



Instituto Federal de Educação, Ciência e Tecnologia da Bahia - IFBA  
Departamento de Ciência da Computação  
Tecnólogo em Análise e Desenvolvimento de Sistemas

## Desenvolvimento ágil

André L. R. Madureira <[andre.madureira@ifba.edu.br](mailto:andre.madureira@ifba.edu.br)>  
Doutorando em Ciência da Computação (UFBA)  
Mestre em Ciência da Computação (UFBA)  
Engenheiro da Computação (UFBA)

# Conceito de *Stakeholder*

---

- É quem tem alguma influência sobre os requisitos do sistema.
  - **Ex:** usuários finais que irão interagir com o sistema
  - **Ex:** qualquer outra pessoa em uma organização que será afetada pelo sistema



# Porque desenvolvimento ágil?

---

- Processos de desenvolvimento de software que planejam completamente os requisitos de um sistema não estão adaptados ao desenvolvimento rápido de software
  - Mudanças ou problemas nos requisitos faz com que o sistema precise ser projetado, implementado e testado novamente
  - **Resultado:** modelo em cascata ou baseado em especificações costuma ser demorado e pode ter custo maior do que o previsto

# Desenvolvimento incremental

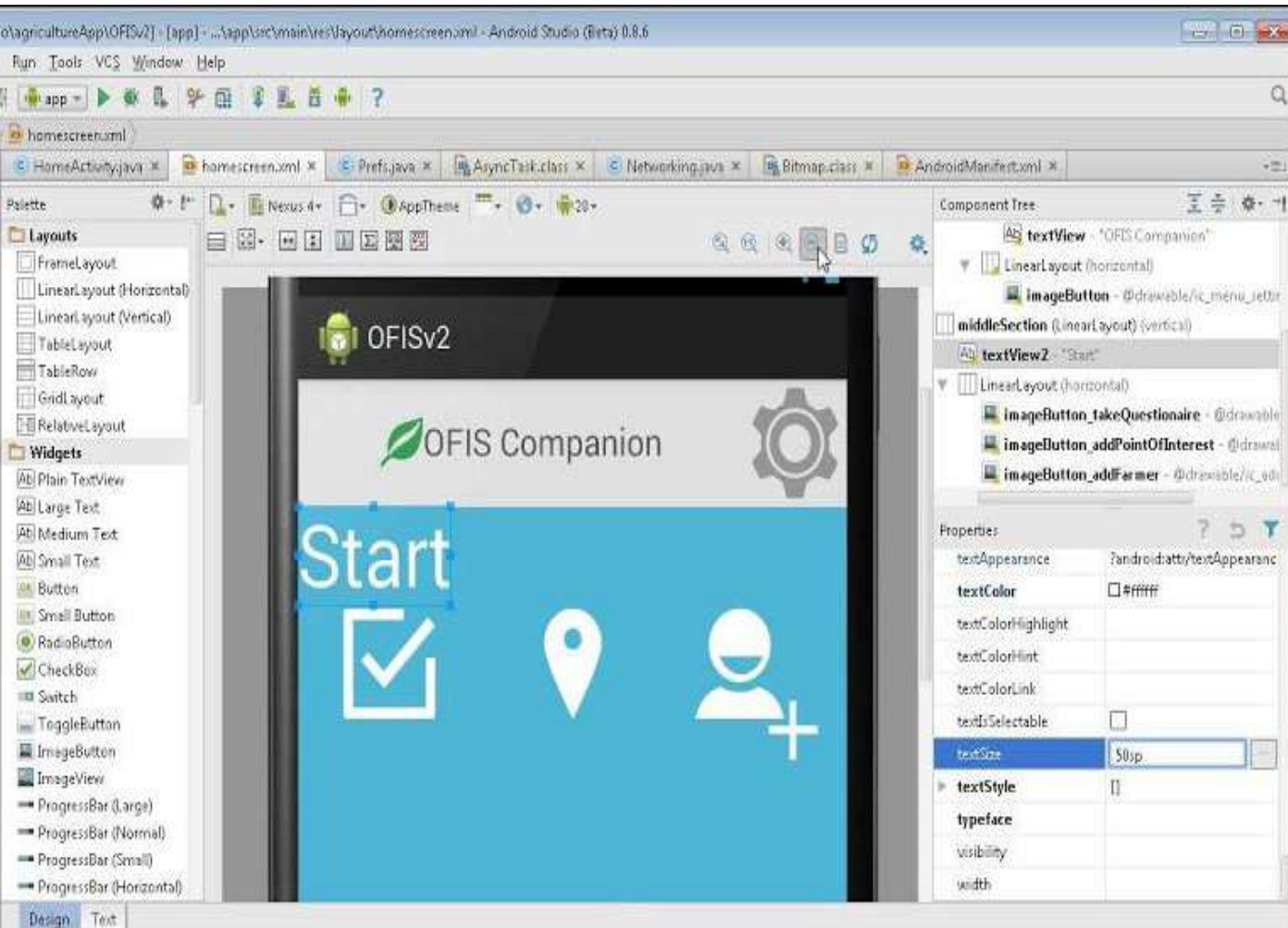
---

- Processos (especificação, projeto e implementação) são **intercalados**
- O sistema é desenvolvido em uma série de **versões**, contendo um conjunto de funcionalidades bem definidas
  - Os usuários finais envolvidos na especificação e avaliação de cada versão.
    - Propõem alterações ao software e novos requisitos para serem implementados em uma versão posterior

# Desenvolvimento incremental

---

- Interfaces de usuário do sistema são geralmente desenvolvidas através de um sistema interativo de desenvolvimento
  - Criação rápida do projeto de interface por meio de desenho e posicionamento de ícones na interface



# Métodos ágeis

---

- Filosofia dos métodos ágeis foi descrita no **manifesto ágil**, no qual há uma valorização maior de:
  - *“Indivíduos e interações do que processos e ferramentas”*
  - *“Software em funcionamento do que documentação abrangente”*
  - *“Colaboração do cliente do que negociação de contrato”*
  - *“Respostas a mudanças do que seguir um plano”*

# Princípios dos Métodos ágeis

---

- Embora existam vários métodos ágeis, eles compartilham um mesmo conjunto de princípios:
  - **Envolvimento do cliente**
  - **Entrega incremental**
  - **Pessoas, não processos**
  - **Aceitar as mudanças**
  - **Manter a simplicidade**



# Envolvimento do cliente

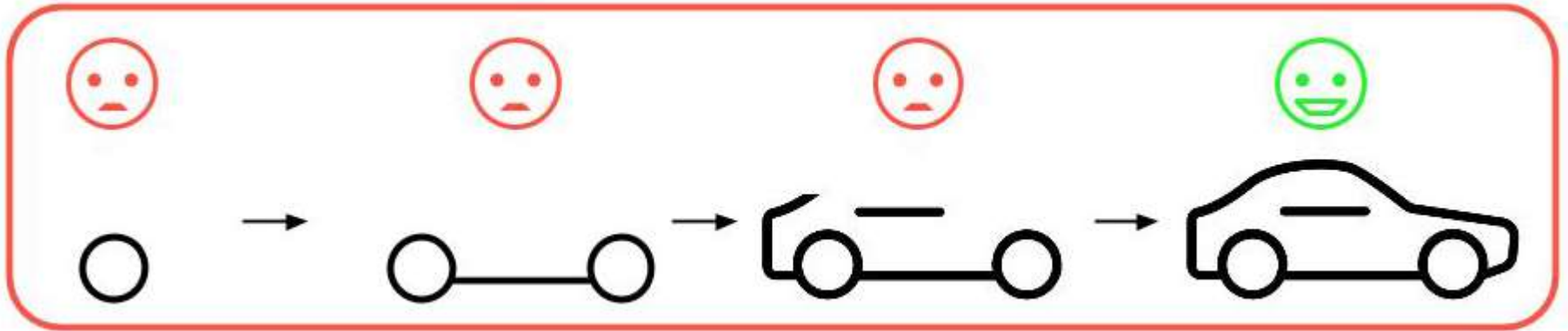
---

- Clientes intimamente envolvidos no processo de desenvolvimento, pois eles:
  - Fornecem e priorizam novos requisitos do sistema
  - Avaliam as iterações do sistema (incrementos / versões)



# Entrega incremental

- O software é desenvolvido em incrementos com o auxílio do cliente
  - Cliente especifica os requisitos a serem incluídos em cada versão / incremento



# Pessoas, não processos

---

- As habilidades da equipe de desenvolvimento devem ser reconhecidas e exploradas
- Membros da equipe devem desenvolver suas próprias maneiras de trabalhar, sem processos prescritivos (obrigatórios)

django



# Pessoas, não processos



Communication



Self-motivation



Leadership



Responsibility



Teamwork



Problem solving



Decisiveness



Ability to Work  
Under Pressure  
and Time Management



Flexibility

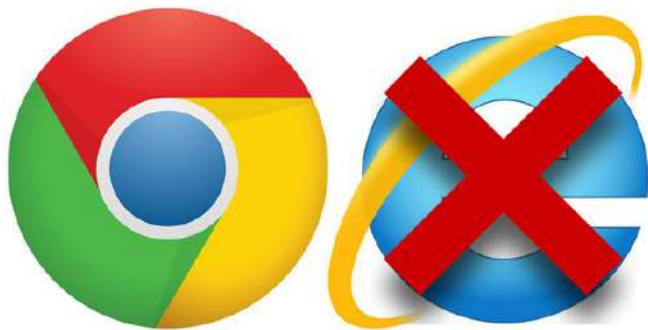


Negotiation  
and Conflict Resolution

# Aceitar mudanças

---

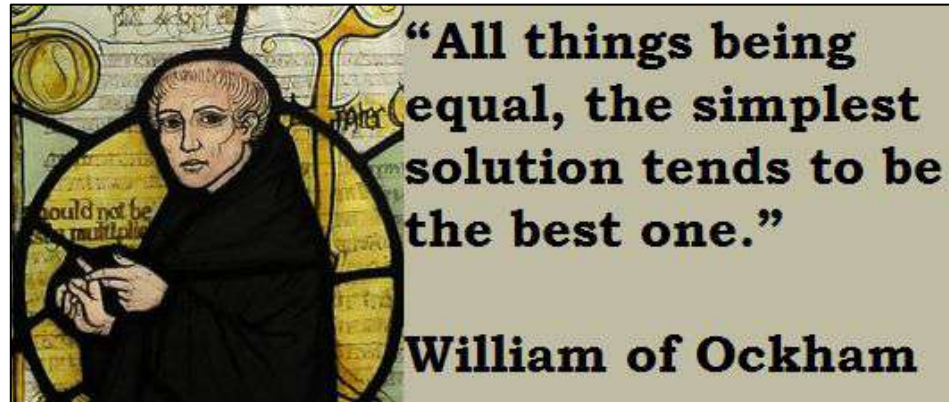
- Projetar o sistema de maneira a acomodar as mudanças, que são inevitáveis em qualquer projeto de sistema
  - Sistemas em evolução tem mudanças constantes em seus requisitos e funcionalidades



# Manter a simplicidade

---

- Foco na simplicidade, tanto do software a ser desenvolvido quanto do processo de desenvolvimento
- Eliminar as complexidades do sistema e dos processos sempre que possível



# Exercício

Considerando o desenvolvimento incremental, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Os processos são sempre intercalados.

II - O sistema é desenvolvido através de versões.

III - Os usuários finais não estão envolvidos no processo de especificação e avaliação das versões.

IV - Cada versão contém somente um conjunto de funcionalidades bem definidas.

- ☐ Todas as assertivas são verdadeiras.
- ☐ Todas as assertivas são falsas.
- ☐ Somente I, II e IV.
- ☐ Somente II.
- ☐ Nenhuma das alternativas anteriores.

# Exercício

Considerando o desenvolvimento incremental, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Os processos são sempre intercalados. **V**

II - O sistema é desenvolvido através de versões.

III - Os usuários finais não estão envolvidos no processo de especificação e avaliação das versões.

IV - Cada versão contém somente um conjunto de funcionalidades bem definidas.

- ☐ Todas as assertivas são verdadeiras.
- ☐ Todas as assertivas são falsas.
- ☐ Somente I, II e IV.
- ☐ Somente II.
- ☐ Nenhuma das alternativas anteriores.



# Exercício

Considerando o desenvolvimento incremental, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Os processos são sempre intercalados. **V**

II - O sistema é desenvolvido através de versões. **V**

III - Os usuários finais não estão envolvidos no processo de especificação e avaliação das versões.

IV - Cada versão contém somente um conjunto de funcionalidades bem definidas.

- ☐ Todas as assertivas são verdadeiras.
- ☐ Todas as assertivas são falsas.
- ☐ Somente I, II e IV.
- ☐ Somente II.
- ☐ Nenhuma das alternativas anteriores.

# Exercício

Considerando o desenvolvimento incremental, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Os processos são sempre intercalados. **V**

II - O sistema é desenvolvido através de versões. **V**

III - Os usuários finais **não estão envolvidos** no processo de especificação e avaliação das versões. **F**

IV - Cada versão contém somente um conjunto de funcionalidades bem definidas.

- ☐ Todas as assertivas são verdadeiras.
- ☐ Todas as assertivas são falsas.
- ☐ Somente I, II e IV.
- ☐ Somente II.
- ☐ Nenhuma das alternativas anteriores.

# Exercício


Considerando o desenvolvimento incremental, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Os processos são sempre intercalados. **V**

II - O sistema é desenvolvido através de versões. **V**

III - Os usuários finais **não estão envolvidos** no processo de especificação e avaliação das versões. **F**

IV - Cada versão contém somente um conjunto de funcionalidades bem definidas. **V**

- ☐ Todas as assertivas são verdadeiras.
- ☐ Todas as assertivas são falsas.
-  ☒ Somente I, II e IV.
- ☐ Somente II.
- ☐ Nenhuma das alternativas anteriores.

# Métodos ágeis

---

- Há uma ênfase maior na escrita de software de alta qualidade
  - A documentação formal de um sistema nem sempre é atualizada
  - Essa documentação pode não refletir exatamente o código do programa
  - Logo, a documentação é vista como “perda de tempo”
  - Softwares fáceis de serem mantidos (manuteníveis) são resultado da produção de códigos de alta qualidade, legíveis

# Métodos ágeis

---

- A principal dificuldade após a entrega do software é manter o envolvimento dos clientes no processo
  - Um cliente pode justificar o envolvimento de um representante em tempo integral, durante o desenvolvimento do sistema
  - Porém, **durante a manutenção**, representantes do cliente são propensos a perder o interesse no sistema
- **Solução:** buscar formas de engajamento do cliente com o projeto

# Métodos ágeis

---

- Os métodos ágeis mais conhecidos são:
  - Extreme programming
  - Scrum
  - Crystal
  - Desenvolvimento de Software Adaptativo
  - DSDM
  - Desenvolvimento Dirigido a Características

# Métodos ágeis

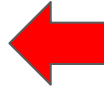
---

- Os métodos ágeis mais conhecidos são:

- **Extreme programming**



- **Scrum**



**Focos desta disciplina**

- Crystal

- Desenvolvimento de Software Adaptativo

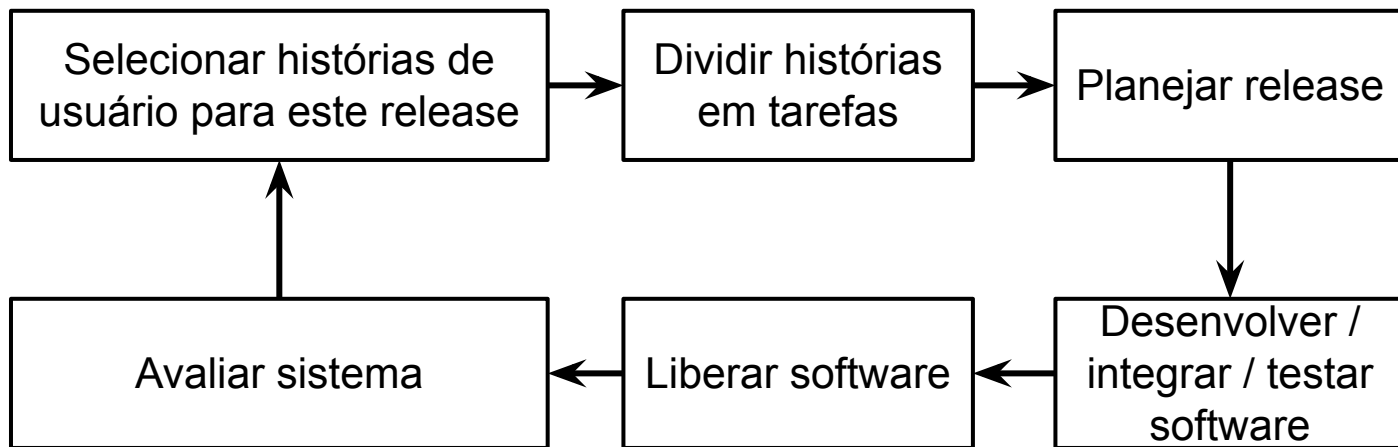
- DSDM

- Desenvolvimento Dirigido a Características

# Extreme programming (XP)

---

- É o mais conhecido e mais utilizado método ágil
- Baseado em versões (*releases*), que possuem o seguinte ciclo:





# Extreme programming (XP)

---

Selecionar histórias de  
usuário para este release

**Histórias de usuário:**  
cenários de uso

# Extreme programming (XP)

---

Selecionar histórias de  
usuário para este release

Os requisitos são gravados  
em **cartões de história**

# Extreme programming (XP)

---

Selecionar histórias de usuário para este release

Como os requisitos mudam, as histórias não implementadas mudam ou podem ser descartadas

# Exemplo de Cartão de história

## Prescrição de medicamentos

Kate é uma médica que deseja prescrever medicamentos para um paciente de uma clínica. O prontuário do paciente já está sendo exibido em seu computador, assim, ela clica o campo 'medicação' e pode selecionar 'medicação atual', 'nova medicação', ou 'formulário'.

Se ela selecionar 'medicação atual', o sistema pede que ela verifique a dose. Se ela quiser mudar a dose, ela altera esta e em seguida, confirma a prescrição.

Se ela escolher 'nova medicação', o sistema assume que ela sabe qual medicação receitar.

Ela digita as primeiras letras do nome do medicamento. O sistema exibe uma lista de possíveis fármacos que começam com essas letras. Ela escolhe a medicação requerida e o sistema responde, pedindo-lhe para verificar se o medicamento selecionado está correto.

Ela insere a dose e, em seguida, confirma a prescrição.

Se ela escolhe 'formulário', o sistema exibe uma caixa de busca para o formulário aprovado.

Ela pode, então, procurar pelo medicamento requerido. Ela seleciona um medicamento e é solicitado que verifique se a medicação está correta. Ela insere a dose e, em seguida, confirma a prescrição.

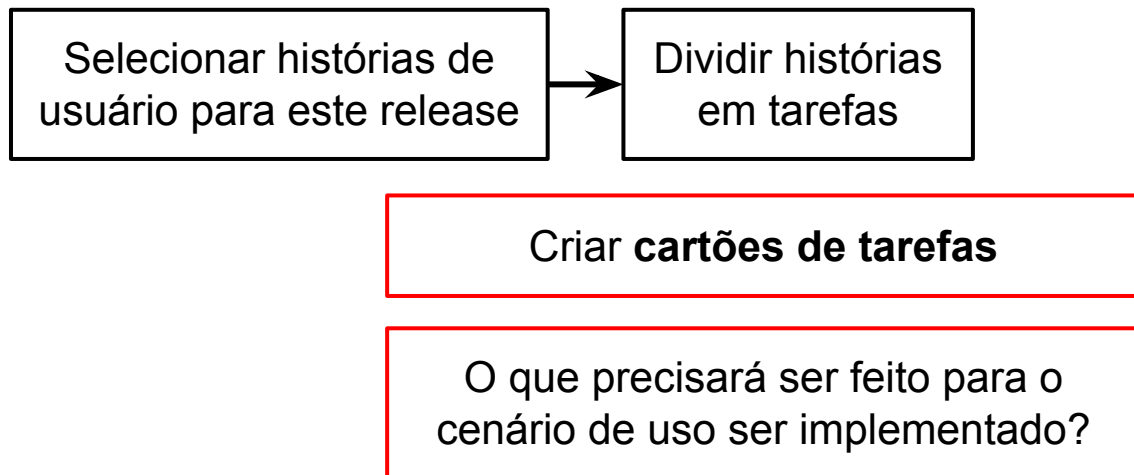
O sistema sempre verifica se a dose está dentro da faixa permitida. Caso não esteja, Kate é convidada a alterar a dose.

Após Kate confirmar a prescrição, esta será exibida para verificação. Ela pode escolher 'OK' ou 'Alterar'. Se clicar em 'OK', a prescrição fica gravada nos bancos de dados da auditoria.

Se ela clicar em 'Alterar', reinicia o processo de 'Prescrição de Medicamentos'.

# Extreme programming (XP)

---



# Exemplo de Cartões de Tarefas

---

**Tarefa 1: Alterar dose de medicamentos prescritos**

**Tarefa 2: Seleção de formulário**

**Tarefa 3: Verificação de dose**

A verificação da dose é uma precaução de segurança para verificar se o médico não receitou uma dose perigosamente pequena ou grande.

Usando o ID do formulário para o nome do medicamento genérico, procure o formulário e obtenha a dose mínima e máxima recomendada.

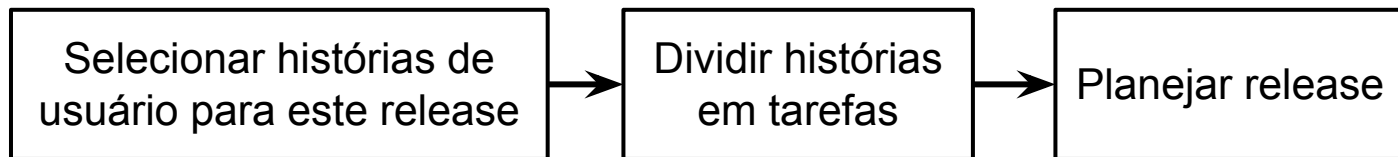
Verifique a dose mínima e máxima prescrita. Caso esteja fora da faixa, emita uma mensagem de erro dizendo que a dose está muito alta ou muito baixa.

Caso esteja dentro da faixa, habilite o botão 'Confirmar'.

# Extreme programming (XP)

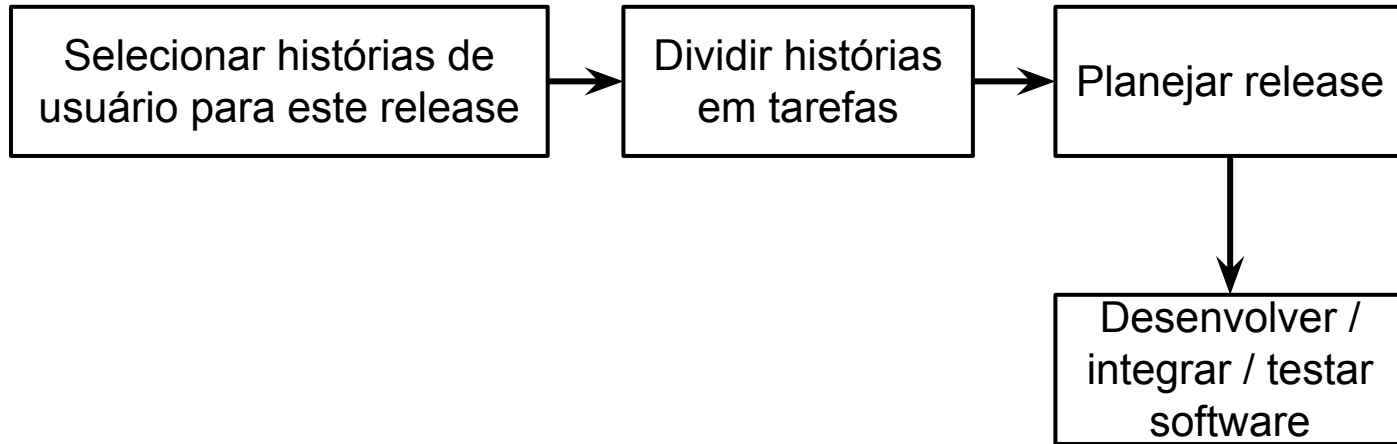
---

Quando cada tarefa será realizada?  
Qual a prioridade de cada uma?



# Extreme programming (XP)

---



Os programadores trabalham em pares e desenvolvem testes para cada tarefa **ANTES** de escreverem o código



# Exemplo de caso de teste

---

## Teste 4: Verificação de dose

### Entrada:

1. Um número em mg representando uma única dose da medicação.
2. Um número que representa o número de doses únicas por dia.

### Testes:

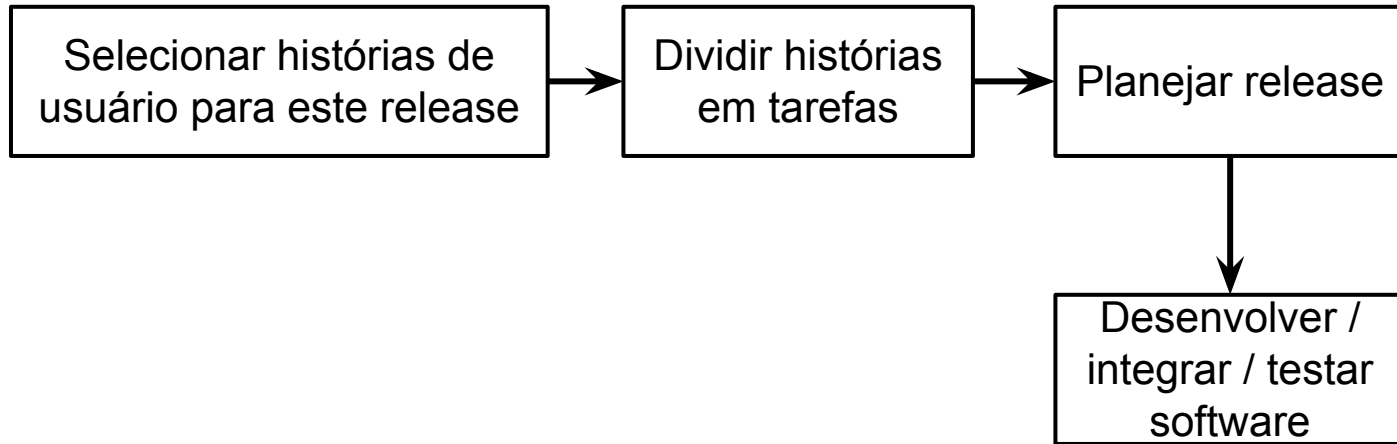
1. Teste para entradas em que a dose única é correta, mas a frequência é muito alta.
2. Teste para entradas em que a única dose é muito alta e muito baixa.
3. Teste para entradas em que a dose única x frequência é muito alta e muito baixa.
4. Teste para entradas em que a dose única x frequência é permitida.

### Saída:

Mensagem de OK ou erro indicando que a dose está fora da faixa de segurança.

# Extreme programming (XP)

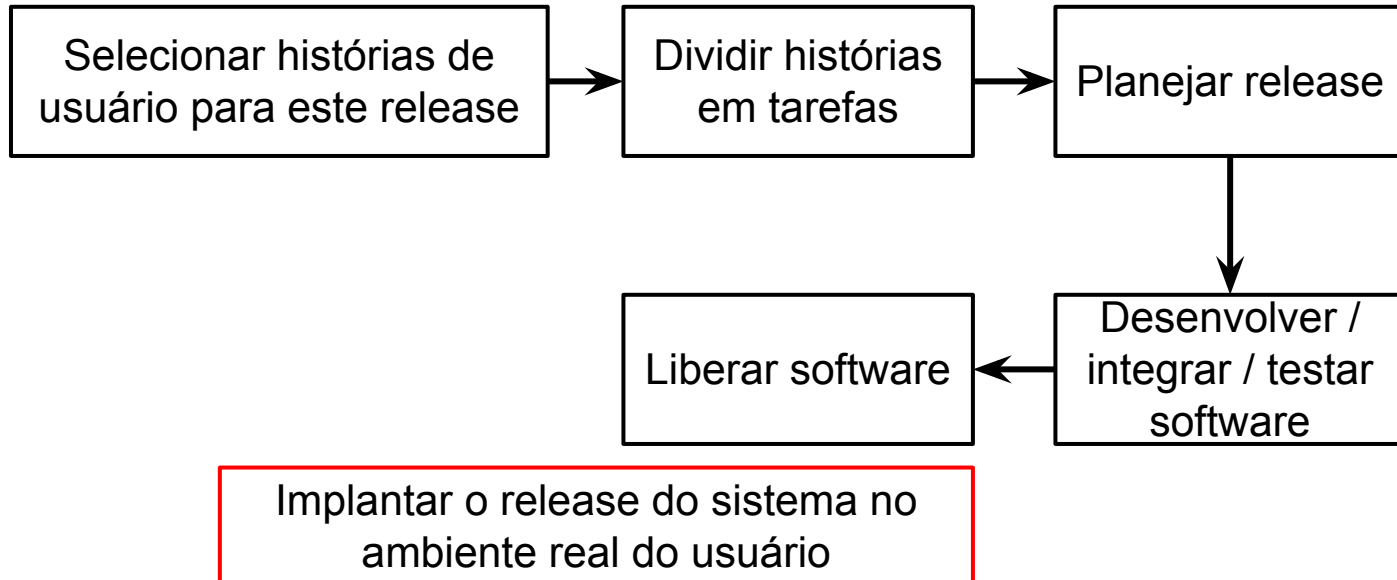
---



Quando o novo código é integrado no sistema, todos os testes devem ser **executados novamente** com sucesso

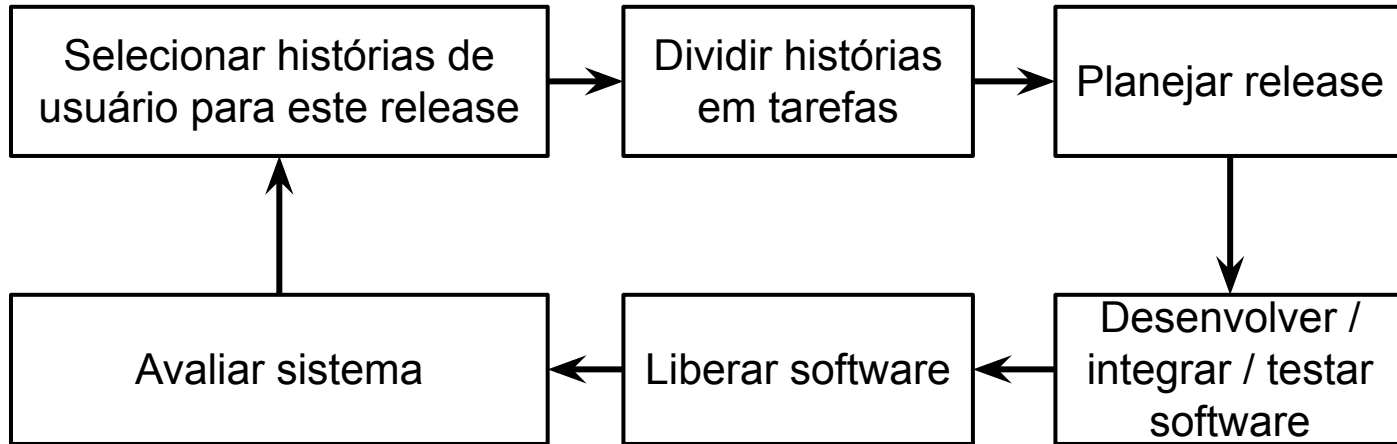
# Extreme programming (XP)

---



# Extreme programming (XP)

---



# Exercício

Considerando os princípios dos métodos ágeis, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - A entrega incremental consiste no desenvolvimento do software em incrementos ou versões, com o auxílio do cliente.

II - O princípio do "foco nas pessoas e não nos processos" estabelece a exploração e reconhecimento das habilidades da equipe de desenvolvimento. Isto é, os membros da equipe desenvolvem suas próprias formas de trabalhar, sem seguir processos obrigatórios.

III - Deve-se projetar o sistema para acomodar mudanças, que são inevitáveis em projetos de sistemas.

IV - Deve-se manter o foco na simplicidade do software e do processo de desenvolvimento, eliminando complexidades.

- ☐ Todas as assertivas são verdadeiras.
- ☐ Somente I.
- ☐ Somente III.
- ☐ Somente IV.
- ☐ Nenhuma das alternativas anteriores.

# Exercício

Considerando os princípios dos métodos ágeis, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - A entrega incremental consiste no desenvolvimento do software em incrementos ou versões, com o auxílio do cliente. **V**

II - O princípio do "foco nas pessoas e não nos processos" estabelece a exploração e reconhecimento das habilidades da equipe de desenvolvimento. Isto é, os membros da equipe desenvolvem suas próprias formas de trabalhar, sem seguir processos obrigatórios.

III - Deve-se projetar o sistema para acomodar mudanças, que são inevitáveis em projetos de sistemas.

IV - Deve-se manter o foco na simplicidade do software e do processo de desenvolvimento, eliminando complexidades.

- ☐ Todas as assertivas são verdadeiras.
- ☐ Somente I.
- ☐ Somente III.
- ☐ Somente IV.
- ☐ Nenhuma das alternativas anteriores.

# Exercício

Considerando os princípios dos métodos ágeis, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - A entrega incremental consiste no desenvolvimento do software em incrementos ou versões, com o auxílio do cliente. **V**

II - O princípio do "foco nas pessoas e não nos processos" estabelece a exploração e reconhecimento das habilidades da equipe de desenvolvimento. Isto é, os membros da equipe desenvolvem suas próprias formas de trabalhar, sem seguir processos obrigatórios. **V**

III - Deve-se projetar o sistema para acomodar mudanças, que são inevitáveis em projetos de sistemas.

IV - Deve-se manter o foco na simplicidade do software e do processo de desenvolvimento, eliminando complexidades.

- ☐ Todas as assertivas são verdadeiras.
- ☐ Somente I.
- ☐ Somente III.
- ☐ Somente IV.
- ☐ Nenhuma das alternativas anteriores.

# Exercício

Considerando os princípios dos métodos ágeis, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - A entrega incremental consiste no desenvolvimento do software em incrementos ou versões, com o auxílio do cliente. **V**

II - O princípio do "foco nas pessoas e não nos processos" estabelece a exploração e reconhecimento das habilidades da equipe de desenvolvimento. Isto é, os membros da equipe desenvolvem suas próprias formas de trabalhar, sem seguir processos obrigatórios. **V**

III - Deve-se projetar o sistema para acomodar mudanças, que são inevitáveis em projetos de sistemas. **V**

IV - Deve-se manter o foco na simplicidade do software e do processo de desenvolvimento, eliminando complexidades.

- ☐ Todas as assertivas são verdadeiras.
- ☐ Somente I.
- ☐ Somente III.
- ☐ Somente IV.
- ☐ Nenhuma das alternativas anteriores.



# Exercício

Considerando os princípios dos métodos ágeis, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - A entrega incremental consiste no desenvolvimento do software em incrementos ou versões, com o auxílio do cliente. **V**

II - O princípio do "foco nas pessoas e não nos processos" estabelece a exploração e reconhecimento das habilidades da equipe de desenvolvimento. Isto é, os membros da equipe desenvolvem suas próprias formas de trabalhar, sem seguir processos obrigatórios. **V**

III - Deve-se projetar o sistema para acomodar mudanças, que são inevitáveis em projetos de sistemas. **V**

IV - Deve-se manter o foco na simplicidade do software e do processo de desenvolvimento, eliminando complexidades. **V**

- ☒ Todas as assertivas são verdadeiras.
- ☐ Somente I.
- ☐ Somente III.
- ☐ Somente IV.
- ☐ Nenhuma das alternativas anteriores.

# Extreme programming (XP)

---

- *Releases* contínuos para os clientes
  - Pequenos e frequentes *releases* do sistema (versões)
- Requisitos baseados em histórias de usuários
  - **Objetivo:** decidir a funcionalidade que deve ser incluída na versão do sistema
- Engajamento contínuo do cliente com a equipe de desenvolvimento
  - Cliente define os **testes de aceitação** para o sistema
  - **Teste de aceitação:** sistema atende a necessidade do cliente?

# Extreme programming (XP)

---

- Desenvolvimento test-first
  - “*Cria o teste primeiro, faz a implementação depois*”
  - Uso de **framework de testes automatizados**
    - Facilita o desenvolvimento de testes
  - Testes de componentes e unidades
    - Simular a submissão de entrada a ser testada e verificar se o resultado atende à especificação de saída

# Desenvolvimento test-first

---

- Não leva, necessariamente, a testes completos do programa, pois:
  - Programadores podem tomar atalhos ao escrever testes
    - Testes incompletos, que não verificam todas as possíveis exceções que podem ocorrer
  - Alguns testes podem ser muito difíceis de escrever de forma incremental
    - **Ex:** interface de usuário complexa

# Desenvolvimento test-first

---

- Não leva, necessariamente, a testes completos do programa, pois:
  - O conjunto de testes pode estar incompleto
    - Partes essenciais do sistema podem não ser executadas e, assim, não são testadas

# Extreme programming (XP)

---

- Integração contínua de novas funcionalidades
  - Assim que uma tarefa é concluída, a funcionalidade é integrada ao sistema como um todo
  - Após essa integração, todos os testes de unidade do sistema devem ser executados novamente com sucesso
- Ritmo sustentável
  - Grandes quantidades de horas-extra não são aceitáveis
  - Horas-extras => redução da qualidade do código e da produtividade a médio prazo

# Spike

- Protótipo ou desenvolvimento-teste, criado sob demanda, para entender um problema e/ou uma solução
- *Spikes* são incrementos em que nenhum tipo de programação é realizado



# Extreme programming (XP)

---

- Manutenção da simplicidade
  - Não antecipar desnecessariamente futuras mudanças no sistema
  - Evitar funcionalidades não solicitadas
- Programação em pares
  - Desenvolvedores trabalham em pares, sentados lado-a-lado
  - Um escreve o código enquanto o outro revisa em tempo real
    - A escolha de quem escreve e quem revisa é **dinâmica**



# Exercício

Considerando o método XP, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Releases contínuos, pequenos e frequentes, são fornecidos para os clientes.

II - Necessita de engajamento contínuo do cliente com a equipe de desenvolvimento.

III - O cliente define os testes de aceitação para o sistema.

IV - A funcionalidade a ser incluída em cada versão do sistema é decidida de acordo com as histórias de usuários.

- ☐ Somente I.
- ☐ Somente I e II.
- ☐ Somente II e IV.
- ☐ Somente I, II e III.
- ☐ Nenhuma das alternativas anteriores.

# Exercício

Considerando o método XP, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Releases contínuos, pequenos e frequentes, são fornecidos para os clientes. **V**

II - Necessita de engajamento contínuo do cliente com a equipe de desenvolvimento.

III - O cliente define os testes de aceitação para o sistema.

IV - A funcionalidade a ser incluída em cada versão do sistema é decidida de acordo com as histórias de usuários.

- ☐ Somente I.
- ☐ Somente I e II.
- ☐ Somente II e IV.
- ☐ Somente I, II e III.
- ☐ Nenhuma das alternativas anteriores.

# Exercício

Considerando o método XP, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Releases contínuos, pequenos e frequentes, são fornecidos para os clientes. **V**

II - Necessita de engajamento contínuo do cliente com a equipe de desenvolvimento. **V**

III - O cliente define os testes de aceitação para o sistema.

IV - A funcionalidade a ser incluída em cada versão do sistema é decidida de acordo com as histórias de usuários.

- ☐ Somente I.
- ☐ Somente I e II.
- ☐ Somente II e IV.
- ☐ Somente I, II e III.
- ☐ Nenhuma das alternativas anteriores.

# Exercício

Considerando o método XP, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Releases contínuos, pequenos e frequentes, são fornecidos para os clientes. **V**

II - Necessita de engajamento contínuo do cliente com a equipe de desenvolvimento. **V**

III - O cliente define os testes de aceitação para o sistema. **V**

IV - A funcionalidade a ser incluída em cada versão do sistema é decidida de acordo com as histórias de usuários.

☐ Somente I.

☐ Somente I e II.

☐ Somente II e IV.

☐ Somente I, II e III.

☐ Nenhuma das alternativas anteriores.

# Exercício

Considerando o método XP, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Releases contínuos, pequenos e frequentes, são fornecidos para os clientes. **V**

II - Necessita de engajamento contínuo do cliente com a equipe de desenvolvimento. **V**

III - O cliente define os testes de aceitação para o sistema. **V**


IV - A funcionalidade a ser incluída em cada versão do sistema é decidida de acordo com as histórias de usuários. **V**

☐ Somente I.

☐ Somente I e II.

☐ Somente II e IV.

☐ Somente I, II e III.

 ☒ Nenhuma das alternativas anteriores.

# Vantagens da Programação em pares

---

- Suporte à ideia de **propriedade e responsabilidade** coletiva
  - Programação sem ego (WEINBERG, 1971)
    - Software é de propriedade da equipe como um todo
    - A equipe tem responsabilidade coletiva
      - Os indivíduos não são responsabilizados por problemas com o código

# Vantagens da Programação em pares

---

- Atua como um processo de revisão informal
  - Processo de inspeção muito mais barato do que inspeções formais de programa
  - Cada linha de código é observada por, pelo menos, duas pessoas
- Suporte à refatoração em tempo-real
  - *“Uma escreve o código enquanto o outro sugere melhorias”*

# Mas a programação em pares é eficiente?

---

- Você pode pensar que a programação em pares é menos eficiente do que a individual
  - Em um determinado período de tempo, um par de desenvolvedores produziria metade da quantidade de código que dois indivíduos trabalhando sozinhos
  - **Mas será que essa afirmação é válida?**
    - A resposta é mais complexa do que um simples SIM ou NÃO!



# Estudo de **Williams** - Eficiência da programação em pares

---

- Usando estudantes voluntários, Williams e seus colaboradores (COCKBURN e WILLIAMS, 2001; WILLIAMS et al., 2000) concluíram que:
  - A **produtividade** na programação em pares parece ser **comparável** com a de duas pessoas que trabalham de forma independente
    - **Porque?**

# Estudo de **Williams** - Eficiência da programação em pares

---

- A **produtividade** na programação em pares parece ser **comparável** com a de duas pessoas que trabalham de forma independente
  - **Pares discutem o software antes do desenvolvimento**
    - Eles cometem menos erros de projeto e, conseqüentemente, têm menos retrabalho

# Estudo de **Williams** - Eficiência da programação em pares

---

- A **produtividade** na programação em pares parece ser **comparável** com a de duas pessoas que trabalham de forma independente
  - **Pares discutem o software antes do desenvolvimento**
    - Eles cometem menos erros de projeto e, conseqüentemente, têm menos retrabalho
  - **A inspeção informal evita o surgimento de erros**
    - Reduz o tempo gasto consertando defeitos descobertos durante o processo de teste
- **Mas será que essa é a resposta definitiva?**

# Estudo de **Arisholm** - Eficiência da programação em pares

---

- Estudos com programadores mais experientes (ARISHOLM et al., 2007; PARRISH et al., 2004) mostram que houve:
  - **Benefícios na qualidade**, mas estes não compensaram o *overhead* da programação em pares.
  - Houve **perda significativa de produtividade** em comparação com dois programadores trabalhando sozinhos

# Resposta final -

## Eficiência da programação em pares

---

- Benefício na qualidade do software construído
- A eficiência da programação em pares varia a depender da experiência da equipe de desenvolvedores
- Apesar disso, o **compartilhamento de conhecimento** da programação em pares é muito importante
  - Reduz os riscos globais para um projeto quando um dos membros da equipe resolve se desligar do projeto
  - **Este aspecto por si só já faz a programação em pares ser válida**

# Exercício

Considerando o método XP, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Programação em pares é utilizada no extreme programming, de forma que um programa enquanto outro senta a seu lado, sem fazer nada.

II - Spikes são incrementos nos quais nenhum tipo de programação é necessário.

III - O método XP não tenta antever mudanças futuras no sistema.

IV - O programador responsável pela escrita do código é feita de maneira dinâmica na programação em pares.

- ☐ Somente II.
- ☐ Somente II, III e IV.
- ☐ Somente II e IV.
- ☐ Somente III.
- ☐ Nenhuma das alternativas anteriores.

# Exercício

Considerando o método XP, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Programação em pares é utilizada no extreme programming, de forma que um programa enquanto outro senta a seu lado, **sem fazer nada.** **F**

II - Spikes são incrementos nos quais nenhum tipo de programação é necessário.

III - O método XP não tenta antever mudanças futuras no sistema.

IV - O programador responsável pela escrita do código é feita de maneira dinâmica na programação em pares.

- ☐ Somente II.
- ☐ Somente II, III e IV.
- ☐ Somente II e IV.
- ☐ Somente III.
- ☐ Nenhuma das alternativas anteriores.

# Exercício

Considerando o método XP, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Programação em pares é utilizada no extreme programming, de forma que um programa enquanto outro senta a seu lado, **sem fazer nada.** **F**

II - Spikes são incrementos nos quais nenhum tipo de programação é necessário. **V**

III - O método XP não tenta antever mudanças futuras no sistema.

IV - O programador responsável pela escrita do código é feita de maneira dinâmica na programação em pares.

- ☐ Somente II.
- ☐ Somente II, III e IV.
- ☐ Somente II e IV.
- ☐ Somente III.
- ☐ Nenhuma das alternativas anteriores.



# Exercício

Considerando o método XP, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Programação em pares é utilizada no extreme programming, de forma que um programa enquanto outro senta a seu lado, **sem fazer nada.** **F**

II - Spikes são incrementos nos quais nenhum tipo de programação é necessário. **V**

III - O método XP não tenta antever mudanças futuras no sistema. **V**

IV - O programador responsável pela escrita do código é feita de maneira dinâmica na programação em pares.

- ☐ Somente II.
- ☐ Somente II, III e IV.
- ☐ Somente II e IV.
- ☐ Somente III.
- ☐ Nenhuma das alternativas anteriores.

# Exercício

Considerando o método XP, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Programação em pares é utilizada no extreme programming, de forma que um programa enquanto outro senta a seu lado, sem fazer nada. **F**

II - Spikes são incrementos nos quais nenhum tipo de programação é necessário. **V**

III - O método XP não tenta antever mudanças futuras no sistema. **V**

IV - O programador responsável pela escrita do código é feita de maneira dinâmica na programação em pares. **V**

- ☐ Somente II.
- ☒ Somente II, III e IV.
- ☐ Somente II e IV.
- ☐ Somente III.
- ☐ Nenhuma das alternativas anteriores.

# Extreme programming (XP)

---

- O desenvolvimento incremental tende a degradar a estrutura do software
  - Código duplicado
  - Reuso inadequado
  - Projeto e estrutura geral do sistema degradada
- Refatoração constante para evitar a degeneração do código
  - Melhoria na qualidade do código

# Refatoração

---

- Equipe foca em possíveis melhorias para o software e na implementação imediata destas
  - **Ex:**
    - Reorganização da hierarquia de classes para eliminação de código duplicado
    - Arrumação e renomeação de atributos e métodos
    - Substituição do código (métodos definidos em uma biblioteca de programas)

# Exemplo de Refatoração na IDE Eclipse

The screenshot displays the Eclipse IDE interface. On the left, a Java class is open in the editor, showing several methods. The code is as follows:

```
10  
11 public String getDetails() {  
12     return "Car [licensePlate=" + licensePlate + ", d  
13 }  
14  
15 public String getlicensePlate() {  
16     return licensePlate;  
17 }  
18  
19 public void setlicensePlate(String licensePlate) {  
20     this.licensePlate = licensePlate;  
21 }  
22  
23 public String getDriverName() {  
24     return driverName;  
25 }  
26  
27 public void setDriverName(String driverName) {  
28     this.driverName = driverName;  
29 }  
30  
31 public String getDriverLicense() {  
32     return driverLicense;  
33 }  
34  
35 public void setDriverLicense(String driverLicense) {  
36     this.driverLicense = driverLicense;  
37 }  
38  
39 }  
40
```

In the center, a context menu is open, listing various actions. The 'Refactor' option is highlighted in blue. The menu items and their shortcuts are:

- Undo Typing (Ctrl+Z)
- Revert File
- Save (Ctrl+S)
- Open Declaration (F3)
- Open Type Hierarchy (F4)
- Open Call Hierarchy (Ctrl+Alt+H)
- Show in Breadcrumb (Alt+Shift+B)
- Quick Outline (Ctrl+O)
- Quick Type Hierarchy (Ctrl+T)
- Open With (>)
- Show In (Alt+Shift+W >)
- Cut (Ctrl+X)
- Copy (Ctrl+C)
- Copy Qualified Name
- Paste (Ctrl+V)
- Quick Fix (Ctrl+1)
- Source (Alt+Shift+S >)
- Refactor (Alt+Shift+T >)**
- Local History (>)
- References (>)
- Declarations (>)
- Add to Snippets...
- Coverage As (>)
- Run As (>)
- Debug As (>)
- Profile As (>)

On the right, a sub-menu for 'Refactor' is open, showing options like 'Move...', 'Extract Interface...', 'Extract Superclass...', 'Use Supertype Where Possible...', 'Pull Up...', 'Push Down...', 'Extract Class...' (highlighted in blue), and 'Infer Generic Type Arguments...'.

# Gerenciamento ágil de projetos

---

- Os gerentes de projeto tem como atribuições (responsabilidades):
  - Supervisionar o trabalho dos engenheiros de software
  - Acompanhar o desenvolvimento do software
  - Garantir que o software seja entregue no prazo
  - Garantir que o custo total do projeto esteja dentro do orçamento previsto
- **Como?**

# Scrum (SCHWABER e BEEDLE, 2001)

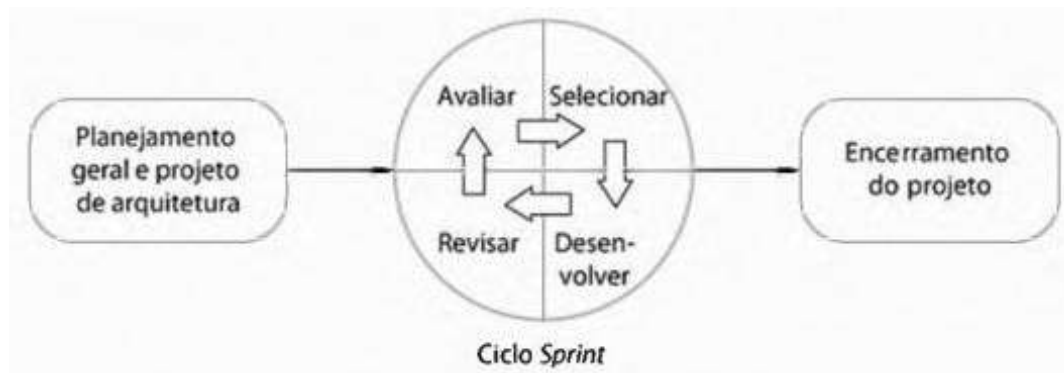
---

- Método ágil geral, cujo foco está no gerenciamento do desenvolvimento iterativo
  - ***Framework* de gerenciamento de projeto**
- Não substitui as boas práticas de programação, como programação em pares e desenvolvimento test-first
  - **Pode ser usado com abordagens ágeis mais técnicas, como XP**

# Scrum (SCHWABER e BEEDLE, 2001)

---

- Existem as seguintes fases no Scrum:
  - **Fase de planejamento geral**
  - **Ciclos de *sprint***
  - **Encerramento**





# *Sprint*

---

- Uma unidade de planejamento na qual
  - O trabalho a ser feito é avaliado
  - Os recursos para o desenvolvimento são selecionados
  - O software é implementado
- Sprints são de comprimento fixo, normalmente duas a quatro semanas

# Exercício

Considerando o desenvolvimento incremental e os métodos ágeis, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - A refatoração é uma técnica que visa fornecer melhorias para o software. Exemplos de uso desta abordagem é a renomeação de atributos e métodos e eliminação de código duplicado.

II - É papel do gerente de projeto: supervisionar o trabalho dos engenheiros, acompanhar o desenvolvimento do software, garantir a entrega dentro dos prazos, e garantir que o custo do projeto esteja dentro do orçamento previsto.

III - O SCRUM é um método ágil cujo foco está no gerenciamento de projetos baseados em métodos ágeis.

IV - Não é possível utilizar o SCRUM em conjunto com outros métodos ágeis.

- ☐ Somente I e II.
- ☐ Somente I, II e III.
- ☐ Somente I e III.
- ☐ Somente IV.
- ☐ Nenhuma das alternativas anteriores.

# Exercício

Considerando o desenvolvimento incremental e os métodos ágeis, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - A refatoração é uma técnica que visa fornecer melhorias para o software. Exemplos de uso desta abordagem é a renomeação de atributos e métodos e eliminação de código duplicado. **V**

II - É papel do gerente de projeto: supervisionar o trabalho dos engenheiros, acompanhar o desenvolvimento do software, garantir a entrega dentro dos prazos, e garantir que o custo do projeto esteja dentro do orçamento previsto.

III - O SCRUM é um método ágil cujo foco está no gerenciamento de projetos baseados em métodos ágeis.

IV - Não é possível utilizar o SCRUM em conjunto com outros métodos ágeis.

- ☐ Somente I e II.
- ☐ Somente I, II e III.
- ☐ Somente I e III.
- ☐ Somente IV.
- ☐ Nenhuma das alternativas anteriores.

# Exercício

Considerando o desenvolvimento incremental e os métodos ágeis, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - A refatoração é uma técnica que visa fornecer melhorias para o software. Exemplos de uso desta abordagem é a renomeação de atributos e métodos e eliminação de código duplicado. **V**

II - É papel do gerente de projeto: supervisionar o trabalho dos engenheiros, acompanhar o desenvolvimento do software, garantir a entrega dentro dos prazos, e garantir que o custo do projeto esteja dentro do orçamento previsto. **V**

III - O SCRUM é um método ágil cujo foco está no gerenciamento de projetos baseados em métodos ágeis.

IV - Não é possível utilizar o SCRUM em conjunto com outros métodos ágeis.

- ☐ Somente I e II.
- ☐ Somente I, II e III.
- ☐ Somente I e III.
- ☐ Somente IV.
- ☐ Nenhuma das alternativas anteriores.

# Exercício

Considerando o desenvolvimento incremental e os métodos ágeis, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - A refatoração é uma técnica que visa fornecer melhorias para o software. Exemplos de uso desta abordagem é a renomeação de atributos e métodos e eliminação de código duplicado. **V**

II - É papel do gerente de projeto: supervisionar o trabalho dos engenheiros, acompanhar o desenvolvimento do software, garantir a entrega dentro dos prazos, e garantir que o custo do projeto esteja dentro do orçamento previsto. **V**

III - O SCRUM é um método ágil cujo foco está no gerenciamento de projetos baseados em métodos ágeis. **V**

IV - Não é possível utilizar o SCRUM em conjunto com outros métodos ágeis.

- ☐ Somente I e II.
- ☐ Somente I, II e III.
- ☐ Somente I e III.
- ☐ Somente IV.
- ☐ Nenhuma das alternativas anteriores.

# Exercício


Considerando o desenvolvimento incremental e os métodos ágeis, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - A refatoração é uma técnica que visa fornecer melhorias para o software. Exemplos de uso desta abordagem é a renomeação de atributos e métodos e eliminação de código duplicado. **V**

II - É papel do gerente de projeto: supervisionar o trabalho dos engenheiros, acompanhar o desenvolvimento do software, garantir a entrega dentro dos prazos, e garantir que o custo do projeto esteja dentro do orçamento previsto. **V**

III - O SCRUM é um método ágil cujo foco está no gerenciamento de projetos baseados em métodos ágeis. **V**

IV - Não é possível utilizar o SCRUM em conjunto com outros métodos ágeis. **F**

- ☐ Somente I e II.
-  ☒ Somente I, II e III.
- ☐ Somente I e III.
- ☐ Somente IV.
- ☐ Nenhuma das alternativas anteriores.

# Scrum (SCHWABER e BEEDLE, 2001)

---

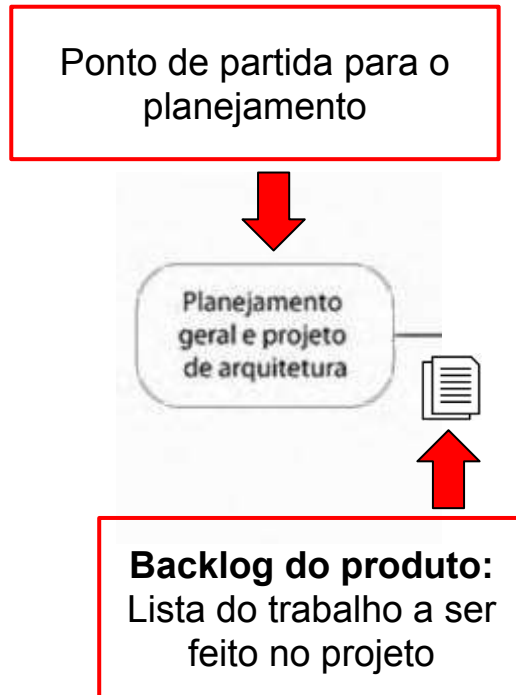
Ponto de partida para o  
planejamento



Planejamento  
geral e projeto  
de arquitetura

# Scrum (SCHWABER e BEEDLE, 2001)

---



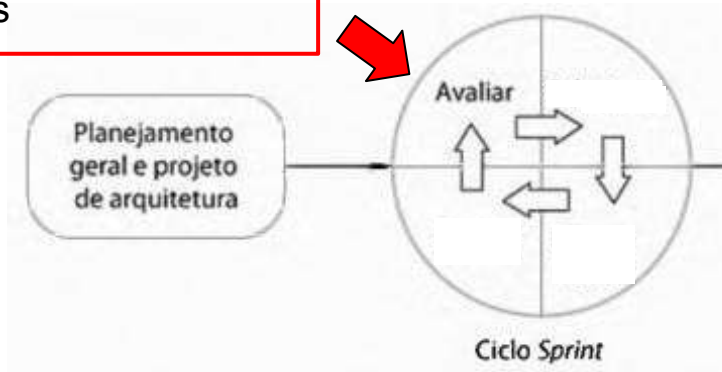


# Scrum (SCHWABER e BEEDLE, 2001)

---

## **Fase de avaliação do *sprint*:**

- Revisão do *sprint*,
- Prioridades e riscos são identificados



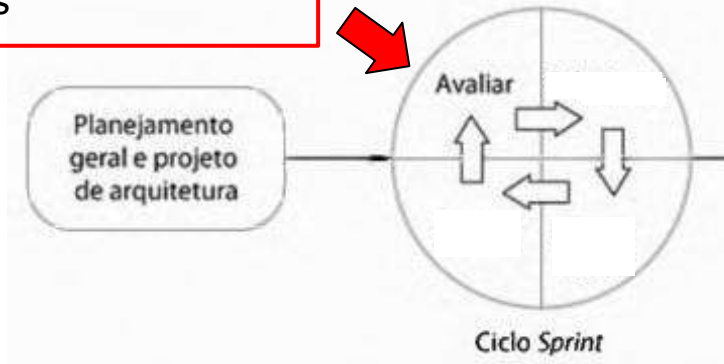
# Scrum (SCHWABER e BEEDLE, 2001)

---

## Fase de avaliação do *sprint*:

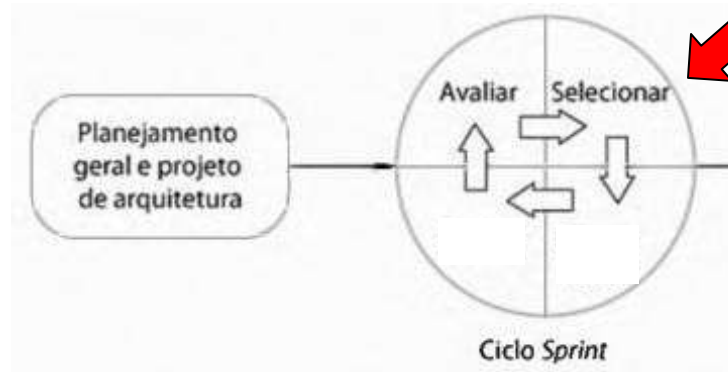
- Revisão do *sprint*,
- Prioridades e riscos são identificados

Cliente pode introduzir novos requisitos ou tarefas



# Scrum (SCHWABER e BEEDLE, 2001)

---

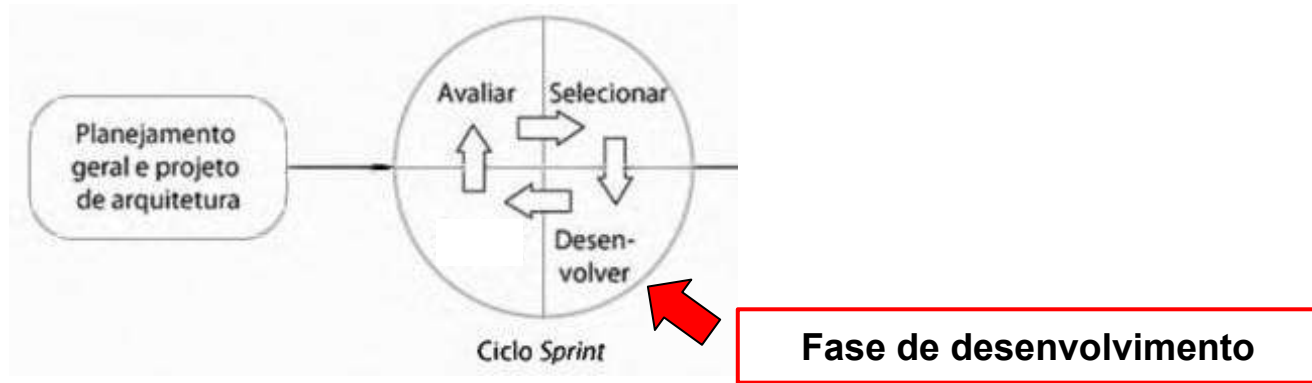


**Fase de seleção:** selecionar os recursos e a funcionalidade a ser desenvolvida

Envolve todos da equipe do projeto que trabalham com o cliente

# Scrum (SCHWABER e BEEDLE, 2001)

---

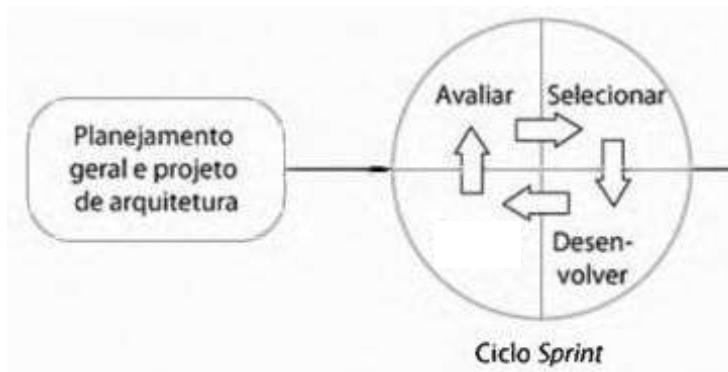


Reuniões diárias rápidas, envolvendo todos os membros da equipe.  
**Objetivo:** analisar os progressos e, se necessário, repriorizar o trabalho

# Scrum (SCHWABER e BEEDLE, 2001)

**Equipe está isolada do cliente e da organização.**

Todas as comunicações com o meio externo à equipe são canalizadas por meio do “**Scrum Master**”



**Scrum Master:** responsável por

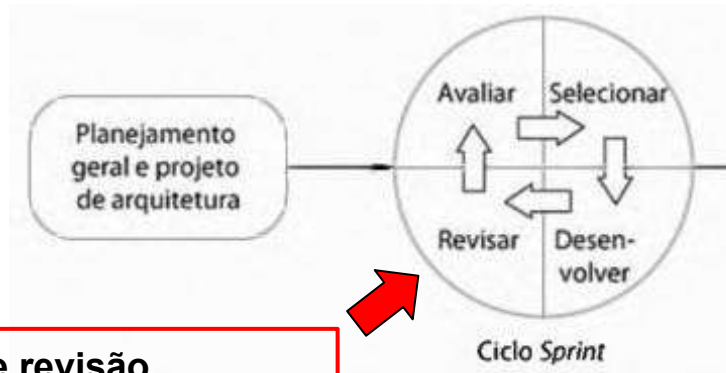
- **Organizar reuniões diárias**
- **Controlar o *backlog* de trabalho**
- **Registrar decisões**
- **Medir progresso** comparado ao backlog
- **Comunicação** com os clientes e a gerência externa à equipe

**Todos participam desse planejamento de curto prazo nas reuniões diárias.**

**Não existe uma hierarquia top-down a partir do Scrum Master.**

# Scrum (SCHWABER e BEEDLE, 2001)

---

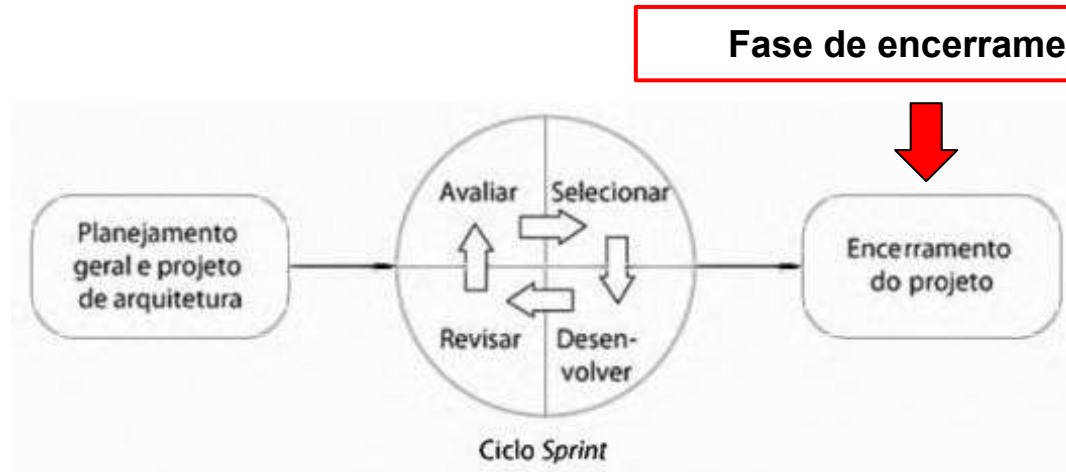


**Fase de revisão**

O trabalho é revisto e apresentado aos **stakeholders** (usuários / pessoas interessadas no software).  
O próximo ciclo *sprint* começa em seguida.

# Scrum (SCHWABER e BEEDLE, 2001)

---



Encerra o projeto, completa a documentação exigida (e.g., ajuda do sistema e manuais do usuário), e avalia as lições aprendidas com o projeto.

# Vantagens do Scrum

---

- O produto é decomposto em um conjunto de partes gerenciáveis e compreensíveis (incrementos e *sprints*)
- Requisitos instáveis não atrasam o progresso
  - Requisitos são ajustados conforme o planejamento dos *sprints*
- Melhor comunicação entre membros da equipe
  - Toda a equipe tem visão de tudo
  - Coordenação das ações pelo **scrum master**



# Vantagens do Scrum

---

- Entrega de incrementos dentro do prazo
  - Os clientes recebem *feedback* sobre como cada incremento do produto funciona
- Estabelecimento de confiança entre clientes e desenvolvedores
  - Cria-se uma cultura positiva, na qual todo mundo espera que o projeto tenha êxito

# Exercício

Considerando o SCRUM, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - O Scrum Master é responsável por intermediar todas as comunicações com o meio externo à equipe.

II - O Scrum Master é semelhante a um gerente de projetos, estando hierarquicamente superior as demais membros da equipe.

III - O Scrum Master avalia o progresso do projeto comparando-o com o backlog.

IV - O scrum Master é responsável por organizar reuniões diárias , registrar decisões e medir o progresso.

- ☐ Todas as assertivas são verdadeiras.
- ☐ Somente I e II.
- ☐ Somente I, III e IV.
- ☐ Somente I, II e IV.
- ☐ Nenhuma das alternativas anteriores.

# Exercício

Considerando o SCRUM, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - O Scrum Master é responsável por intermediar todas as comunicações com o meio externo à equipe. **V**

II - O Scrum Master é semelhante a um gerente de projetos, estando hierarquicamente superior as demais membros da equipe.

III - O Scrum Master avalia o progresso do projeto comparando-o com o backlog.

IV - O scrum Master é responsável por organizar reuniões diárias , registrar decisões e medir o progresso.

- ☐ Todas as assertivas são verdadeiras.
- ☐ Somente I e II.
- ☐ Somente I, III e IV.
- ☐ Somente I, II e IV.
- ☐ Nenhuma das alternativas anteriores.

# Exercício

Considerando o SCRUM, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - O Scrum Master é responsável por intermediar todas as comunicações com o meio externo à equipe. **V**

II - O Scrum Master é semelhante a um gerente de projetos, estando **hierarquicamente superior** as demais membros da equipe. **F**

III - O Scrum Master avalia o progresso do projeto comparando-o com o backlog.

IV - O scrum Master é responsável por organizar reuniões diárias , registrar decisões e medir o progresso.

- ☐ Todas as assertivas são verdadeiras.
- ☐ Somente I e II.
- ☐ Somente I, III e IV.
- ☐ Somente I, II e IV.
- ☐ Nenhuma das alternativas anteriores.

# Exercício

Considerando o SCRUM, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - O Scrum Master é responsável por intermediar todas as comunicações com o meio externo à equipe. **V**

II - O Scrum Master é semelhante a um gerente de projetos, estando **hierarquicamente superior** as demais membros da equipe. **F**

III - O Scrum Master avalia o progresso do projeto comparando-o com o backlog. **V**

IV - O scrum Master é responsável por organizar reuniões diárias , registrar decisões e medir o progresso.

- ☐ Todas as assertivas são verdadeiras.
- ☐ Somente I e II.
- ☐ Somente I, III e IV.
- ☐ Somente I, II e IV.
- ☐ Nenhuma das alternativas anteriores.

# Exercício

Considerando o SCRUM, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - O Scrum Master é responsável por intermediar todas as comunicações com o meio externo à equipe. **V**

II - O Scrum Master é semelhante a um gerente de projetos, estando hierarquicamente superior as demais membros da equipe. **F**

III - O Scrum Master avalia o progresso do projeto comparando-o com o backlog. **V**

IV - O scrum Master é responsável por organizar reuniões diárias , registrar decisões e medir o progresso. **V**

- ☐ Todas as assertivas são verdadeiras.
- ☐ Somente I e II.
- ☒ Somente I, III e IV.
- ☐ Somente I, II e IV.
- ☐ Nenhuma das alternativas anteriores.

# Escalamento de métodos ágeis

---

- Os métodos ágeis foram desenvolvidos para:
  - Serem usados por equipes de programação de pequeno porte
  - Equipes que podiam trabalhar juntas na mesma sala
  - Equipes com comunicação informal
- Logo, esses métodos foram usados em projetos de pequeno e médio porte
  - **Como adaptá-los para projetos maiores?**
  - **O que torna sistemas de grande porte diferentes?**



## Como sistemas de grandes porte são diferentes dos demais?

---

- Sistemas de grande porte geralmente são coleções de sistemas separados que se comunicam (***brownfield systems***)
  - Equipes separadas desenvolvem cada um dos sistemas
  - Equipes trabalham em lugares e fusos horários diferentes
  - **Consequencia:** É praticamente impossível que cada membro de cada equipe tenha a visão de todo o sistema
    - A prioridade da equipe é completar sua parte do sistema, sem levar em conta questões mais amplas do sistema como um todo



## Como sistemas de grandes porte são diferentes dos demais?

---

- Incompatibilidade entre integração de subsistemas e desenvolvimento incremental
  - Sempre que vários sistemas estão integrados para criar um único, a configuração do sistema se torna o principal foco da equipe de desenvolvimento
  - O desenvolvimento do código original é um alvo secundário

## Como sistemas de grandes porte são diferentes dos demais?

---

- Sistemas de grande porte e seus processos de desenvolvimento são frequentemente restringidos por regras externas e regulamentos
  - Limitações no desenvolvimento
    - Exigem certos tipos de documentação a ser produzida
    - Prazos definidos para a produção das documentações
    - Etc

## Como sistemas de grandes porte são diferentes dos demais?

---

- Dificuldade em manter equipes coerentes que conheçam sobre o sistema
  - Pessoas, inevitavelmente, deslocam-se para outros trabalhos e projetos
  - Sistemas de grande porte têm um longo tempo de aquisição e desenvolvimento

# Como sistemas de grandes porte são diferentes dos demais?

---

- Dificuldade em envolver os diferentes *stakeholders* no desenvolvimento do sistema
  - Conjunto diverso de *stakeholders*
    - **Ex:** enfermeiros, administradores, pessoal médico sênior, gerentes de hospital, etc



# Métodos ágeis em sistemas de grande porte

---

- Segundo Sommerville (2003), é necessário realizar as seguintes adaptações aos métodos ágeis para utilização em sistemas de grande porte:
  - Foco não somente no código, mas também na arquitetura e documentação do sistema
  - Uso de mecanismos de comunicação entre equipes (e.g., email, video conferencias, telefonemas, etc)
  - Manter construções freqüentes e releases regulares, sem integração contínua

# Atividade em sala

---

- Em grupo, discutam se (e qual) metodologia ágil é adequada ao projeto do grupo
  - Como metodologias ágeis podem ser empregadas no projeto?
  - Ajustem seu planejamento de acordo com a metodologia
  - Investiguem técnicas de modelagem de sistemas usando diagramas UML
  - Discutam quais diagramas vocês irão adotar no projeto

# Referencial Bibliográfico

---

- SOMMERVILLE, Ian. **Engenharia de Software**. 6. ed. São Paulo: Addison-Wesley, 2003.
- PRESSMAN, Roger S. **Engenharia de Software**. São Paulo: Makron Books, 1995.
- JUNIOR, H. E. **Engenharia de Software na Prática**. Novatec, 2010.