferramentas para o Processo Ágil

- Ferramentas voltas para a comunicação e para a colaboração são, em geral, de tecnologia de base e incorporam qualquer mecanismo que fornece informações e coordenação entre desenvolvedores. A comunicação ativa é obtida por meio de dinâmicas de grupo, enquanto a comunicação passiva é obtida através dos “irradiadores de informações” (por exemplo, um display de um painel fixo).
- Quaisquer dessas coisas são ferramentas? Serão, caso facilitem o trabalho desenvolvido por um membro da equipe ágil e venha a aprimorar a qualidade do produto final.