Alunos: Tiago Peggorini, Maurício Macário de Farias Júnior, Jean Rodrigo, Gabriel Conti, Leonardo Maurício de Farias

XP – Extreme Programming

Definição:

Extreme Programing: O XP é um método de desenvolvimento de software, leve, não é prescritivo e procura fundamentar as suas práticas por um conjunto de valores . O XP, também pode ser adotado por desenvolvedores médios e não apenas por desenvolvedores experientes.

Príncipios, práticas e valores:

O objetivo principal do XP é levar ao extremo um conjunto de práticas que são ditas como boas na engenharia de software.

- Já que testar é bom, que todos testem o tempo todo;
- Já que revisão é bom, que se revise o tempo todo;
- Se projetar é bom, então refatorar o tempo todo;
- Se teste de integração é bom, então que se integre o tempo todo;
- Se simplicidade é bom, desenvolva uma solução não apenas que funcione, mas que seja a mais simples possível;
- Se iterações curtas é bom, então mantenha-as realmente curtas;

O XP reconhece cinco valores, são os seguintes:

Comunicação: Construir sistemas de software requer a comunicação dos requisitos do sistemas para os desenvolvedores do sistema, normalmente executado através de documentação. O objeto do XP é dar a todos os desenvolvedores uma visão compartilhada do sistema. Por isso o XP favorece designs simples, metáforas comuns, colaboração de programadores e usuários, comunicação verbal frequente e feedback.

Simplicidade: É encorajado a solução mais simples, o foco é em resolver o problema hoje, mais funcionalidade podem ser adicionadas mais tarde, esse valor facilita também a comunicação, pois uma solução simples é mais fácil de ser entendida pelos desenvolvedores

Feedback: Possui três dimensões:

- Feedback do sistema, produzidos pelos testes executados, sejam de unidade ou teste periódicos de integração.
- Feedback do cliente, os testes funcionais (testes de aceitação) são escritos pelos usuários e testadores.
- Feedback da equipe, quando clientes criam novos requisitos e a equipe cria uma estimativa do tempo que irá levar para implementar aquilo.

Esse valor é diretamente ligado a comunicação e simplicidade.

Coragem: Muitas praticas envolvem coragem, uma delas é o fato de resolver os problemas para hoje, a coragem ajuda os desenvolvedores a se sentirem confortáveis a refatorar o código sempre que necessário.

Respeito: Esse valor inclui respeito a si mesmo e aos outros, programadores não devem realizar mudanças que quebram o código, ou que fazem algum teste de unidade falhar, gerando problemas para o próximo ou para si mesmo.

Fases, iterações, atividades/cerimônias, papéis, artefatos:

A fase de exploração: Antecede á construção do sistema, e é nela eu são feitas as investigações a respeito das soluções, ambiente tecnológico. Para se construir o primeiro release.

A fase de planejamento inicial: Nessa fase clientes e programadores planejam o lançamento do primeiro release, se juntam e definem os casos de uso classificando suas dificuldades e quantos casos de uso são executados em cada iteração, isso é chamado de planejamento de iteração. Com uma definição de tempo de iteração de uma a três semanas e de dois a quatro meses para cada release.

A fase das iterações do release: são escritos os casos de teste funcionais e de unidade. Os programadores vão seguindo mais ou menos o seguinte fluxo de atividades na seguinte ordem (Em cada iteração): escrita dos casos de testes; projeto e refatoramento; codificação; realização dos testes; e integração. Depois de terminado o primeiro release, já se terá uma idéia melhor das tecnologias e do domínio do problema de modo que as iterações poderão ser mais curtas nos releases subseqüentes e já se podem fazer estimativas mais confiáveis com o que se aprendeu das iterações passadas.

A fase de manutenção: Em XP você está simultaneamente produzindo novas funcionalidades, mantendo o sistema existente rodando, incorporando novas pessoas na equipe e melhorando o código. Mecanismos como: refatoramento, introdução de novas tecnologias, e introdução de novas idéias de arquitetura podem ser utilizados em um projeto XP. É importante ressaltar que a manutenção dada em um sistema que já está em produção deve ser feita com muita cautela, pois uma alteração errada pode paralisar o funcionamento do sistema resultando em prejuízos para o cliente.

A fase de morte: corresponde ao término de um projeto XP. Existem duas razões para se chegar ao final de um projeto, uma boa e a outra ruim. A boa razão é quando o cliente já está satisfeito com o sistema existente e não enxerga nenhuma funcionalidade que possa vir a ser implementada no futuro. A má razão para a morte em XP seria a do projeto ter se tornado economicamente inviável, devido a dificuldades de adicionar funcionalidades a um custo baixo e devido a uma alta taxa de erros.

O XP descreve quatro atividades básicas que são executadas no processo de desenvolvimento de software, são as seguintes:

Codificação: Acredita-se que o único produto importante do desenvolvimento de software é código. Codificação também pode ser usado para descobrir a mais adequada solução para um problema.

Testes: Existem dois tipos de testes:

- Testes unitários: Determinam quando uma determinada funcionalidade está funcionando como o esperado, cada peça de código é testada antes de mover para a próxima funcionalidade.
- Testes de aceitação: Verifica que os requisitos, como são entendidos pelos programadores satisfazem realmente os requisitos do cliente.

Listening: Programadores precisam ouvir o que os clientes precisam que os sistemas façam, qual a lógica de negocio. É preciso entender essas necessidades para poder se comunicar com os clientes sobre o que pode ser feito para resolver os problemas.

Concepção: Caso essa fase não for executada, o sistema começa a ficar muito complexo. A concepção ajuda a evitar que haja muitas dependências internas no projeto, isso significa que mudar uma parte do projeto não irá afetar as outras partes do mesmo.

Os papéis que se mostram mais efetivos em XP são:

Desenvolvedor ou programador: O papel mais importante do desenvolvimento, é compreendido por:

- *Direitos:*
  - Saber o que é preciso, com declarações claras de prioridades
  - Poder pedir ajuda aos seus superiores, parceiros e usuários
  - Poder decidir suas próprias estimativas
  - Você aceitar suas responsabilidades, em vez de elas serem atribuídas a você
- *Deveres e responsabilidades:*
  - Estimar estórias
  - Definir tarefas a partir de estórias
  - Estimar tarefas
  - Escrever testes de unidades
  - Escrever códigos para passar pelos testes de unidades
  - Realizar testes de unidade
  - Refatorar
  - Integração continua
  - Propriedade coletiva
- *Habilidades:*
  - Programação em par
  - Comunicação
  - Codificar apenas o necessário
  - Manter a simplicidade

Cliente: Define o que vai ser programado, são necessárias algumas habilidades, como:

- Detalhar o problema
- Influenciando um projeto sem conseguir controla-lo
- Sempre disponível para decisões de funcionalidades
- Escrever testes funcionais

Responsabilidades:

- Escrever estórias
- Escrever testes funcionais
- Definir prioridades nas estórias
- Explicar estórias
- Decidir questões sobre as estórias

Gerente ou rastreador: Suas responsabilidades englobam:

- Definir as regras do planejamento
- Familiarizar a equipe e os clientes com o planejamento
- Monitorar o planejamento, ajudar qualquer desvio, modificar as regras se e quando requerido
- Programar e conduzir o planejamento de lançamento e as reuniões de planejamento de iteração.
- Ajudar a equipe quando elas fizerem estimativas, para dar feedback, para assim saber se estão conforme as estimativas passadas, ambas em níveis de equipe e individuais.
- Garantir que a equipe esteja trabalhando para o próximo lançamento.
- Rastrear as falhas de testes funcionais
- Monitorar tempo gasto por cada membro.

Treinador (Coach): Recebe a responsabilidade por toda a equipe, seus principais deveres são:

- Entender a fundo a aplicação do XP
- Identificar as práticas do XP que deverão ser utilizadas
- Manter-se calmo
- Observar e analisar a equipe, e apenas interferir quando realmente preciso

Quando utilizar? Discussão sobre pontos fortes e fracos (críticas)

Não existe uma regra geral de como devemos adotar o XP na nossa organização. O ideal é que o processo seja gradual, caso seja a primeira vez a utiliza-lo e, com grande interação do cliente com a equipe no processo.

Quando usar?

- Equipes com no máximo 12 integrantes
- Clientes que estejam dispostos a se dedicar semanalmente pois requer várias reuniões com os clientes

- Ter um espaço físico onde é possível todos os envolvidos ficarem próximos uns dos outros.

Vantagens:

- Um dos benefícios é tornar o processo ágil e mais flexível. As práticas do XP foram criadas para funcionarem juntas
- Fornecem mais valor do que cada uma poderia fornecer individualmente
- Analise previa dos acontecimentos dentro do projeto, o que oferece qualidade, confiança, datas de entrega e custos promissores.
- O XP é ideal para projetos em que o cliente não sabe o exatamente o que quer, pois, os feedbacks constantes tornam possível as mudanças para atender os requisitos de forma rápida.
- Entregas constantes de partes funcionais do software assim o cliente não precisa esperar muito para ver o software funcionando
- Programação em dupla reduz o número de erros e aumenta a legibilidade do código o que facilita manutenções futuras.
- Erros são encontrados em um estágio inicial, pois são realizados diversos testes de diversas formas.

Criticas:

- Requer uma grande quantidade de reuniões, gerando grandes despesas para o cliente.
- Pode levar a negociações de contrato mais difíceis
- Pode ser ineficiente, pois áreas de código podem ser modificadas várias vezes
- Pode aumentar a chance de escorregamento de escopo, que é quando o escopo vai aumentando conforme o desenvolvimento do projeto.
- Impossível realizar estimativas exatas, pois no começo do projeto, não se sabe o escopo inteiro do projeto.
- Existe uma perda de produtividade adotando programação em pares
- Exige muito código para testes e a razão entre linhas de código de teste para linhas de código de projeto deve ser no mínimo 2:1
- A análise de requisitos informal pode não ser bem visto pelos clientes que podem se sentir inseguros
- Uma cultura na qual você e requisitado a trabalhar horas e horas para provar seu “comprometimento com a empresa”, você não conseguira executar o XP se estiver cansado

Processo de Software Lean

Definição: Tem o intuito de aplicar esta metodologia em empresas de software com uma abordagem de desenvolvimento enxuto no desenvolvimento de software com isso evoluiu da experiência de gerenciamento de riscos desenvolvendo uma abordagem que possui três camadas para que seja possível transformar as mudanças em uma vantagem competitiva, mas tem um motivo que ela não é específica para a prática do desenvolvimento de software descrevendo apenas uma série de princípios, valores e ferramentas que devem ser utilizados para tornar o desenvolvimento de software um desenvolvimento enxuto.

Princípios, Valores e praticas: A metodologia Lean é distribuída em 7 princípios:

Eliminar o desperdício- Durante o desenvolvimento de software existem muitos cenários com desperdícios, e a eliminação de desperdícios como, dinheiro, recursos, esforço e tempo, a cada etapa e atividade realizada o processo deve contribuir para um resultado com mais qualidade, menos custo e maior rapidez.

Amplificar o aprendizado- Tentar tornar todas as dificuldades encontradas em fonte de conhecimento para amadurecer a equipe e influenciar no processo de forma positiva e construtiva. A capacidade de absorver mudanças pode ser usada como estratégia para chegar em um objetivo.

Adiar comprometimentos- Com o objetivo de mudanças serem vistas como oportunidades para aprender e cumprir as metas, surge o princípio de adiar comprometimentos e manter a flexibilidade, pois adiar as decisões permite que as escolhas sejam apoiadas com mias experiencias.

Manter a flexibilidade- Mudanças serão vistas como oportunidades para aprender e atingir metas.

Entregar rápido- A experiencia e a segurança na tomada de decisões são resultados obtidos pela equipe através de iterações curtas, influenciando no atendimento das necessidades atuais dos clientes.

Tornar a equipe responsável- o envolvimento das pessoas nas decisões dos detalhes técnicos é de fundamental importância, E a metodologia Lean usa técnica que ajudam os desenvolvedores a identificar o trabalho que precisa ser realizado.

Construir integridade e visualizar o todo- está relacionado ao dever da equipe de elaborar soluções que tragam segurança de que o desenvolvimento será realizado com qualidade, a integridade do software é construída a partir de uma arquitetura coerente. E a análise total do produto é essencial devido a muitas complexidades que envolvem os sistemas, para tornar possível o aumento da satisfação do cliente e tornar o software consistente.

Fases, iterações, atividades/cerimônias, papéis, artefatos:

Sensibilização e Preparação

Consiste na compreensão de conceitos da filosofia Lean e as suas estratégias em uma linguagem simples e objetiva auxiliando na compreensão do processo, nesta fase procura-se mostrar a importância de se ter a definidas as estratégias da empresa para buscar oportunidades de negócios procura-se enfatizar a importância do planejamento com esforços concentrados na fase inicial de projeto.

Sensibilização e Preparação

Aplicar todos os passos estabelecidos na fase anterior, de acordo com as necessidades e as prioridades que requer cada tarefa e também procura praticar os conhecimentos adquiridos sobre os princípios *Lean*, com objetivo de criar um conjunto de rotinas que torne o desenvolvimento de produto buscando o desenvolvimento do conceito mais flexível, competitivo e inovador possível.

Aplicação e Avaliação

Busca avaliar, ajustar e reavaliar o processo como um todo dando ênfase a dois momentos mais importantes da avaliação primeiro é a avaliação do conjunto de atividades que foram realizadas seguindo o método proposto e o segundo, analisar o produto obtido como resultado dessas atividades.

SCRUM

DEFINIÇÃO

É um framework para organizar e gerenciar trabalhos complexos, tal como projetos de desenvolvimento de software.

A **metodologia Scrum** em si é utilizada para organizar e gerenciar projetos utilizando-se dos valores e princípios de outros métodos de desenvolvimento de software (usualmente ágil) em conjunto com um fluxo e os elementos definidos no framework Scrum.

O foco da analise do Scrum neste trabalho é em sua forma ágil.

PRINCIPIOS

Segue os princípios de desenvolvimento de software abordados pela metodologia de desenvolvimento de software escolhida. Como usualmente tende a ser conjugado com o método de desenvolvimento ágil, neste caso os princípios seriam aqueles definidos no **Manifesto ágil**.

VALORES/BASE FUNDAMENTAL

PRÁTICAS

No Scrum, os projetos são divididos em ciclos (tipicamente mensais) chamados de **Sprints**. O Sprint representa um Time Box dentro do qual um conjunto de atividades deve ser executado.

Ao início de cada Sprint, é realizado um **Sprint Planning Meeting**, uma reunião de planejamento na qual o **Product Owner** (um dos papéis) prioriza os itens do **Product Backlog** (lista de funcionalidades) e a equipe seleciona as atividades que

ela será capaz de implementar durante o Sprint que se inicia. As tarefas alocadas em um Sprint são transferidas do Product Backlog para o **Sprint Backlog**.

FASES / ITERAÇÕES

Metodologias ágeis de desenvolvimento de software são iterativas, ou seja, o trabalho é dividido em iterações, que são chamadas de Sprints no caso do Scrum.

ATIVIDADES/CERIMONIAS

- **Sprint Planning Meeting:** uma reunião de planejamento.
- **Daily Scrum:** A cada dia de uma Sprint, a equipe faz uma breve reunião (normalmente de manhã).
- **Sprint Review Meeting:** Ao final de uma Sprint a equipe apresenta as funcionalidades implementadas.
- **Sprint Retrospective:** A equipe parte para o planejamento do próximo Sprint. Assim reinicia-se o ciclo.

PAPÉIS

Product Owner

- Faz na prática o papel do cliente, representando tanto quem está pagando pelo produto quanto quem irá utilizá-lo efetivamente.
- Define o que é esperado do produto final, quais os requisitos e quais funcionalidades deverão ser entregues.
- Participa durante todos o processo para a definição das prioridades de entrega de cada Sprint e define as mudanças do que está contido no Product Catalog.
- Comunica a todos os participantes uma visão clara do que a equipe está buscando alcançar no projeto. Também deve estar sempre disponível para esclarecer eventuais questionamentos assim que eles apareçam.

Scrum Master

- Não é considerado exatamente um “gerente”, porém possui sim um papel de liderança, é um facilitador dentro do time.
- É responsável por garantir que todo o time esteja atuando com base nos princípios e práticas do Scrum, mantendo o time produtivo e focado no que foi planejado.
- Possui o dever de ajudar o time a resolver problemas e interferências externas que possam prejudicar o andamento do desenvolvimento das atividades planejadas.

Scrum Team

- É um time multidisciplinar responsável pelo desenvolvimento e entrega das funcionalidades.
- Esse time deve ter a capacidade de realizar todas as fases do processo de desenvolvimento, incluindo design, codificação, testes, documentação e etc.
- Preferencialmente devem ser mantidos times pequenos, com não mais que 9 profissionais.
- O Scrum Team deve se auto organizar, com o apoio do Scrum Master, para atender da melhor maneira possível o que está sendo definido pelo Product Owner.

ARTEFATOS

As funcionalidades a serem implementadas em um projeto são mantidas em uma lista que é conhecida como Product Backlog.
Tarefas alocadas em um Sprint são transferidas do Product Backlog para o Sprint Backlog.

QUANDO UTILIZAR?

O Scrum é utilizado principalmente quando o projeto é complexo, no qual é muito difícil predeterminar totalmente o que irá acontecer até a entrega final.

VANTAGENS

- **Realização:** A conclusão por etapas é extremamente motivadora para a equipe.
- **Transparência:** A metodologia prevê que o projeto seja observado e acompanhado por todos que fazem parte dele ou que fazem parte da organização.
- **Menos falhas:** O foco da metodologia na qualidade faz com que haja uma redução dramática na quantidade de bugs nos softwares produzidos sob essa gestão.
- **Reordenação:** A segmentação do projeto torna possível inverter prioridades, de acordo com o andamento de cada etapa, e concentrando esforços para finalizar etapas que ainda não foram terminadas, por exemplo.

DESVANTAGENS

- **Segmentação:** A segmentação e a tentativa de ser ágil pode levar a equipe a perder a perspectiva do projeto como um todo. Causando falhas na hora de encaixar as partes e concluir o programa.
- **Documentação:** O fato da gestão do projeto ser dividida em caixas faz com que, muitas vezes, apenas as etapas estejam documentadas. Dessa forma, o projeto não tem um acompanhamento passa a passo como um todo.
- **Prazos:** O foco na qualidade de cada uma das etapas pode levar a atrasos. Gerando, dessa forma, desgastes com os clientes na negociação de novos prazos.
- **Referências:** Não se trata da única metodologia ágil disponível no mercado e, muitas vezes, é difícil encontrar bons scrum masters para adotar como referência no projeto.

Referências:

https://rafaellsantana.wordpress.com/2014/10/01/ciclo-de-vida-do-extreme-programing/

https://www.devmedia.com.br/introducao-ao-extreme-programming-xp/29249

http://www.extremeprogramming.org/

https://en.wikipedia.org/wiki/Extreme_programming#Activities

http://wiki.c2.com/?ExtremeProgramming

http://www.extremeprogramming.org/

https://www.tutorialspoint.com/extreme_programming/extreme_programming_roles.htm

http://www.mindmaster.com.br/scrum/

http://www.scrum.mindmaster.com.br

https://www.desenvolvimentoagil.com.br/scrum/

https://youtu.be/XfvQWnRgxG0