Arquitetura de software

Professor: Eduardo Cizeski Meneghel

REVISÃO AULA ANTERIOR

PADRÕES E ESTILOS DE ARQUITETURAS

PADRÕES E ESTILOS

- Padrões e estilos de arquitetura de software são abordagens e diretrizes para projetar a estrutura de um sistema de software.

ESTILOS ABORDADOS ANTERIORMENTE

- Monolítica vs distribuída;
- Cliente-servidor;
- Arquitetura em camadas;
- Arquitetura orientada a serviços (SOA);
- Arquitetura baseada em microsserviços.

ESTILOS ABORDADOS ANTERIORMENTE

- Arquitetura em pipes e filtros;
- Arquitetura orientada a eventos
- Arquitetura em estilo publicador-subscritor;
- Arquitetura estilo MVC.

ESTILO DE ARQUITETURA MVVM

PADRÕES E ESTILOS

- O objetivo do MVVM é prover uma separação de responsabilidades entre a view e sua lógica, contudo, isto agrega benefícios como aumento da testabilidade da aplicação.

PADRÕES E ESTILOS

Model

- O Model tem função similar ao do MVC, contendo implementação do modelo de domínio da aplicação que inclui o modelo de dados, regras de negócio e validações de lógica.

PADRÕES E ESTILOS

View

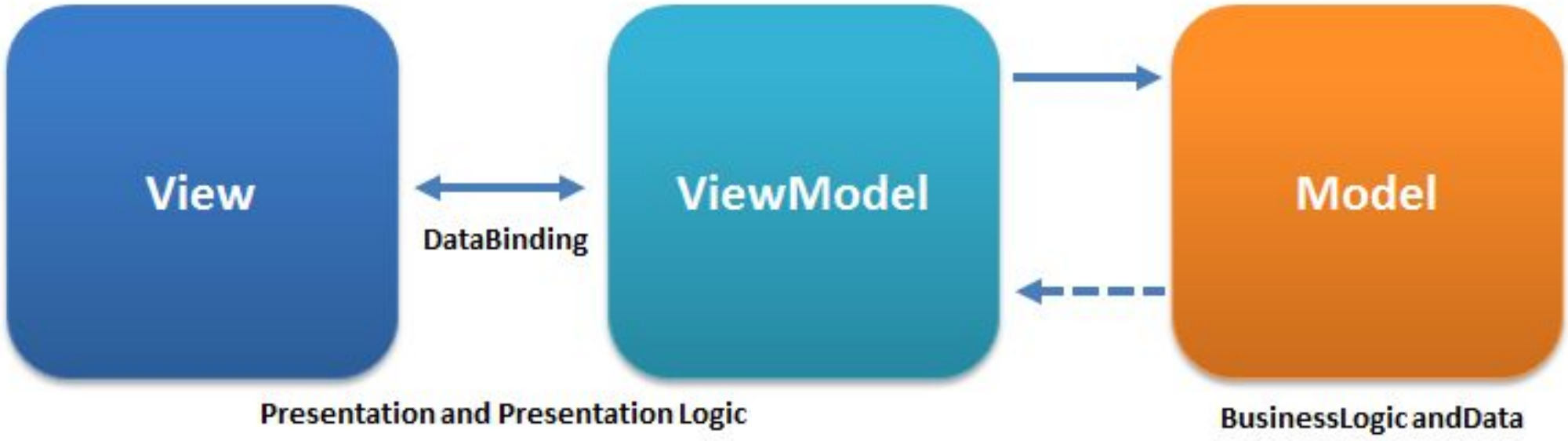
- Entidade responsável por definir a estrutura, layout e aparência do que será exibido na tela.

PADRÕES E ESTILOS

ViewModel

- Ela disponibiliza para a View uma lógica de apresentação e coordena as iterações da View com a Model, além de poder implementar a lógica de validação para garantir a consistência dos dados.

PADRÕES E ESTILOS



PADRÕES E ESTILOS

Vantagens

- Permite simplificar testes de unidades;
- O desenvolvedor pode optar por reaproveitar um ViewModel em várias Views;
- O código para atualizar a View é pequeno, uma vez que as alterações podem ser propagadas automaticamente.

PADRÕES E ESTILOS

Desvantagens

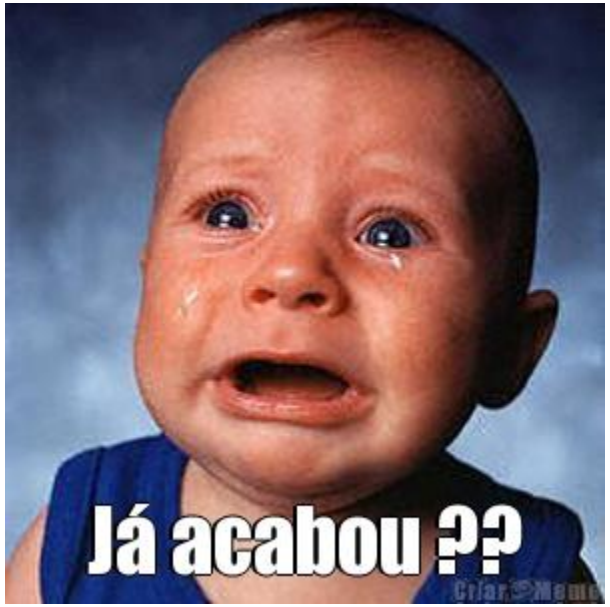
- É uma estrutura com uma complexidade maior e para aplicações pequenas poderá adicionar camadas e regras desnecessárias.
- Aplicações que possuem vários pontos de alteração de um mesmo Model, podem acabar exibindo dados desatualizados.

PADRÕES E ESTILOS

- Tal modelo foi criado pelo arquiteto de software do WPF e Silverlight da Microsoft, **John Gossman** em 2005. Desde então, ele vem sendo usado principalmente no desenvolvimento mobile.

- O MVVM pode ser usado em Swift, Java, Dart (através do framework Flutter) e ainda os frameworks de JavaScript.

ESTILO DE ARQUITETURA MDA



PADRÕES E ESTILOS

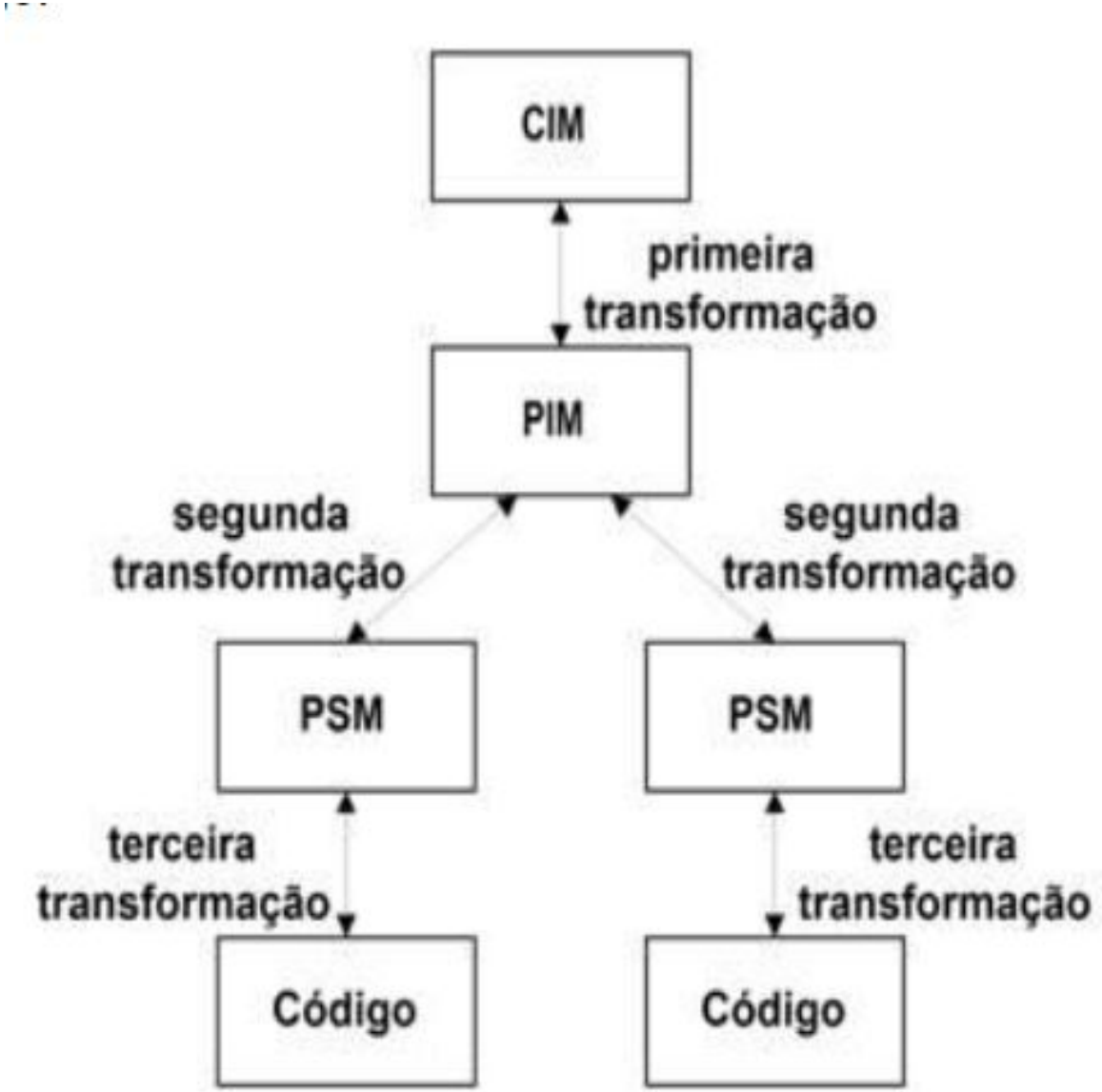
- MDA - Model Driven Architecture ou arquitetura orientada por modelos;
- Criado em 2001 pela OMG (Object Management Group).

PADRÕES E ESTILOS

- A MDA utiliza para construir seu ciclo de vida, os chamados modelos, que são todos os artefatos como especificação de requisitos, descrição da arquitetura, descrição do modelo e código.

- A MDA define frameworks que separam as especificações da funcionalidade do sistema de sua implementação em uma plataforma específica.

PADRÕES E ESTILOS



PADRÕES E ESTILOS

Modelo CIM (Modelo Independente de Computação)

- É onde se inicia a primeira fase, ou seja, o entendimento adequado do negócio, especializando-se nos processos e determinando o que necessariamente o sistema que irá automatizar o determinado processo deve fazer.

PADRÕES E ESTILOS

Modelo PIM (Modelo independente de plataforma)

- O modelo PIM foca no ponto de visão de operações independente de plataforma.

- O PIM define um conjunto de componentes e funcionalidades, que são definidas independentemente de qualquer plataforma, ou seja, sem a necessidade de adicionar no modelo tecnologias, que serão utilizadas no projeto. Ele por de ser visto como uma abstração do sistema que pode ser realizado por diferentes plataformas específicas.

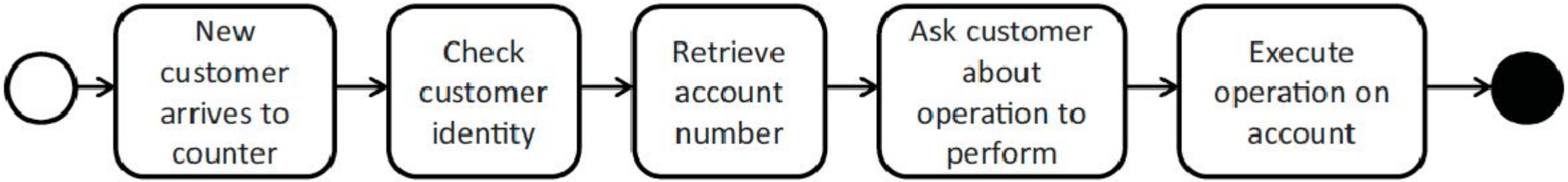
PADRÕES E ESTILOS

Modelo PSM (Modelos Específicos de Plataforma)

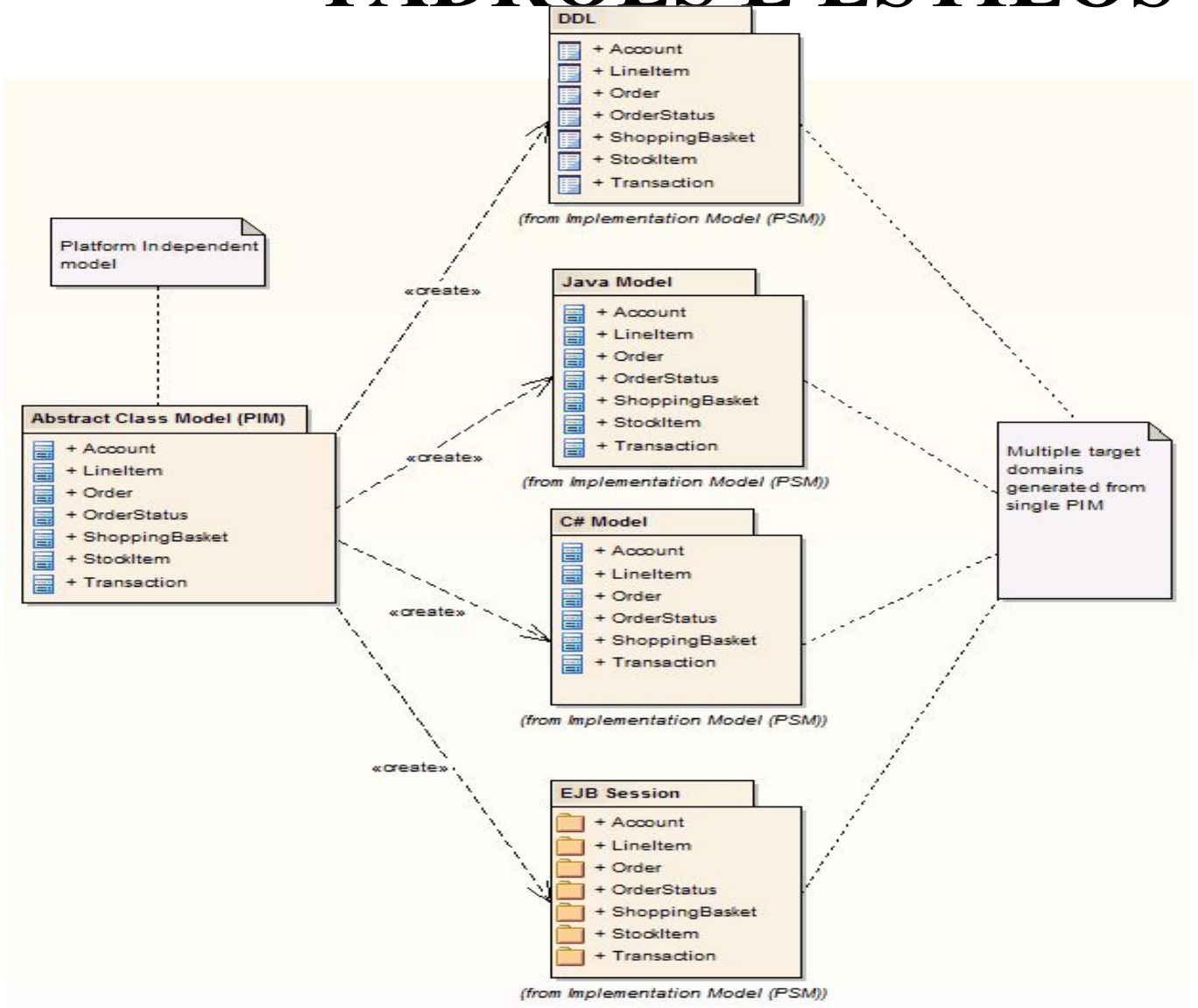
- O PSM é mais específico para o sistema em termos de tecnologia de implementação, como um modelo de banco de dados ou um modelo EJB (Enterprise JavaBeans).

PADRÕES E ESTILOS

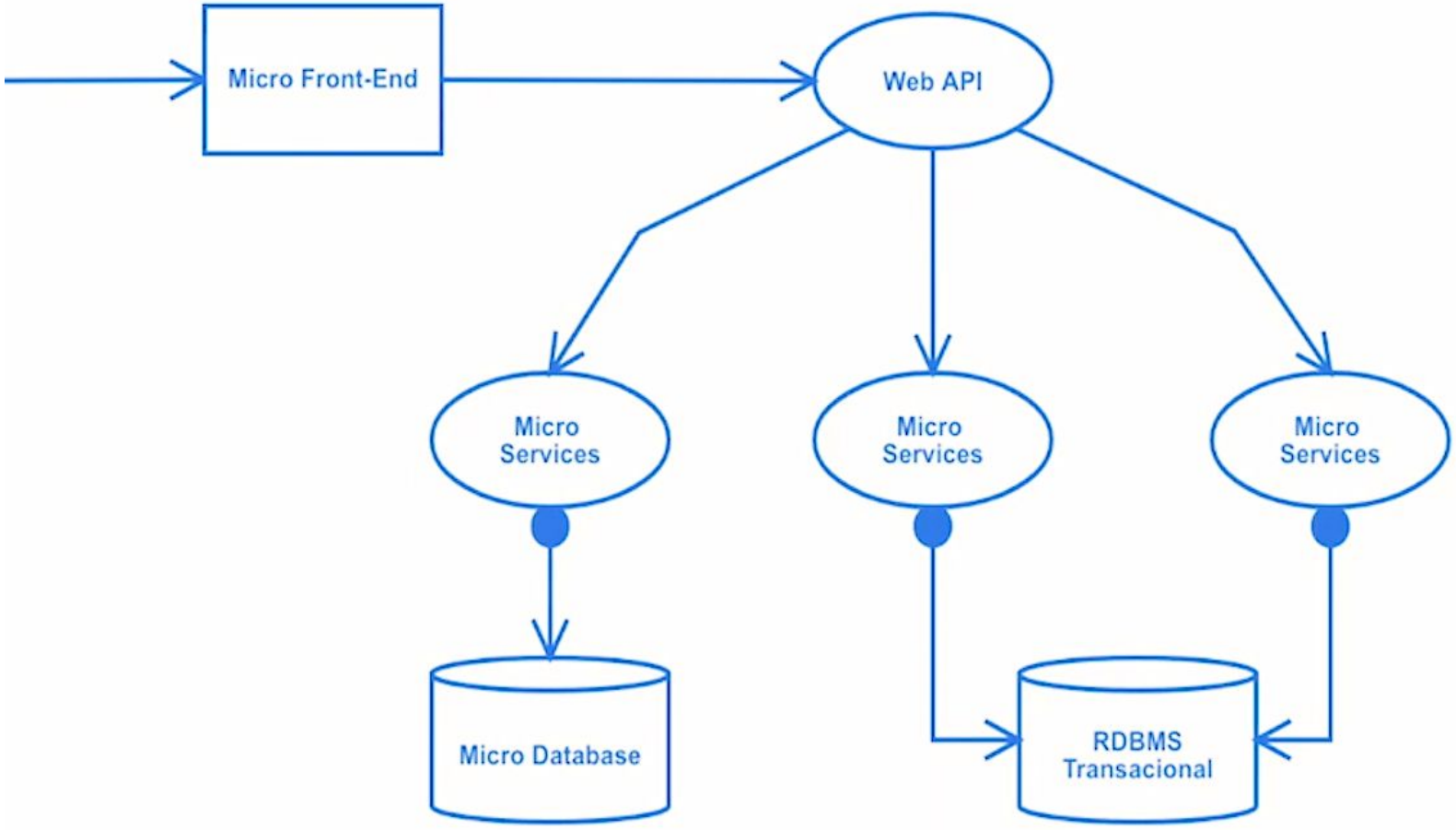
Modelo CIM (Modelo Independente de Computação)



PADRÕES E ESTILOS



PADRÕES E ESTILOS



PADRÕES E ESTILOS

- Produtividade: A **transformação** do PIM para o PSM precisa ser definida uma única vez e pode ser aplicada no desenvolvimento de diversos sistemas. Devido a este fato, tem-se uma redução no tempo de desenvolvimento.

PADRÕES E ESTILOS

- **Portabilidade:** Dentro da MDA a portabilidade é alcançada através do foco dado no desenvolvimento do PIM, que é independente de plataforma. Um mesmo PIM pode ser automaticamente transformado em vários PSMs de diferentes plataformas.

PADRÕES E ESTILOS

- Planejamento e detalhamento da aplicação;

PADRÕES E ESTILOS

Opinião de Martin Fowler sobre MDA:

“Como exemplo, considere a lógica comportamental. Não consigo ver que desenhar diagramas de sequência ou diagramas de atividades seja tão bom, e muito menos melhor, do que escrever código em uma linguagem moderna.”

PADRÕES E ESTILOS

Opinião de Martin Fowler sobre MDA:

“Mesmo que a UML forme um ambiente de programação eficaz, ela ainda precisa se tornar popular. Como ex-Smalltalker, sei muito bem que mesmo as melhores línguas nem sempre se tornam populares.”

PRINCÍPIOS DE INTEGRAÇÃO ENTRE SISTEMAS

PADRÕES E ESTILOS

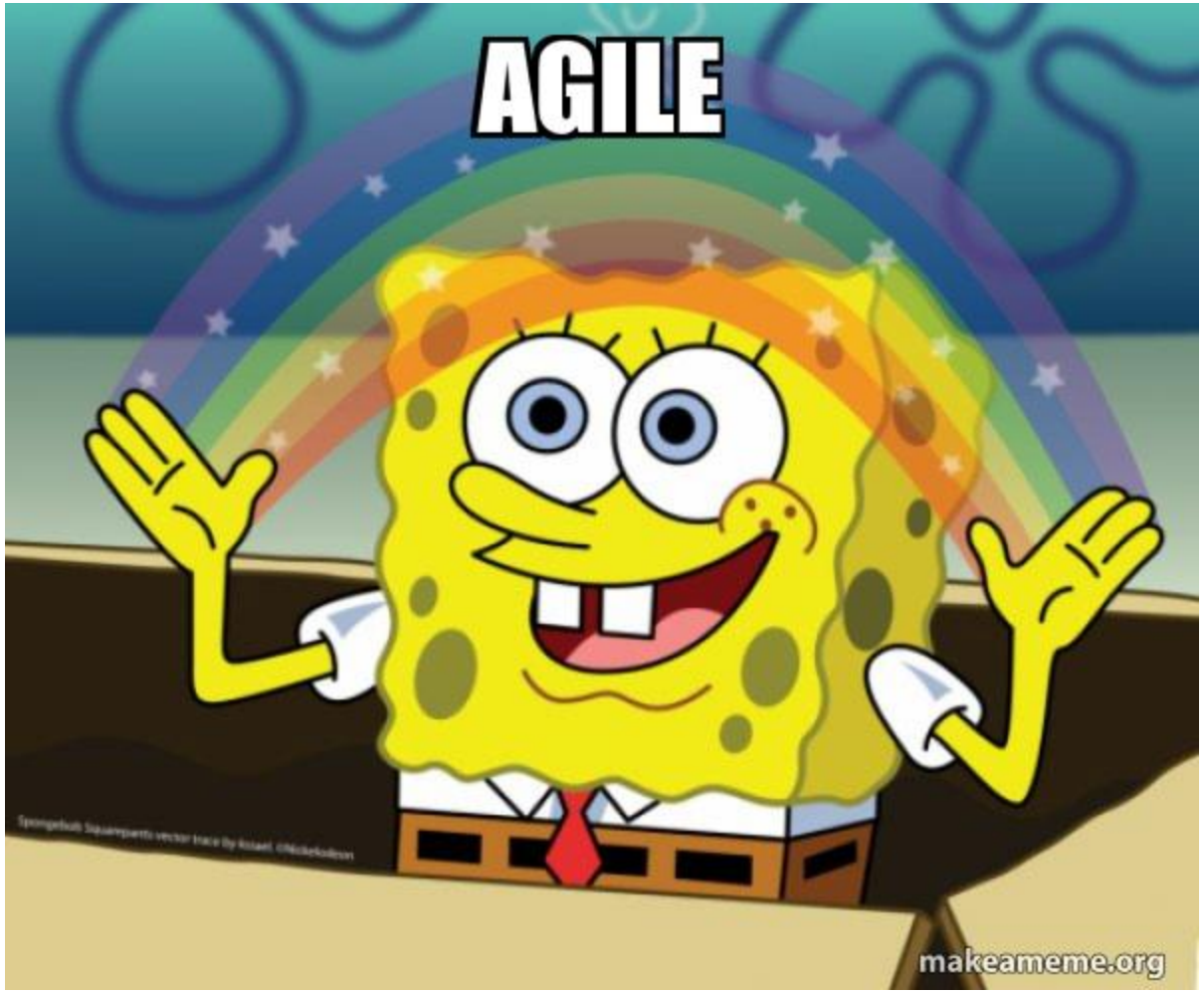
- Trabalho 1 - Artigo;

- Aula remota (08/09/2023).

PADRÕES E ESTILOS

- Listas de exercícios - MVVM e MDA

ARQUITETURA DE SOFTWARE



AGILIDADE

Métodos ágeis são métodos de desenvolvimento incremental de software, concentrado na velocidade do desenvolvimento, entregas frequentes do software, redução de overheads dos processos e produção de códigos de alta qualidade, envolvendo o cliente diretamente no processo de desenvolvimento.

AGILIDADE

- Nova forma de gestão e desenvolvimento de software;
- Mentalidade.

AGILIDADE

Nas décadas de 80 e 90, os engenheiros de software tinham uma visão de que a melhor maneira de se conseguir um bom planejamento era por meio de processos cuidadosos e minuciosos, construindo softwares grandes e com prazos gigantescos.

AGILIDADE

Quando aplicado em processos de pequeno porte, esses planejamentos eram tão dispendiosos que dominavam o processo de desenvolvimento de software, ou seja, o tempo gasto em análises e planejamento era muito superior ao desenvolvimento do sistema.

AGILIDADE

O Manifesto Ágil foi publicado nos dias 11 a 13 de fevereiro de 2001 como um trabalho de 17 desenvolvedores de software interessados em buscar uma alternativa aos atuais processos de desenvolvimento de software.

AGILIDADE

Signatários:

- Robert C. Martin, o “Uncle Bob”;
- Ken Schwaber, co-criador do Scrum.
- Jeff Sutherland, o inventor do Scrum.
- Kent Back, co-criador da eXtreme Programming (XP).
- Ron Jeffries, co-criador da eXtreme Programming (XP).
- Mike Beedle, co-autor de Desenvolvimento Ágil de Software com Scrum.
- Arie van Bennekum, da Integrated Agile.
- Alistair Cockburn, criador da Metodologia Ágil Crystal.
- Ward Cunningham, criador do conceito wiki.
- Martin Fowler, desenvolvedor parceiro da Thoughtworks.
- James Grenning, autor de Test Driven Development.
- Jim Highsmith, criador do Adaptive Software Development (ASD).
- Andrew Hunt, co-autor de O Programador Pragmático.
- Jon Kern, atuante até os dias de hoje em assuntos de agilidade.
- Brian Marick, cientista da computação e autor de vários livros sobre programação.
- Steve Mellor, cientista da computação e um dos idealizadores da Análise de Sistema Orientado a Objetos (OOSA).
- Dave Thomas, programador e co-autor de The Pragmatic Programmer.

AGILIDADE

https://agilemanifesto.org/iso/ptbr/manifesto.html

AGILIDADE

O manifesto conta com **QUATRO** valores e **DOZE** princípios.

AGILIDADE

Valor 1:

Indivíduos e interações mais que processos e ferramentas.

AGILIDADE

Valor 2:

Software em funcionamento mais que documentação abrangente.

AGILIDADE

Valor 3:

Colaboração com o cliente mais que negociação de contratos.

AGILIDADE

Valor 4:

Responder a mudanças mais que seguir um plano.

AGILIDADE

Principais princípios:

Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado.

AGILIDADE

Principais princípios:

Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.

AGILIDADE

Principais princípios:

Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo.

AGILIDADE

Principais princípios:

Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.

AGILIDADE

Principais princípios:

Contínua atenção à excelência técnica e bom design aumenta a agilidade.

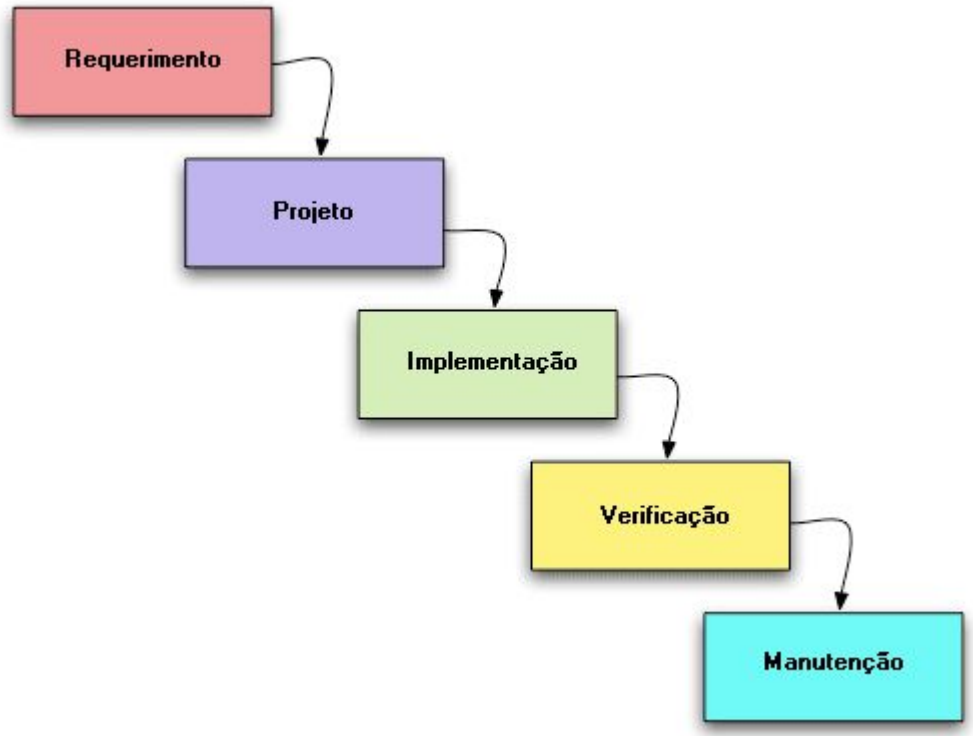
AGILIDADE

Principais princípios:

Simplicidade – a arte de maximizar a quantidade de trabalho não realizado – é essencial.

AGILIDADE

Desenvolvimento Watterfall (tradicional).



AGILIDADE

Nas metodologias clássicas de gerenciamento de projeto, a flexibilidade de alteração de escopo é muito estressante, pois não oferece – ou oferece poucas – oportunidades de alterações ou atualizações de escopo.

AGILIDADE

No gerenciamento tradicional, a responsabilidade pertence totalmente ao gerente de projeto, cabendo a ele planejar e documentar todo o ciclo de vida deste.

Ele trata diretamente com o cliente, e os prazos em necessidades de desenvolvimento são negociados entre os dois. Os membros da equipe só terão ciência após tudo planejado

AGILIDADE

Nas metodologias clássicas, o processo é “engessado” ao escopo inicial, ao prazo e orçamento estimado. Portanto, qualquer alteração ou feedback necessário, se não ignorado, é tratado com burocracias, como alterações de escopo etc.

AGILIDADE

https://www.youtube.com/watch?v=f-ByA2TLpnA

AGILIDADE

- Lista de exercícios - Agilidade


@Cinemex
HASTA
LA VISTA, BABY
TERMINATOR 2