Introdução aos Sistemas de informação

Methods

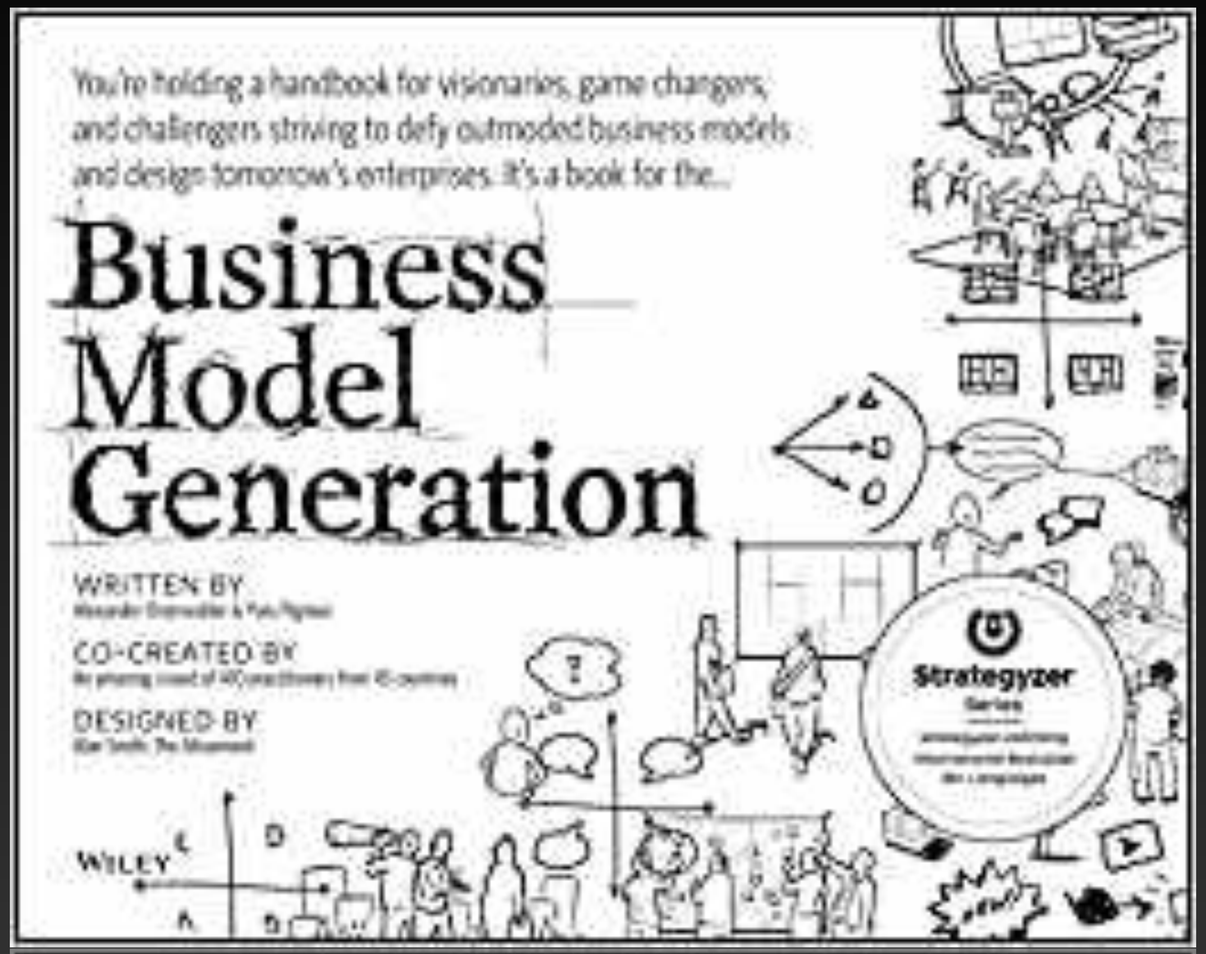
Metodologias e diagramas para desenvolvimento de produtos

Antônio Carlos de Renzede Filho
Brunna Ferreira Gonçalves
Gabriel de Sousa Borges
José Maxwell Ismael de Oliveira
Manuela Oliveira Rocha
Leonardo Balestere Alves

Índice

- Canvas
- eXtreme programming
- Integração contínua (CI)
- SOA
- Scrum
- UML
- FDD

Canvas



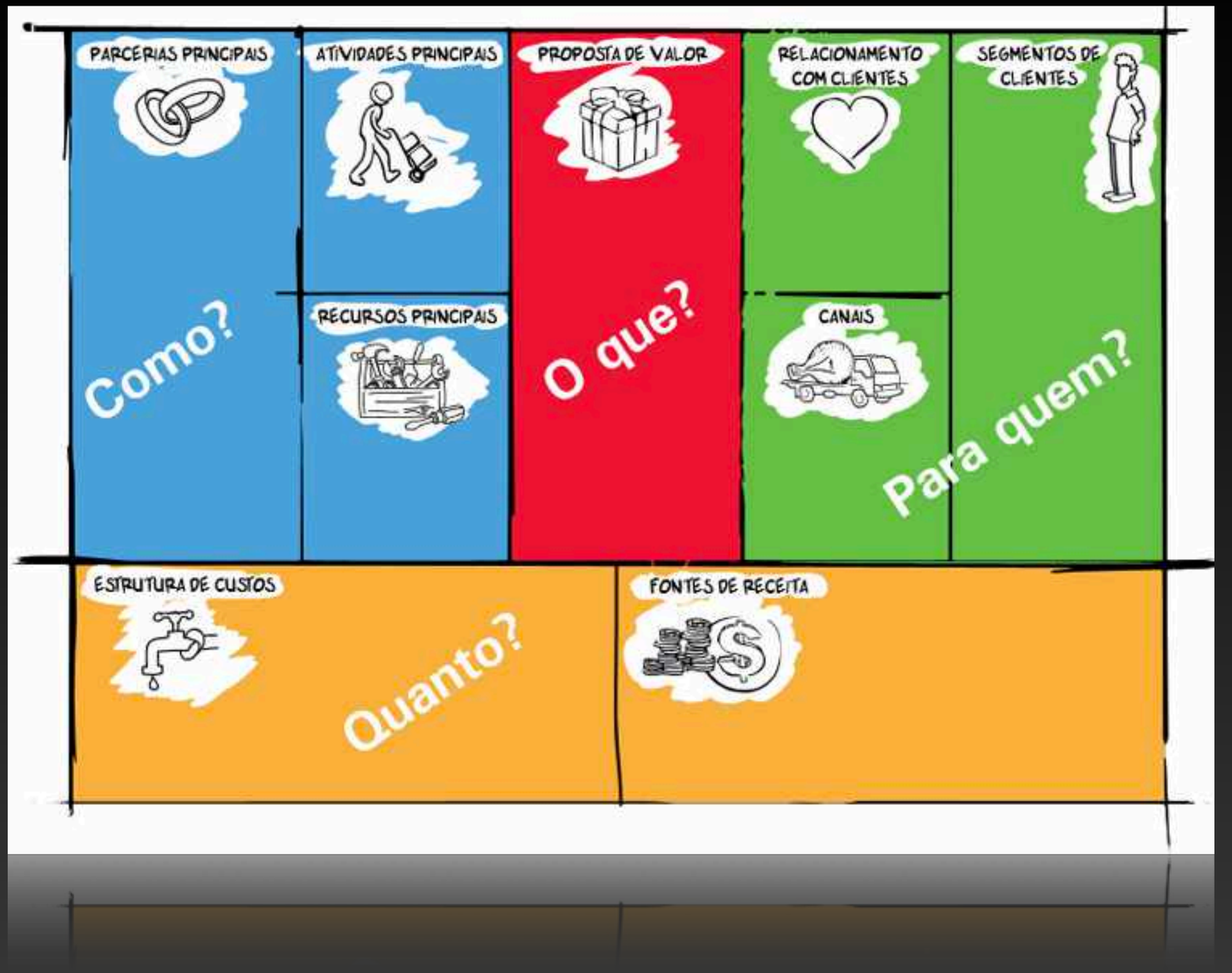
Alexander Osterwalder e Yves Pigneur

https://www.amazon.com.br/Business-Model-Generation-Visionaries-Challengers/dp/0470876417

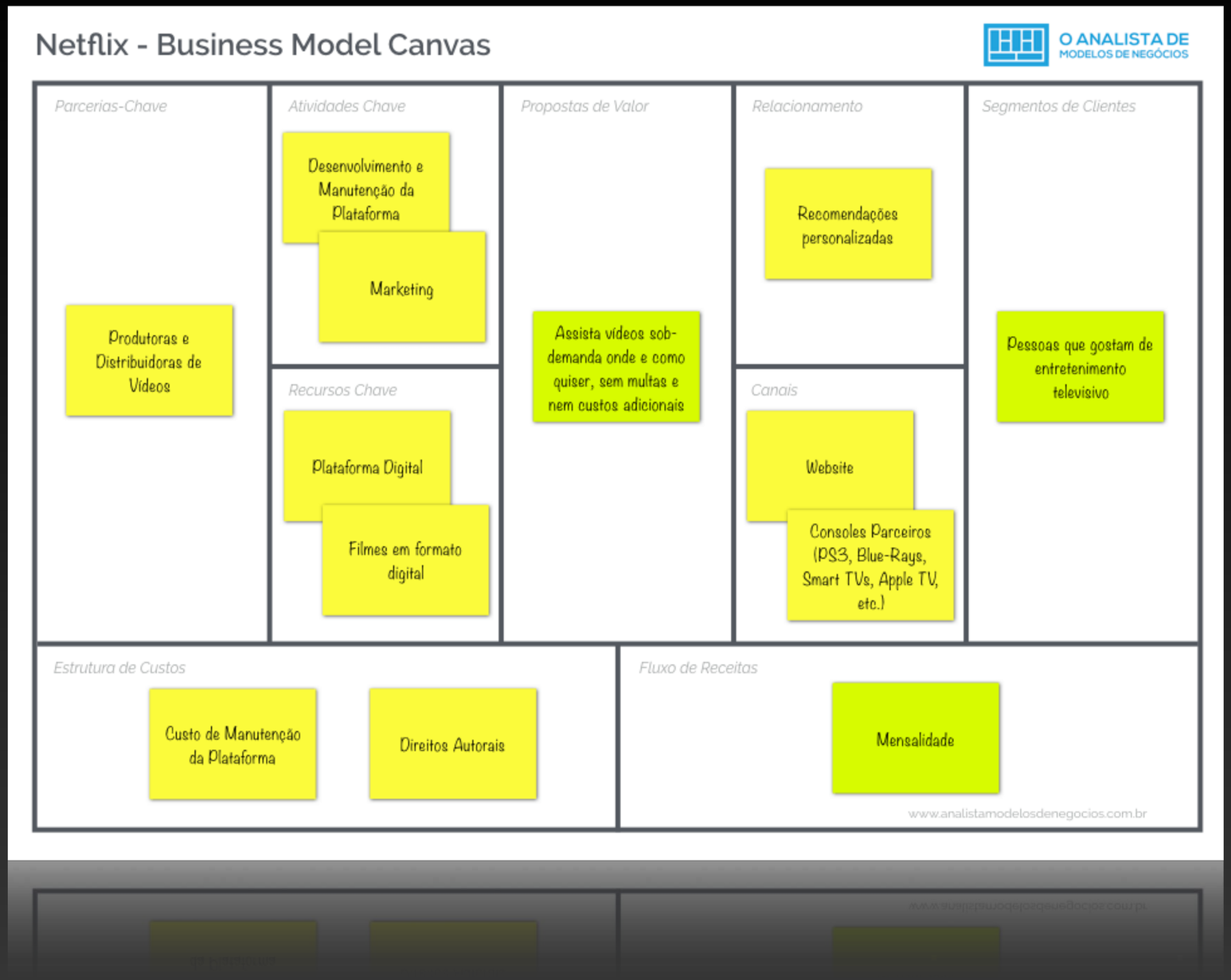
Modelo de negócio x Plano de negócio

Modelo de negócio x Plano de negócio

- Um Modelo de Negócio é a representação de como a sua empresa gera e entrega valor para os seus clientes.
- Um Plano de Negócios é um documento detalhado em que o empreendedor deve descrever todas as etapas que envolvem ou fazem parte do seu negócio.



http://www.sourceinnovation.com.br/wiki/Projeto_em_Canvas



https://analistamodelosdenegocios.com.br/modelo-de-negocio-da-netflix/

eXtreme programming

O que é?

- Criada por Kent Baeck em 1996 durante o projeto Daimler Chrysler.
- Chrysler Comprehensive Compensation System – Sistema de Compensação Abrangente da Chrysler).
- O sucesso de XP advém da intensa satisfação do cliente.
- Cliente satisfeito é o melhor indicativo de sucesso de um projeto.
- Esta metodologia foi criada para produzir o software que o cliente precisa seguindo as especificações à risca.
- XP encoraja os desenvolvedores a atender as requisições de mudanças dos requisitos do software, no momento em que isto acontece.

Valores

- Simplicidade: Conceito de simplificar e tentar não complicar, otimizando a codificação.
- Comunicação: A comunicação precisa ser muito clara, deve levar a informação correta para o público alvo.
- Feedback: Retornos constantes em relação ao time (profissionais).
- Coragem: Ambiente encorajador.
- Respeito: Ambiente respeitoso entre os membros do time.

Metodologia

Metodologia

- Jogo de Planejamento
- Fases Pequenas
- Equipe (Tecnico e clientes)
- Testes de Aceitação
- Semana de 40 horas
- Propriedade Coletiva
- Integração continua
- Ritmo Sustentavel

Metodologia

- Desenvolvimento em Pares
- Teste Unitario
- Refatoração
- Design Simples

Integração contínua (CI)

O que é?

- A integração contínua pode ser explicada como uma pratica que está inserida no desenvolvimento de softwares. Conhecido como DevOps, ou seja, os desenvolvedores fazem a junção das alterações de código em um repositório central. Depois disso, criações e testes são executados.
- É originada da metodologia ágil XP e sempre foi usada em várias outras metodologias. A grande sacada está no fato de o desenvolvedor conseguir fazer a integração do código alterado ou desenvolvido do projeto, porém, com muito mais agilidade.

Objetivos

ENCONTRAR E INVESTIGAR BUGS MAIS RAPIDAMENTE.

MELHORAR A QUALIDADE DO SOFTWARE.

REDUZIR O TEMPO QUE LEVA PARA VALIDAR E LANÇAR NOVAS ATUALIZAÇÕES DE SOFTWARE

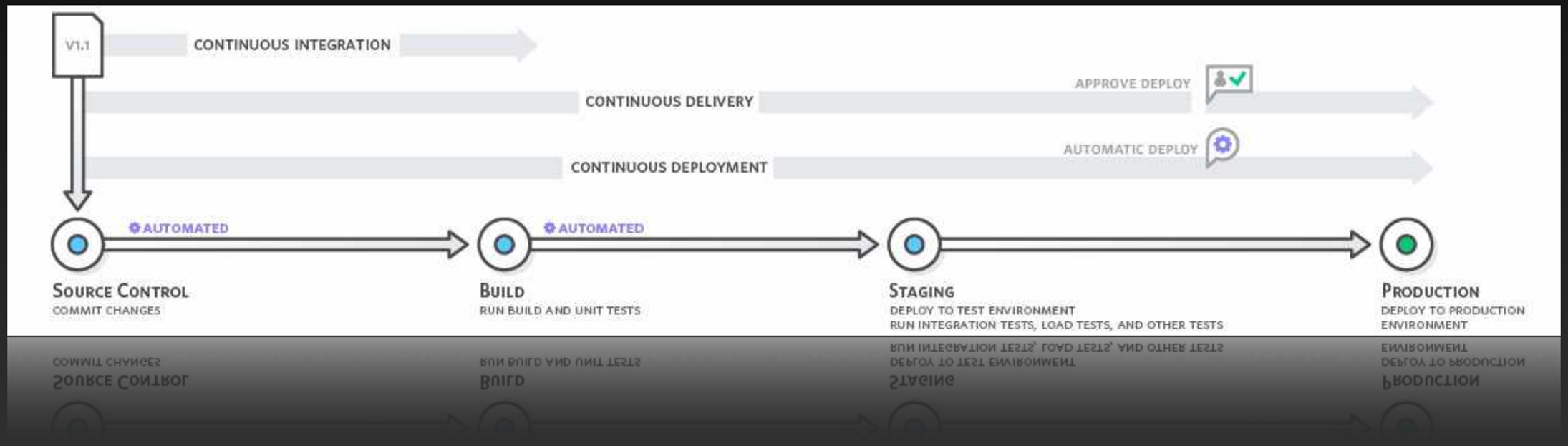
Por que a Integração contínua é necessária

- No passado, os desenvolvedores de uma equipe podiam trabalhar isoladamente por um longo período e só juntar suas alterações ao repositório central quando concluíssem seu trabalho. Dessa forma, a junção das alterações de códigos era difícil e demorada, além de resultar no acúmulo de erros sem correção por longos períodos. Estes fatores dificultavam uma distribuição de atualizações rápida e efetiva para os clientes.

Como funciona a integração contínua?

- Com a integração contínua, os desenvolvedores frequentemente usam um repositório compartilhado que utiliza de um sistema de controle de versão, como o Git. Antes de cada confirmação, os desenvolvedores podem escolher executar testes de unidade locais em seus códigos como uma camada de verificação extra anterior à integração. Um serviço de integração contínua cria e executa automaticamente testes de unidade nas novas alterações de código para destacar imediatamente todos os erros.

Fluxo do processo de Integração contínua



https://aws.amazon.com/pt/devops/continuous-integration/

Principais ferramentas da Integração contínua

- Jenkins
- Hudson
- GitlabCI
- Bitbucket Pipelines
- AWS CodePipeline
- Circle CI

Benefícios da Integração contínua

- A integração contínua ajuda sua equipe a ser mais produtiva ao liberar os desenvolvedores de tarefas manuais e encorajar comportamentos que ajudam a reduzir o número de erros e bugs implantados para os clientes.
- Com testes mais frequentes, sua equipe pode descobrir e investigar bugs mais cedo, antes que no futuro os problemas cresçam demais.
- A integração contínua ajuda a sua equipe a distribuir atualizações para os clientes mais rapidamente e com maior frequência.

SOA

Service-Oriented Architecture ou Arquitetura Orientada a Serviços

De onde veio?

- Em 1996 foi proposto pelos pesquisadores Roy Schulte e Yefim Natis do Gartner group o conceito de SOA.

SOA é uma abordagem arquitetural corporativa
que permite a criação de serviços de negócio interoperáveis
que podem facilmente ser reutilizados e compartilhados
entre aplicações e empresas.
(Gartner Group)

O que é?

É um padrão de arquitetura de software que pode ser usado por qualquer empresa, não significa que seja obrigatório porém possui vários benefícios.

Beneficios

- A diminuição do tempo de desenvolvimento
- Facilidade de Manutenção
- Menor custo
- Flexibilidade
- Melhor controle

Considerações sobre o SOA

- SOA não é uma tecnologia.
- SOA não é uma metodologia.
- SOA pode ser considerada uma filosofia arquitetural.
- SOA não é algo que se possa comprar ou instalar.
- SOA não é um web service.
- SOA não cria nada SOA

Scrum

O que é?

Scrum é uma metodologia ágil para gestão e planejamento de projetos de software.

Quando surgiu?

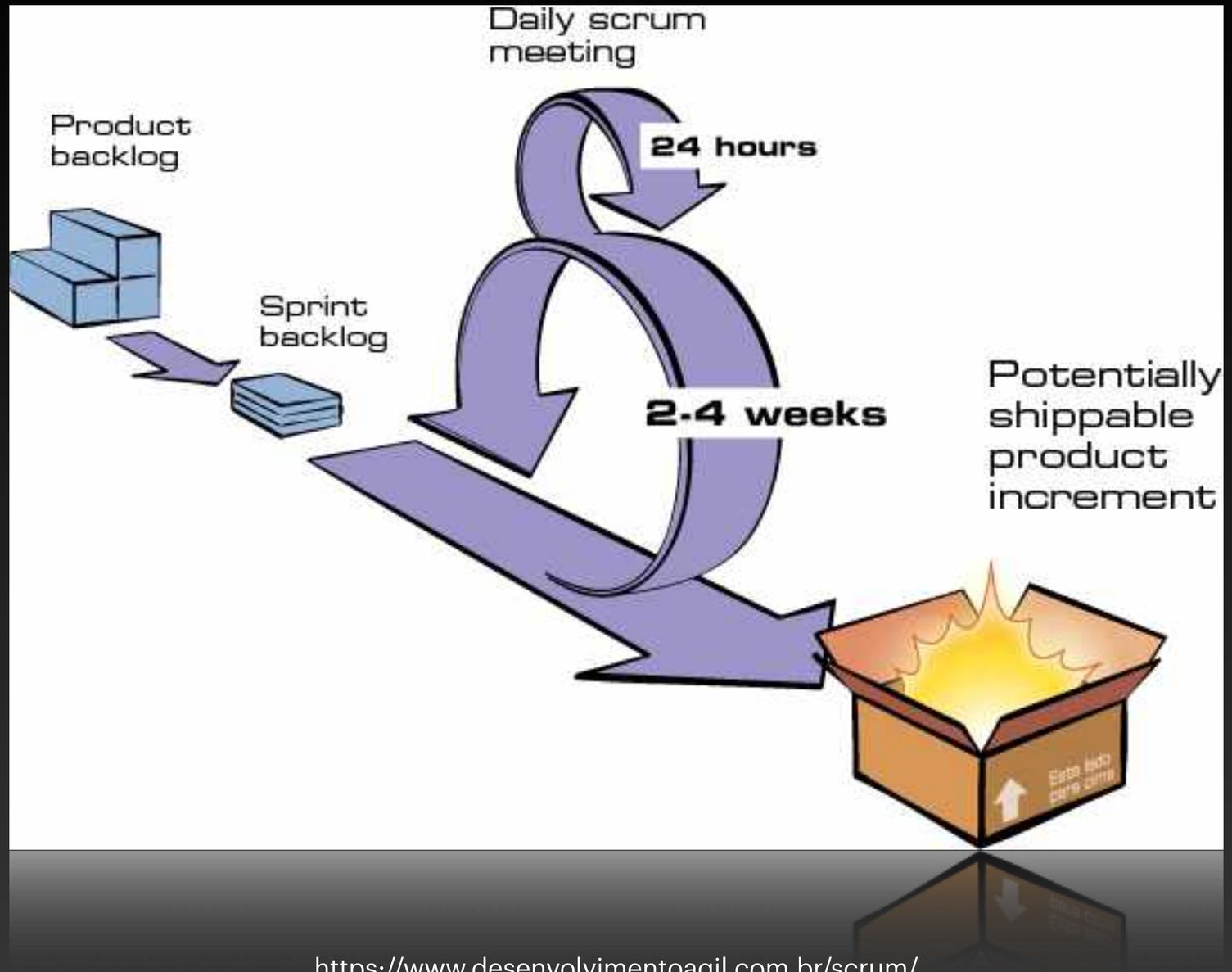
Inicialmente, o Scrum foi concebido como um estilo de gerenciamento de produtos em empresas de fabricação de automóveis e produtos de consumo, por Takeuchi e Nonaka no artigo "The New Product Development Game" (Harvard Business Review, Janeiro-Fevereiro 1986). Eles notaram que projetos usando equipes pequenas e multidisciplinares (cross-functional) produziram os melhores resultados, e associaram estas equipes altamente eficazes à formação Scrum do Rugby (utilizada para reinício do jogo em certos casos). Jeff Sutherland, John Scumniotales e Jeff McKenna conceberam, documentaram e implementaram o Scrum na empresa Easel Corporation em 1993, incorporando os estilos de gerenciamento observados por Takeuchi e Nonaka.

O que é?

Scrum é uma metodologia ágil para gestão e planejamento de projetos de software.

Como funciona?

- 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.
- As funcionalidades a serem implementadas em um projeto são mantidas em uma lista que é conhecida como Product Backlog.
- No início de cada Sprint, faz-se um Sprint Planning Meeting, ou seja, uma reunião de planejamento na qual o Product Owner prioriza os itens do Product Backlog 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.
- A cada dia de uma Sprint, a equipe faz uma breve reunião (normalmente de manhã), chamada Daily Scrum.
- O objetivo é disseminar conhecimento sobre o que foi feito no dia anterior, identificar impedimentos e priorizar o trabalho do dia que se inicia.



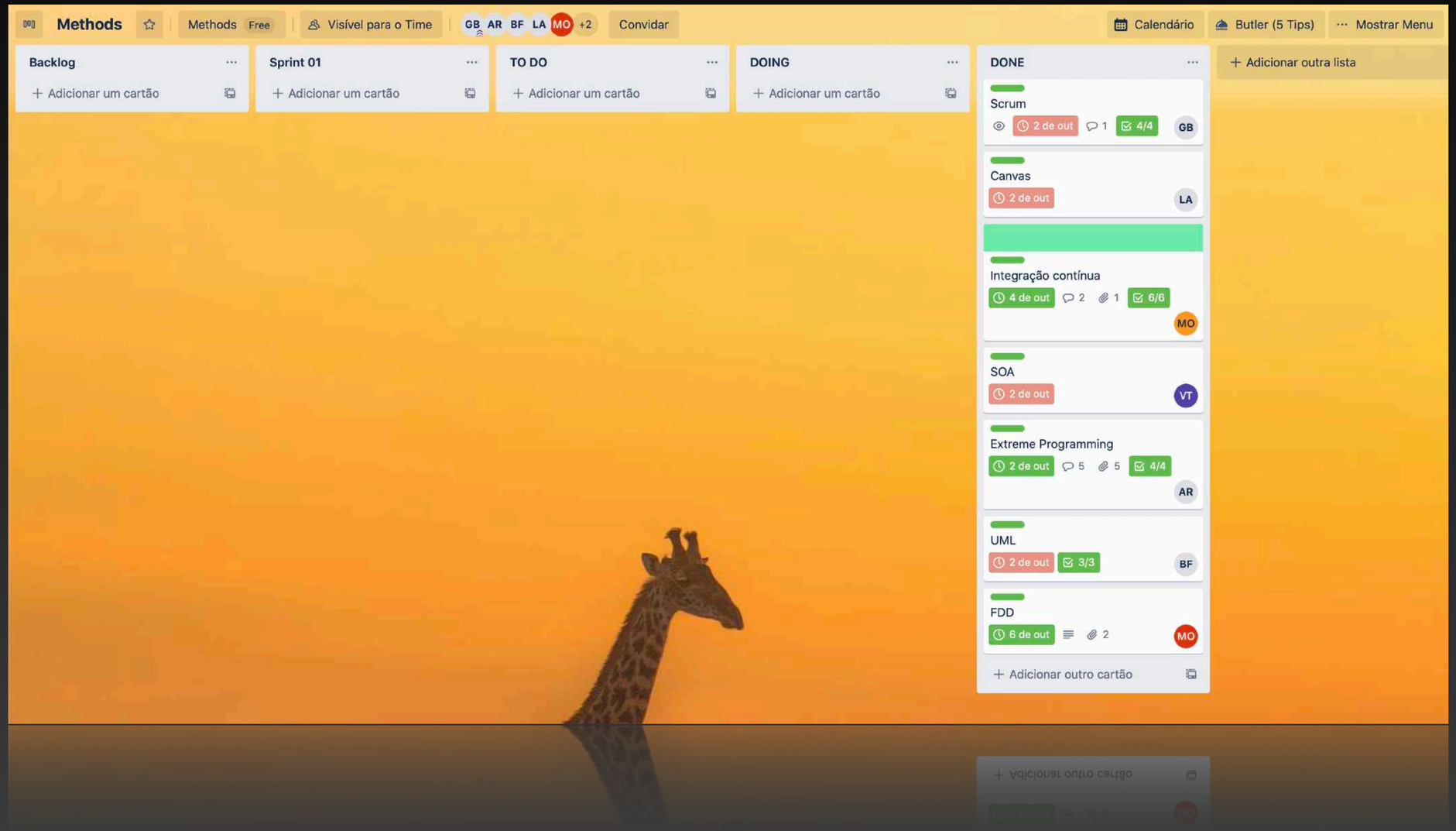
https://www.desenvolvimentoagil.com.br/scrum/

Que bom, acabou por aqui...

Nada disso, precisamos finalizar processo

- Ao final de um Sprint, a equipe apresenta as funcionalidades implementadas em uma Sprint Review Meeting.
- Finalmente, faz-se uma Sprint Retrospective e a equipe parte para o planejamento do próximo Sprint. Assim reinicia-se o ciclo.

Nosso board <3



UML

Unified Modeling Language

O que é?

- A Unified Modeling Language, ou Linguagem Unificada de Modelagem, é uma linguagem de notação utilizada para modelar e documentar as fases do desenvolvimento de sistemas orientados a objetos.
- Ela define uma série de elementos gráficos, que são usados em diferentes diagramas para representar os componentes de uma aplicação, suas interações e mudanças de estados.
- Surgiu no final dos anos 1990, da união de três metodologias de modelagem:
    1. Método de Booch - Grady Booch
    2. Método OMT - Ivar Jacobson
    3. Método OOSE - James Rumbaugh

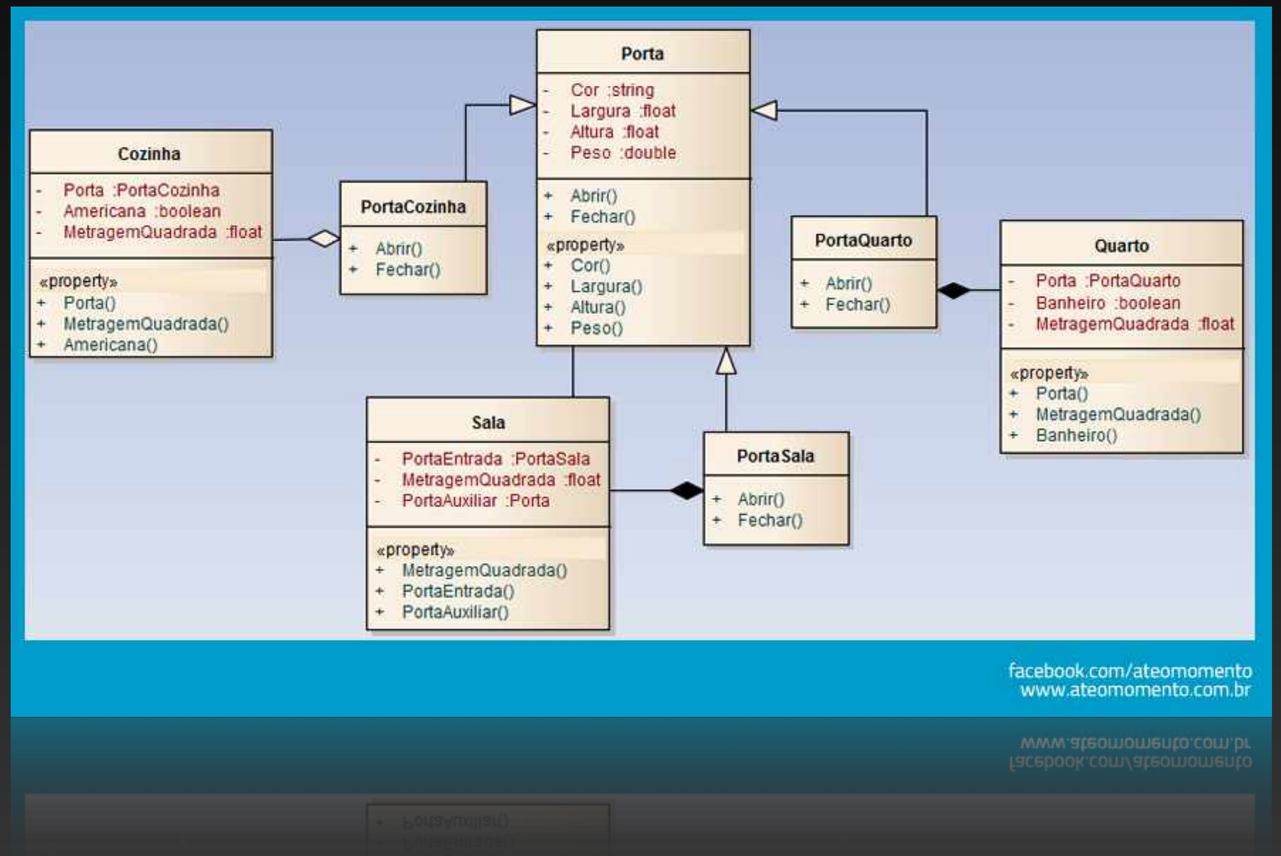
Diagramas estruturais e Diagramas comportamentais

Diagramas estruturais

- Diagrama de Objetos - Apresenta o estado de instâncias de objetos dentro de um sistema, levando em conta para isto um intervalo de tempo específico.
- Diagrama de Componentes - Está associado à linguagem de programação e tem por finalidade indicar os componentes do software e seus relacionamentos
- Diagrama de implantação - Determina as necessidades de hardware e características físicas do Sistema
- Diagrama de Pacotes - Representa os subsistemas englobados de forma a determinar partes que o compõem.
- Diagrama de Estrutura Composta - Utilizado para demonstrar a estrutura interna de uma classe, incluindo referências que apontam para outras partes de um sistema.

Diagramas estruturais

Diagrama de Classes - Mostra o conjunto de classes com seus atributos e métodos e os relacionamentos entre classes

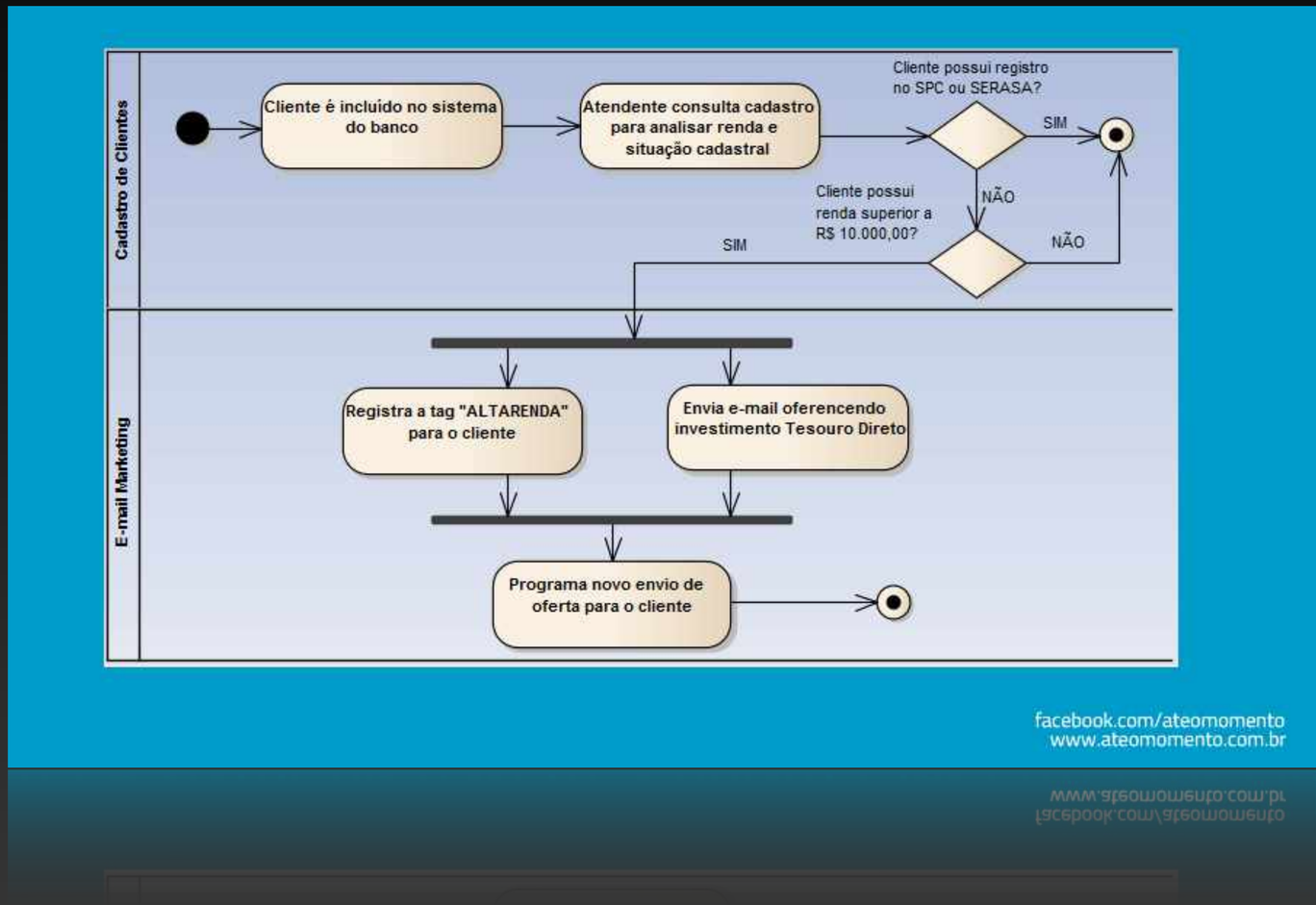


Diagramas Comportamentais

- Especificam detalhes do comportamento do sistema (parte dinâmica), por exemplo: como as funcionalidades devem funcionar, como componentes estruturais trocam mensagens e como respondem às chamadas etc.
- Diagrama de Casos de Uso - voltado à apresentação de funcionalidades e características de um sistema.
- Diagrama de Transição de Estados - detalha os diferentes estados pelos quais pode passar um objeto, tomando por base a execução de um processo dentro do sistema que se está considerando.

Diagramas Comportamentais

Diagrama de Atividades - descreve os passos a serem percorridos para a conclusão de uma atividade.

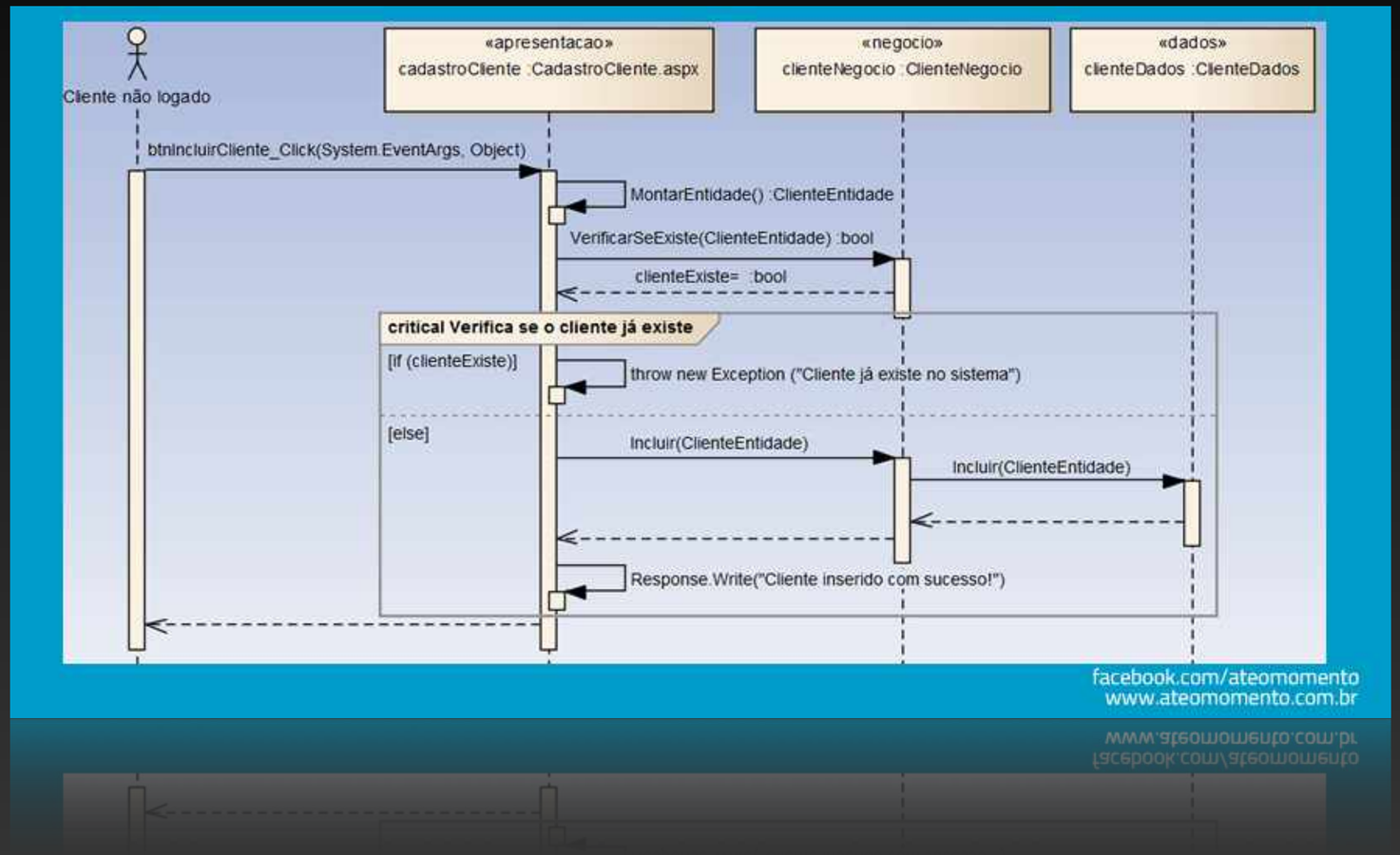


Diagramas Comportamentais

- Diagrama de Interação - considerados um subgrupo dos diagramas comportamentais, sendo normalmente utilizados na representação de interações entre objetos de uma aplicação.
- Diagrama de visão geral - mostra o fluxo principal das interações dentro do sistema.
- Diagrama de comunicação - complementa o anterior, mostrando os vínculos entre cada objeto.
- Diagrama de tempo - descreve o comportamento das instâncias de uma classe durante um intervalo específico.

Diagramas Comportamentais

Diagrama de sequência - descreve a interação entre os objetos de um caso de uso.



FDD

Feature Driven Development

O que é?

- É um método leve e interativo para desenvolvimento de Software que foi criado por Jeff de Luca e Peter Coad em 1997 em Cingapura.
- Combina as melhores práticas do gerenciamento ágil de projetos com uma abordagem completa de engenharia de software orientada a objetos e possui como lema: “Resultados frequentes, tangíveis e funcionais”.

O que são Features?

- São características ou Funcionalidades que representam algum valor para o cliente e devem ser expressas da seguinte forma:

<action> <result> <object>

Exemplo

<Calcular> a <nota final> do <aluno>

- Calcular: Ação
- Nota final: Resultado
- Aluno: Objeto

Práticas do FDD

- Modelagem dos objetos de domínio
- Desenvolvimento através de funcionalidades
- Propriedade individual das classes
- Equipes de funcionalidades
- Inspeções
- Construções regulares
- Administração de configuração
- Relatórios de resultados

Papéis principais

- Gerente de projeto
- Arquiteto chefe
- Gerente de desenvolvimento
- Programadores chefes
- Proprietários das classes
- Especialistas do domínio

Papéis de apoio

- Gerente do domínio
- Gerente de versão
- Especialista de linguagem
- Coordenador de construção
- Ferramenteiro/toolsmith
- Administrador de Sistema

Papéis de adicionais

- Testador
- Desenvolvedores
- Escritor técnico

Fases do FDD

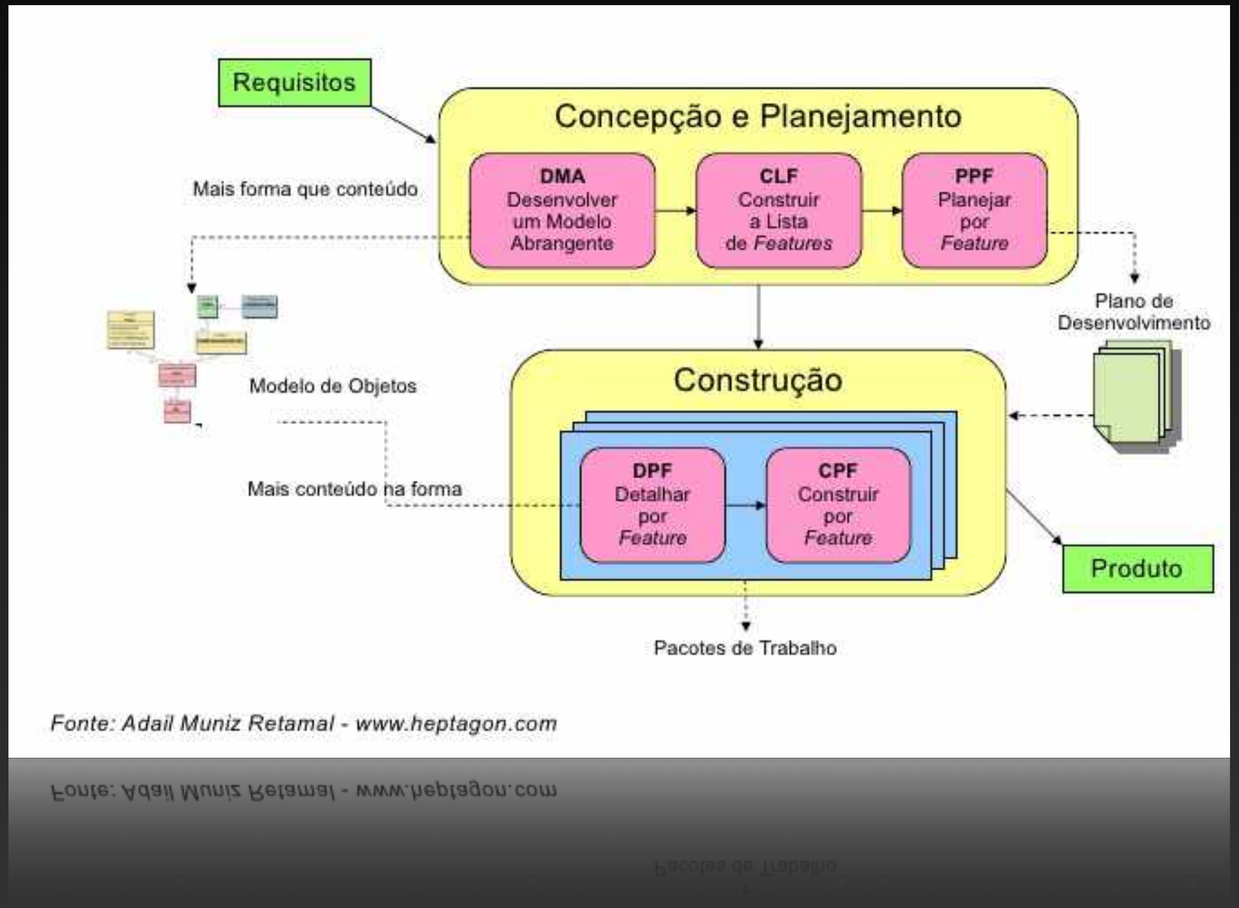
Concepção e Planejamento: pensar antes de fazer (1 a 2 semanas):

- Desenvolvimento de modelo abrangente (Análise orientada por objetos)
- Construção de lista de funcionalidades (Decomposição funcional)
- Planejar por funcionalidade (Planejamento Incremental)

Construção: fazer de forma iterativa (2 semanas)

- Detalhar por funcionalidade (Desenho orientado a objetos)
- Construção por funcionalidade (Programação e teste orientado a objetos)

“ A soma das partes é maior do que o todo”



Percentual de tempo gasto em cada etapa

Projetar pelas características			Construir pelas características		
Análise do domínio	Projeto	Inspeção do projeto	Código	Inspeção do código	Geração de build
1%	40%	3%	45%	10%	1%

Vantages do FDD

- É um método ágil e altamente adaptável
- Oferece vantagens dos métodos pesados
- Oferece vantagens dos métodos extremamente ágeis
- É orientada às necessidades dos clientes, gerentes e desenvolvedores

Referências

- https://meusucesso.com/artigos/gestao/canvas-passo-a-passo-para-o-modelo-de-negocios-1616/
- https://www.agendor.com.br/blog/dor-do-cliente/
- devmedia.com.br/introducao-ao-extreme-programming-xp/29249
- hiperbytes.com.br/xp/metodologia-xp-extreme-programming-breve-historico-da-xp/
- robsoncamargo.com.br/blog/Extreme-Programming
- treinaweb.com.br/blog/o-que-e-xp-extreme-programming/
- https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_extrema

Referências

- https://www.mundodevops.com/blog/o-que-e-integracao-continua/
- https://www.opus-software.com.br/o-que-e-integracao-continua/#:~:text=Integra%C3%A7%C3%A3o%20cont%C3%ADnua%20(continuous%20integration)%20%C3%A9,realizado%20com%20entregas%20mais%20frequentes.
- https://mundodevops.com.br/blog/principais-ferramentas-de-integracao-continua/
- https://aws.amazon.com/pt/devops/continuous-integration/
- https://blog.iprocess.com.br/2012/10/soa-arquitetura-orientada-a-servicos/#:~:text=SOA%20significa%20Service-Oriented%20Architecture,Yefim%20Natis%20do%20Gartner%20Group
- https://www.opus-software.com.br/o-que-e-soa-e-quais-os-beneficios/

Referências

- https://pt.wikipedia.org/wiki/Scrum_(desenvolvimento_de_software)#História
- https://www.desenvolvimentoagil.com.br/scrum/
- https://blog.betrybe.com/tecnologia/uml/
- https://www.infoescola.com/engenharia-de-software/uml/
- https://www.devmedia.com.br/modelagem-de-sistemas-atraves-de-uml-uma-visao-geral/27913
- https://pt.qwe.wiki/wiki/Booch_method#:~:text=O%20m%C3%A9todo%20Booch%20%C3%A9%20um,um%20conjunto%20de%20pr%C3%A1ticas%20recomendadas.
- FDD Numa Casca de Banana, Um guia de rápido aprendizado para a Feature Driven Development; Mar 2007; Alexandre Magno Figueiredo
- https://www.devmedia.com.br/introducao-ao-fdd-feature-driven-development/27971
- https://www.featuredrivendevelopment.com/
- https://homepages.dcc.ufmg.br/~figueiredo/disciplinas/aulas/uml-intro_v01.pdf
- https://www.ateomomento.com.br/diagramas-uml/

Obrigado