* **Extreme Programming(XP):**

Etapas utilizadas no XP:

**Planning:** As histórias do usuário (semelhante aos casos de uso) são documentadas para dar uma estimativa de tempo e visão ao planejamento, também direcionam os testes de aceitação. Outra diferença entre histórias e um documento de requisitos é o foco nas necessidades do usuário. Você deve tentar evitar detalhes de tecnologia, layout de banco de dados e algoritmos específicos. Você deve tentar manter as histórias focadas nas necessidades e benefícios do usuário, em vez de especificar layouts de GUI.

Prevê a entrega frequente de small releases / pequenos lançamentos.

Planejamentos das releases gera a programação das releases / entregas / lançamentos.

O planejamento das iterações inicia cada iteração.

**Managning:** Dê à equipe um espaço de trabalho aberto. Defina um ritmo sustentável. Uma reunião stand up (reunião em pé com a participação de todos envolvidos) começa todos os dias, os desenvolvedores relatam o que foi realizado ontem e o que será tentado hoje, e problemas que estão causando atrasos.

A velocidade do projeto é medida no gerenciamento.

Mova as pessoas.

Consertar XP quando ele quebrar, faça reuniões de retrospectiva para falar sobre o que está funcionando e o que não está, e planejar maneiras de melhor o XP.

**Designing:** Design simples leva menos tempo para terminar. Escolha o sistema metaphor / metáfora, para que caso entrem novas pessoas no projeto seja fácil de explicar do sistema, as classes e métodos devem ter uma nomenclatura consistente. Use cartões CRC – Colaboração de Classes e responsabilidades, para projetar o sistema como uma equipe. Criar spike solutions para reduzir riscos, é um programa simples para explorar soluções potenciais para problemas sob observação, ignorando todas as outras preocupações.

Nenhuma funcionalidade extra deve ser adicionada antes, devido ao risco de atrasar as reais necessidades do projeto.

Refatore quando for possível, quando removemos a redundância, eliminamos a funcionalidade não utilizada e renovamos designs obsoletos estamos refatorando. Refatorar ao longo de todo o ciclo de vida do projeto economiza tempo e aumenta a qualidade.

**Coding:** O cliente está sempre disponível, como parte do desenvolvimento.

O código deve ser escrito de acordo com os padrões de codificação.

Codifique o teste de unidade primeiro.

Todo o código de produção é programado em pares.

Apenas um par integra código de cada vez.

Integração contínua.

Configure um computador de integração dedicado.

Use propriedade coletiva, para que todas contribuam com novas ideias.

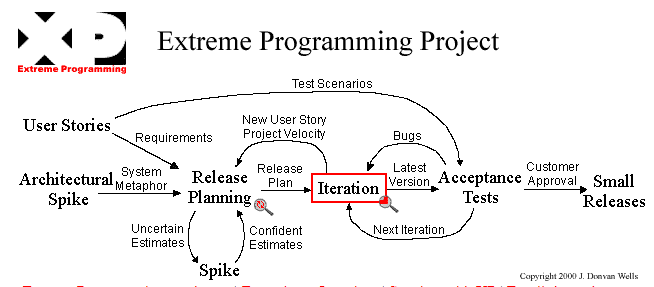
**Testing:** Todo código deve ter testes unitários, e deve passar em todos os testes unitários antes de ser lançado / released.

Quanto um bug é encontrado, os testes são criados.

Os testes de aceitação são executados com frequência e a pontuação publicada.

Exemplos de organizações que utlizam XP em alguns de seus projetos:

* + - IBM
    - Google.

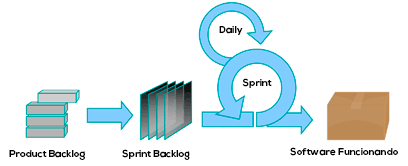


Fonte: <http://www.extremeprogramming.org/map/project.html>

* **SCRUM:**

Scrum é um framework para utilizado na gestão de projetos e desenvolvimento ágil de software.

Os projetos devem ter um princípio, meio e fim, também deve estar focado em gerar valor através de entregas continuas de software funcionando. As entregas normalmente são realizadas em ciclos com time box definido, esses ciclos são chamados de Sprint.



Product Backlog – A lista de desejos

As funcionalidades a serem desenvolvidas ficam em um artefato que se chama Product Backlog, que é de responsabilidade do Product Owner. No backlog os itens devem estar priorizados por valor de negócio.

Sprint em ação!

Antes de iniciar um Sprint, o time de desenvolvimento se reuni em um rito que chama, Sprint Planning, nessa reunião, o Product Owner apresenta para o time os itens mais prioritários do Product Backlog e o time selecionam os itens que farão parte do Sprint Backlog. Em um segundo momento da Planning, o time irá detalhar e definir como serão desenvolvidos os itens do Sprint Backlog.

Exemplos de organizações que utlizam SCRUM em alguns de seus projetos:

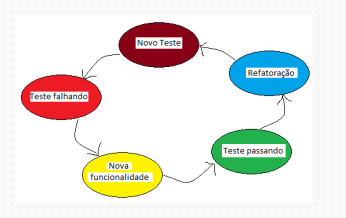
* Samsung
* Lenovo
* Visa
* PayPal
* **TDD Test Driven Develpment:**

Basicamente o TDD se baseia em pequenos ciclos de repetições, onde para cada funcionalidade do sistema um teste é criado antes. Este novo teste criado inicialmente falha, já que ainda não temos a implementação da funcionalidade em questão e, em seguida, implementamos a funcionalidade para fazer o teste passar! Simples assim!

Ciclo de desenvolvimento

Red,Green, Refactor. Ou seja:

* Escrevemos um Teste que inicialmente não passa (Red);
* Adicionamos uma nova funcionalidade do sistema;
* Fazemos o Teste passar (Green);
* Refatoramos o código da nova funcionalidade (Refactoring);
* Escrevemos o próximo Teste.



**Motivos para o uso**

* Feedback rápido sobre a nova funcionalidade e sobre as outras funcionalidades existentes no sistema
* Código mais limpo, já que escrevemos códigos simples para o teste passar
* Segurança no Refactoring pois podemos ver o que estamos ou não afetando
* Segurança na correção de bugs
* Maior produtividade já que o desenvolvedor encontra menos bugs e não desperdiça tempo com depuradores
* Código da aplicação mais flexível, já que para escrever testes temos que separar em pequenos "pedaços" o nosso código, para que sejam testáveis, ou seja, nosso código estará menos acoplado.

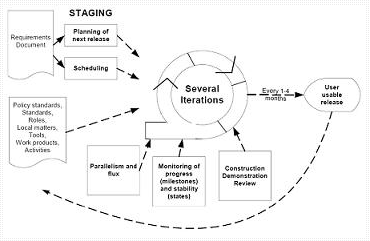
Exemplos de organizações que utlizam SCRUM em alguns de seus projetos:

* Amazon
* Grupo Bandeirantes
* **Crystal:**

Inclui grande número de métodos diferentes que são selecionados de acordo com as características do projeto a ser desenvolvido. Hoje em dia, apenas dois dos quatro métodos propostos mantêm o desenvolvimento de novas práticas para cobrir diferentes tipos de projetos.

Os membros da família Crystal são identificados por cores que indicam a intensidade do método. Neste caso, quanto mais escura a cor, maior é a complexidade do projeto.

Existem algumas características comuns à família Crystal, tais como o desenvolvimento incremental com ciclos de no máximo quatro meses, ênfase maior na comunicação e cooperação das pessoas, não limitação de quaisquer práticas de desenvolvimento, ferramentas ou produtos de trabalho e incorporação de objetivos para reduzir produtos de trabalho intermediários e desenvolvê-los como projetos evoluídos.



O ciclo de vida desta família de metodologia é baseado nas seguintes práticas:

* Staging: Planejamento do próximo incremento do sistema. A equipe seleciona os requisitos que serão implementados na iteração e o prazo para sua entrega;
* Edição e revisão: Construção, demonstração e revisão dos objetivos do incremento;
* Monitoramento: O processo é monitorado com relação ao progresso e estabilidade da equipe. É medido em marcos e em estágios de estabilidade;
* Paralelismo e fluxo: Em Crystal Orange as diferentes equipes podem operar com máximo paralelismo. Isto é permitido através do monitoramento da estabilidade e da sincronização entre as equipes;
* Inspeções de usuários: são sugeridas duas a três inspeções feitas por usuários a cada incremento;
* Workshops refletivos: são reuniões que ocorrem antes e depois de cada iteração com objetivo de analisar o progresso do projeto.
* Local matters: são os procedimentos a serem aplicados, que variam de acordo com o tipo de projeto.
* Work Products (Produtos de Trabalho): seqüência de lançamento, modelos de objetos comuns, manual do usuário, casos de teste e migração de código. Especificamente para o Clear: casos de uso e descrição de funcionalidade; Especificamente para o Orange: documento de requisitos.
* Standards (padrões): padrões de notação, convenções de produto, formatação e qualidade usadas no projeto.
* Tools: ferramentas mínimas utilizadas. Para Crystal Clear: compiladores, gerenciadores de versão e configuração, ferramentas de versão, programação, teste, comunicação, monitoramento de projeto, desenho e medição de performance.

Exemplos de organizações que utlizam SCRUM em alguns de seus projetos:

* IBM
* **Desenvolvimento de Software Adaptativo (ASD)**
* Desenvolvimento de Software Adaptativo (Adaptative Software Development - ASD) é uma técnica para construção de software e sistemas altamente complexos. Esse modelo se concentra na colaboração e auto-organização das equipes.
* O criador do modelo Adaptativo define um ciclo de vida para o modelo baseando-se em três fases: especulação, colaboração e aprendizagem.
* Na fase de especulação o projeto é iniciado e tem-se o planejamento de ciclos adaptáveis. Esse planejamento de ciclos adaptáveis usa as informações contidas no início do projeto como: a missão do cliente, restrições do projeto e os requisitos básicos. Os requisitos básicos serão utilizados para definir o conjunto de ciclos da versão, ou seja, os incrementos de software operacional. Vale salientar que esse plano de ciclos sofrerá mudanças. Após completar cada ciclo o plano é revisto e ajustado para que tenhamos o trabalho reajustado à realidade que a equipe ASD está trabalhando.
* A colaboração é um tema bastante discutido e enfatizado nos métodos ágeis. A colaboração envolve confiança, críticas sem animosidade, auxílio, trabalho árduo, comunicação dos problemas ou preocupações de forma a conduzir ações efetivas, etc. Dessa forma, a colaboração ajuda bastante no levantamento de necessidades, especificações, etc.
* O ASD também enfatiza que o aprendizado é um elemento-chave para que possamos conseguir uma equipe auto-organizada. O criador do método argumenta que os desenvolvedores superestimam o seu próprio entendimento quanto à tecnologia, processo ou mesmo quanto ao projeto. Dessa forma, enfatiza-se que o aprendizado irá ajudar a todos os desenvolvedores a aumentar os níveis reais de entendimento. Com base nisso, as equipes ASD aprendem através de três maneiras: grupos focados, revisões técnicas e autópsias de projetos.

Exemplos de organizações que utlizam ASD em alguns de seus projetos:

* Não encontrado
* **Desenvolvimento dirigido à funcionalidade (FDD)**

Peter Coad e seus colegas criaram um modelo de processo prático para a engenharia de software orientada a objetos chamada Desenvolvimento Dirigido a Funcionalidade ou Feature Driven Development (FDD). Após o trabalho de Coad surgiu outro modelo de processo que aperfeiçoou o trabalho desenvolvido. Este trabalho foi realizado por Stephen Palmer e John Felsing que descreveram um processo ágil adaptativo que pode ser aplicado em projetos de médio e grande porte.

Seguindo as outras abordagens ágeis, o FDD também evidencia a colaboração entre as pessoas da equipe, gerencia os problemas e complexidades de projetos utilizando a decomposição baseada em funcionalidades que é seguida pela integração dos incrementos de software e por fim também enfatiza a comunicação de detalhes técnicos usando formas verbais, gráficos e texto. Além disso, o FDD enfatiza as atividades de garantia da qualidade por meio da estratégia de desenvolvimento incremental, inspeções de código e de projeto, auditorias, coleta de métricas e o uso de padrões para análise, projeto e construção.

O FDD preconiza que a funcionalidade é uma função valorizada pelo cliente passível de ser implementada em até duas semanas ou menos. Essa ênfase na definição de funcionalidades tem como benefícios uma descrição mais fácil das funcionalidades por parte do usuário, uma melhor compreensão como elas se relacionam entre si e uma melhor revisão das ambiguidades, erros ou omissões. Isso ocorre porque elas são pequenas, formadas em pequenos blocos. Além disso, outro benefício é que as funcionalidades podem ser organizadas em um agrupamento hierárquico relacionados com o negócio, pode ser entregue a cada duas semanas, o projeto e o código são mais facilmente inspecionados e o planejamento, cronograma e acompanhamento do projeto são guiados pela hierarquia de funcionalidades ao invés de tarefas de engenharia de software.

Coad e seus colegas sugerem os seguintes modelos para definirmos uma funcionalidade: <ação> o <resultado> <por|para quem|de|para que> um <objeto> Onde objeto é "uma pessoa, local ou coisa".

Um exemplo de funcionalidades poderia ser:

Adicione o produto ao carrinho

Mostre as especificações técnicas do produto

O FDD também define cinco processos, são eles: Desenvolver um Modelo Geral, Construir uma Lista de Funcionalidades, Planejar por Funcionalidades, Projetar por Funcionalidade, Desenvolver por Funcionalidade.

Um aspecto interessante do FDD em relação a outros métodos ágeis é que ele dá maior ênfase às diretrizes e técnicas de gerenciamento de projetos do que outros métodos ágeis disponíveis.

Exemplos de organizações que utlizam ASD em alguns de seus projetos:

* Ericsson
* **Dynamic System Development Method (DSDM)**

O DSDM (Dynamic System Development Model ou Desenvolvimento de Sistemas Dinâmicos em português), como o próprio nome indica, é um arranjo dinâmico de desenvolvimento de sistemas criado no ano de 1990, no Reino Unido, por meio do consórcio DSDM.

O consórcio formado pela reunião de especialistas e consultores da área de engenharia de software somou o seu know-how para desenvolver uma extensão independente do RAD (Rapid Application Development) para o uso em projetos com prazos e orçamentos restritos.

O resultado da iniciativa é o DSDM. Uma solução que também é reconhecida como sendo o pai dos métodos ágeis e que foi pensada para o desenvolvimento de projetos de alta qualidade, mas limitados em termos de prazos curtos e orçamentos fixos. Essa rigidez um pouco mais acentuada é o que diferencia o DSDM de outras metodologias.

Como funciona o DSDM?

As melhores práticas que caracterizam essa metodologia são:

* Desenvolvimento incremental (ciclo contínuo de melhorias das versões anteriores);
* Colaboração entre cliente e equipe de trabalho;
* Funções integradas;
* Flexibilidade nas exigências.

O funcionamento do DSDM está baseado na construção e na entrega de partes de um módulo que, em seguida, são interligadas a outras de maneira sucessiva até que os requisitos e as funcionalidades do software ou projeto em questão estejam todos implementados.

Outra particularidade sobre o funcionamento do DSDM é a interação frequente entres os usuários finais e a equipe de desenvolvimento, que trocam informações entre si em cada novo estado do projeto e a cada novo período de testes.

Exemplos de organizações que utlizam ASD em alguns de seus projetos:

* Sema Group.
* Boston Globe.
* Scottish Natural Heritage.
* **Processo Unificado Ágil (AUP)**

O Processo Unificado Ágil ou Agile Unified Process (AUP) adota uma filosofia serial (sequência linear de atividades) para o que é amplo e iterativo para o que é particular. Este processo possui atividades nas fases clássicas adotadas pelo Processo Unificado: Início, Elaboração, Construção e Transição. Dentro de cada uma das atividades a equipe itera ou se repete para alcançar a agilidade e entregar incrementos de software para os usuários finais. É importante que essa entrega seja mais rápida quanto possível.

A AUP possui seis atividades, na qual a equipe irá sempre iterar. As atividades são:

* Modelagem: Modelos UML representando o negócio são criados. Os modelos devem ser suficientemente bons e adequados para permanecer ágil.
* Implementação: Modelos são traduzidos para o código-fonte.
* Teste: A equipe projeta e executa uma série de teste a fim de descobrir possíveis erros e para assegurar que o código-fonte se ajuste aos requisitos.
* Aplicação: É a entrega de um incremento de software e a aquisição de feedback dos usuários finais com base neste incremento.
* Gerenciamento de Configuração e Gerenciamento de projeto: No contexto da AUP, gerenciamento de configuração refere-se ao gerenciamento de alterações, riscos e de controle de qualquer artefato persistente que são produzidos pela equipe. O gerenciamento de projeto controla o progresso de uma equipe e coordena suas atividades.
* Gerenciamento de Ambiente: Coordena toda a infraestrutura de processos incluindo padrões, ferramenta e outras tecnologias de suporte disponíveis para a equipe.

Exemplos de organizações que utlizam AUP em alguns de seus projetos:

* Não encontrado