Fatec Guaratinguetá

Centro Paula Souza

Governo do Estado de São Paulo

Engenharia de Software I

MODELOS DE PROCESSOS DE SOFTWARE

Prof. Claudemir Santos Pinto

profdemir@yahoo.com.br

Modelos de processo de software

Processo de software: É uma sequencia coerente de práticas que objetiva o desenvolvimento ou evolução de sistemas de software. Estas práticas englobam as atividades de especificação, projeto, implementação, testes e caracterizam-se pela interação de ferramentas, pessoas e métodos.

Não há um processo ideal e até dentro da mesma empresa pode haver muitos processos diferentes utilizados para o desenvolvimento de software

Modelos de processos de software

- A busca pela qualidade e menores custos produzem uma mudança cultural que permite o desenvolvimento crescente de abordagens mais maduras para a Engenharia de Software.

- Existem vários modelos de processo de software (ou paradigmas de engenharia de software)
- Cada um representa uma tentativa de colocar ordem em uma atividade inerentemente caótica
- Definem a sequência em que as atividades do processo serão realizadas

Exemplos de Modelos de Processos de Software

1. Modelo Cascata ou Clássico
2. Modelo Evolucionário
3. Engenharia de Software Baseada em Componentes
4. Processos Iterativos
5. Processo Unificado
6. Prototipação

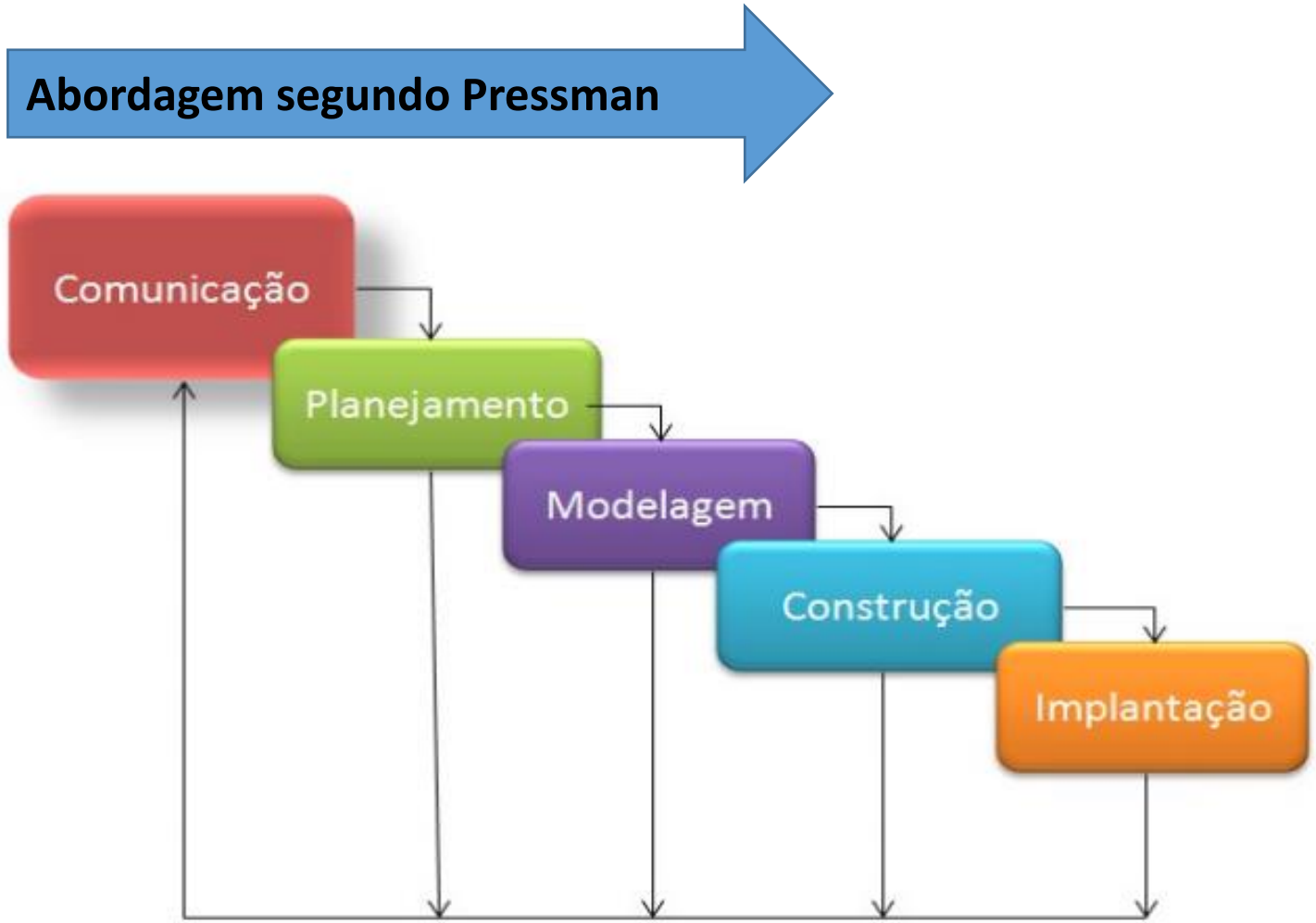
Modelo Cascata

(MODELO CLÁSSICO / WATERFALL)

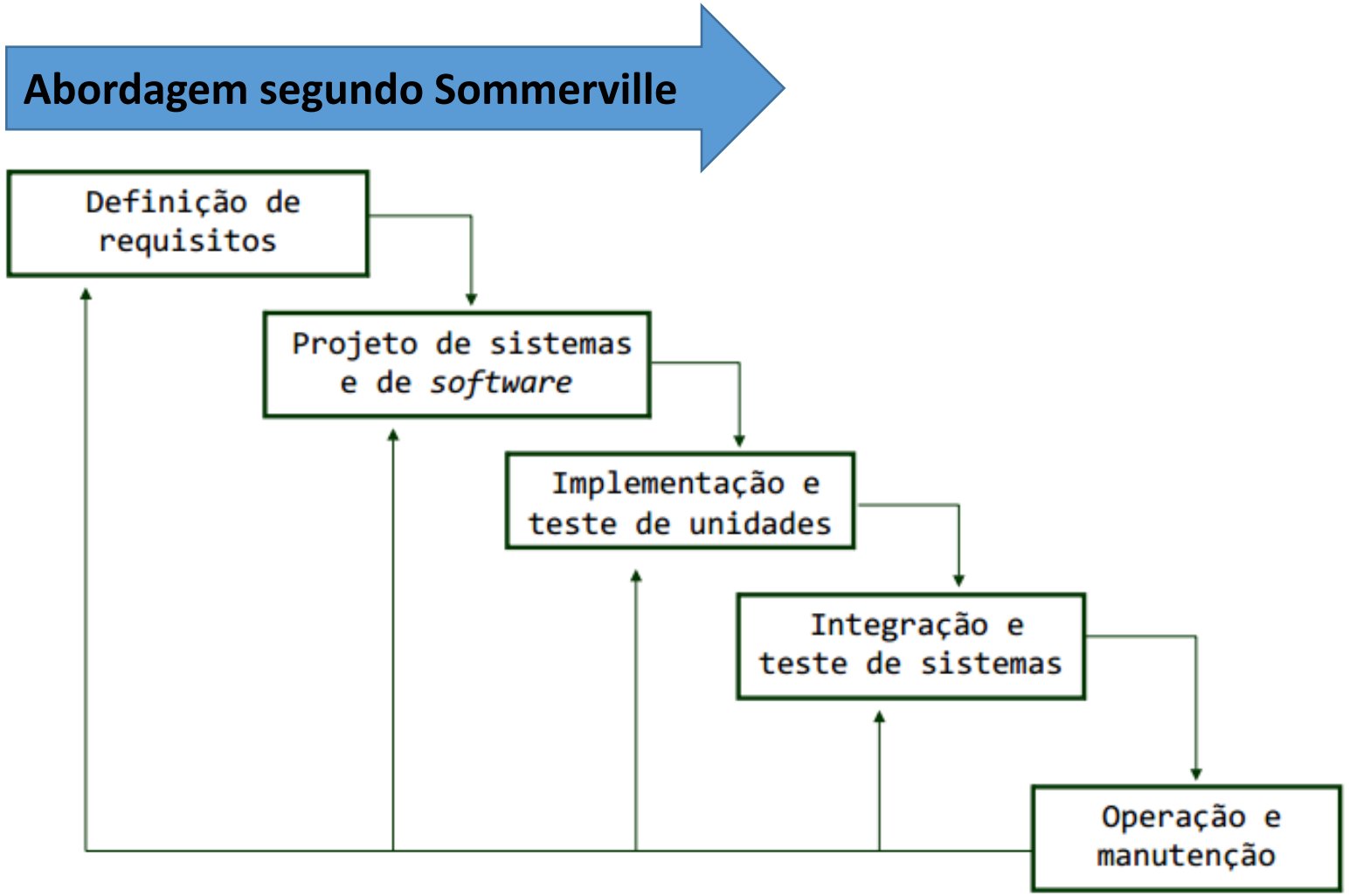
Modelo Cascata

- Primeiro modelo publicado do processo de desenvolvimento de software. Também conhecido como Ciclo de Vida Clássico ou Modelo Clássico
- Modelado em função do ciclo da engenharia convencional
- Requer uma abordagem **sistemática**, sequencial ao desenvolvimento de software
- O resultado de uma fase se constitui na entrada da outra

Modelo Cascata



Modelo Cascata



Modelo Cascata

1. Análise e definição de requisitos

(especificação de requisitos)

- As funções, as restrições e os objetivos do sistema são estabelecidos por meio da consulta aos usuários do sistema e outros interessados

- Em seguida, são definidos em detalhes e servem como uma especificação do sistema

Modelo Cascata

2. Projeto de sistemas e de software

- Agrupa os requisitos em sistemas de hardware ou de Software
- Estabelece uma arquitetura do sistema geral

Modelo Cascata

3. Implementação e teste de unidades

- Durante esse estágio, o projeto de software é compreendido como um conjunto de programas ou de unidades de programa

- O teste de unidade envolve verificar que cada unidade atenda a sua especificação

Modelo Cascata

4. Integração e teste de sistemas

- As unidades de programa ou programas individuais são integrados e testados como um sistema completo a fim de garantir que os requisitos de software foram atendidos

- Depois dos testes, o sistema de software é entregue ao cliente

Modelo Cascata

5. Operação e Manutenção

- Normalmente, esta é a fase mais longa do ciclo de vida
- Provavelmente o software deverá sofrer mudanças depois que for entregue ao cliente
- Causas das mudanças: erros, adaptação do software para acomodar mudanças em seu ambiente externo e exigência do cliente para acréscimos funcionais em função da descoberta de novos requisitos e também melhora de desempenho

Problemas com o Modelo Cascata

- Em princípio, o resultado de cada fase envolve um ou mais documentos que são aprovados. Gera muita documentação, nem sempre utilizada posteriormente
- A fase seguinte não deve iniciar até que a fase precedente tenha sido concluída
- Logo no início é difícil estabelecer explicitamente todos os requisitos. No começo dos projetos sempre existe uma incerteza natural
- O cliente deve ter paciência. Uma versão executável do software só fica disponível numa etapa avançada do desenvolvimento (na instalação)

Problemas com o Modelo Cascata

Embora o Modelo em Cascata tenha fragilidades, ele é significativamente melhor do que uma abordagem casual de desenvolvimento de software

Contribuições do Modelo Cascata

O Modelo de processo em Cascata trouxe contribuições importantes para o processo de desenvolvimento de software:

- Imposição de disciplina, planejamento e gerenciamento, a implementação do produto deve ser postergada até que os objetivos tenham sido completamente entendidos;

- Permite gerência do baseline, que identifica um conjunto fixo de documentos produzidos ao longo do processo de desenvolvimento

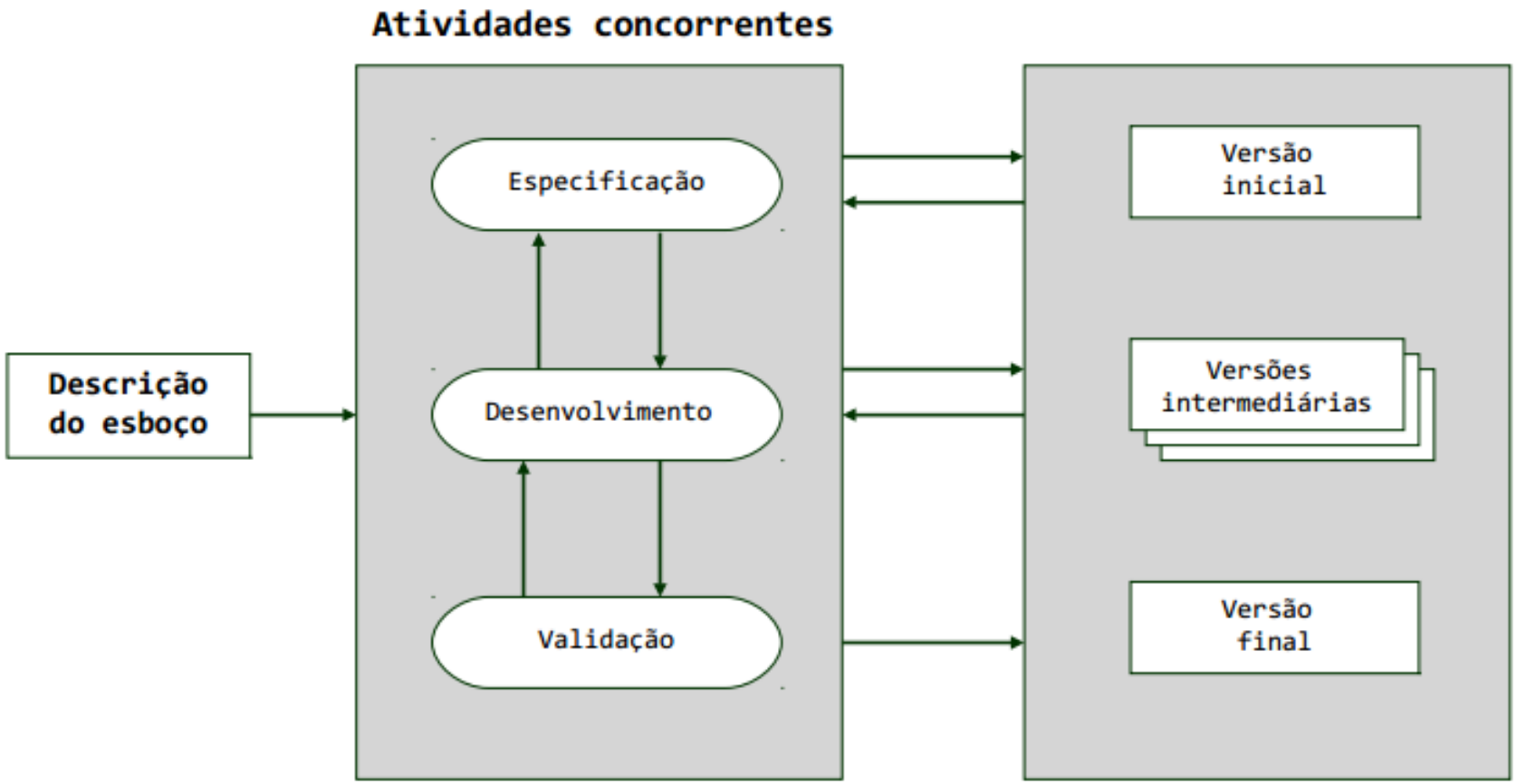
Modelo Evolucionário

Modelo Evolucionário

- Tem como base a ideia de desenvolver uma implementação inicial, expor o resultado ao comentário do usuário e fazer seu aprimoramento por meio de muitas versões, até que um sistema adequado tenha sido desenvolvido

- Em vez de ter as atividades de especificação, desenvolvimento e validação em separado, todo esse trabalho é realizado concorrentemente com um rápido feedback por meio dessas atividades

Modelo Evolucionário



Modelo Evolucionário

VANTAGENS:

- A abordagem evolucionária do desenvolvimento de software, muitas vezes, é mais eficaz do que a abordagem em cascata, no sentido de produzir sistemas que atendam às necessidades imediatas dos clientes
- A especificação pode ser desenvolvida gradativamente. À medida que os usuários desenvolvem uma compreensão melhor de seus problemas, isso pode ser refletido na melhoria do software em construção

Modelo Evolucionário

DESVANTAGENS:

- Como os softwares são desenvolvidos rapidamente, não é viável produzir documentos que reflitam cada versão do sistema

- Os softwares frequentemente são mal estruturados → a mudança constante tende a corromper a estrutura do software

- Incorporar modificações torna-se cada vez mais difícil e oneroso

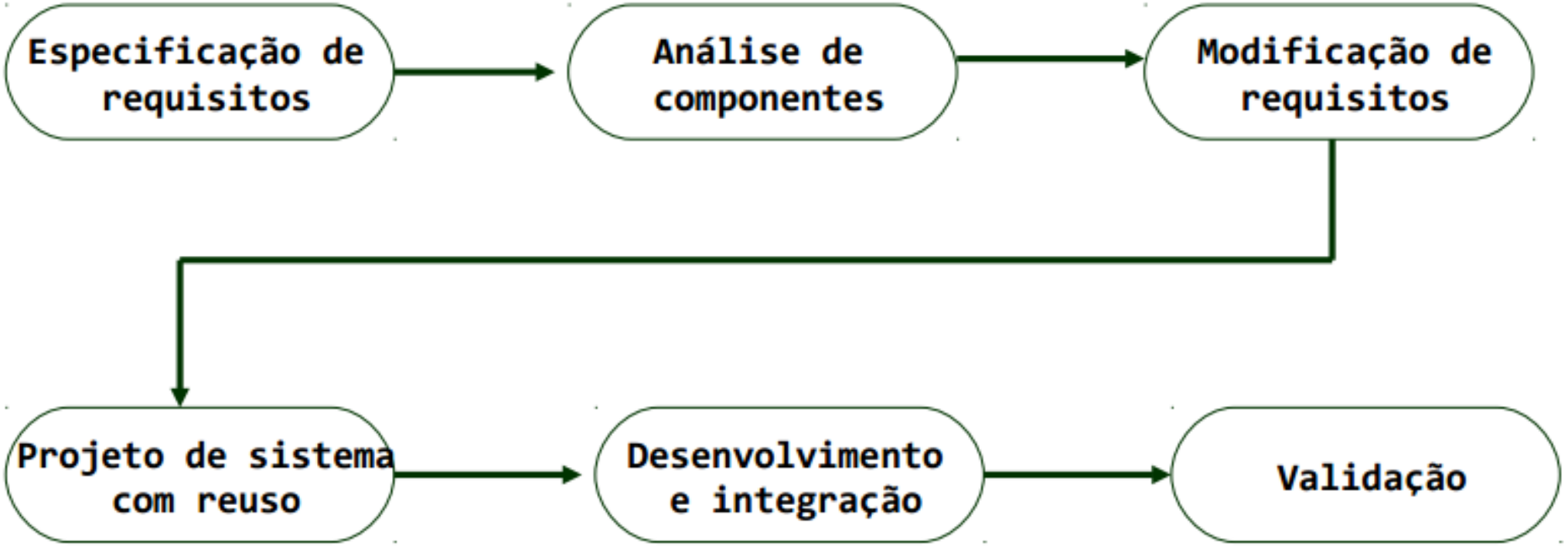
Engenharia de Software baseada em componentes

Engenharia de Software baseada em componentes

- Essa abordagem tem como base a existência de um número significativo de componentes reutilizáveis

- O processo de desenvolvimento de software se concentra na integração desses componentes, ao invés de proceder ao desenvolvimento a partir do zero

Modelo genérico de processo para a Engenharia de Software baseada em componentes



Engenharia de Software baseada em componentes

ESPECIFICAÇÃO DOS REQUISITOS

- Comparável com outros processos, como por exemplo, o modelo cascata

- As funções, as restrições e os objetivos do software são estabelecidos por meio da consulta aos usuários

- Em seguida, são definidos em detalhes e servem como uma especificação do software

Engenharia de Software baseada em componentes

ANÁLISE DE COMPONENTES

- Com base na especificação de requisitos, é feita uma busca de componentes para implementar essa especificação

- Nem sempre é possível encontrar uma combinação exata e os componentes que podem ser utilizados fornecem somente parte da funcionalidade requerida

Engenharia de Software baseada em componentes

MODIFICAÇÃO DE REQUISITOS

- Durante esse estágio, os requisitos são analisados, utilizando-se as informações sobre os componentes que foram encontrados

- Eles são então modificados para refletir os componentes disponíveis

- Quando as modificações forem impossíveis, a atividade de análise de componentes é refeita, a fim de procurar soluções alternativas

Engenharia de Software baseada em componentes

PROJETO DE SISTEMA COM REUSO

- O software que não puder ser comprado, será desenvolvido, e os componentes e sistemas COTS (commercial off-the-shelf– sistemas comerciais de prateleira) serão integrados, a fim de atender por completo a especificação do usuário

Engenharia de Software baseada em componentes

VALIDAÇÃO DO SISTEMA

- O software deve ser validado para garantir que atende a especificação do usuário

Engenharia de Software baseada em componentes

VANTAGENS

- Reduz a quantidade de software a ser desenvolvida, reduzindo custos e riscos
- Geralmente propicia a entrega mais rápida do software

Engenharia de Software baseada em componentes

DESVANTAGENS

- As adequações nos requisitos são inevitáveis, e isso pode resultar em um software que não atenda às reais necessidades dos usuários

- O controle sobre a evolução do software se perde, uma vez que novas versões dos componentes reutilizáveis não estão sob o controle da organização que utiliza esses componentes

Atividade de Fixação

1. Explique uma das etapas do Modelo Cascata
2. Cite uma característica do Modelo Cascata (positiva ou negativa)
3. Explique com suas palavras o Modelo Evolucionário
4. Cite uma vantagem e uma desvantagem do Modelo Evolucionário
5. Em sua opinião, qual a principal característica da Engenharia de Software Baseada em Componentes (ESBC) ?
6. Cite uma vantagem e uma desvantagem da ESBC.

Tarefa agendada. Entrega até 08/05/2020 via Teams.

Processos Iterativos

Modelo Incremental

- A cada entrega, os requisitos são refinados para que haja a expansão das funcionalidades
- Utilizado quando os requisitos iniciais são razoavelmente bem definidos e exige-se que o cliente tenha contato com um conjunto funcional do software para que ele, posteriormente, seja refinado e expandido.

Modelo Incremental

- Produz softwares com funções básicas nos primeiros incrementos, para posteriormente trazer funções mais sofisticadas.

- O primeiro incremento é sempre o produto essencial, com a entrega dos requisitos básicos.

- O planejamento já considera a mudança do produto essencial, para melhor se adequar à necessidade do cliente e a os novos incrementos

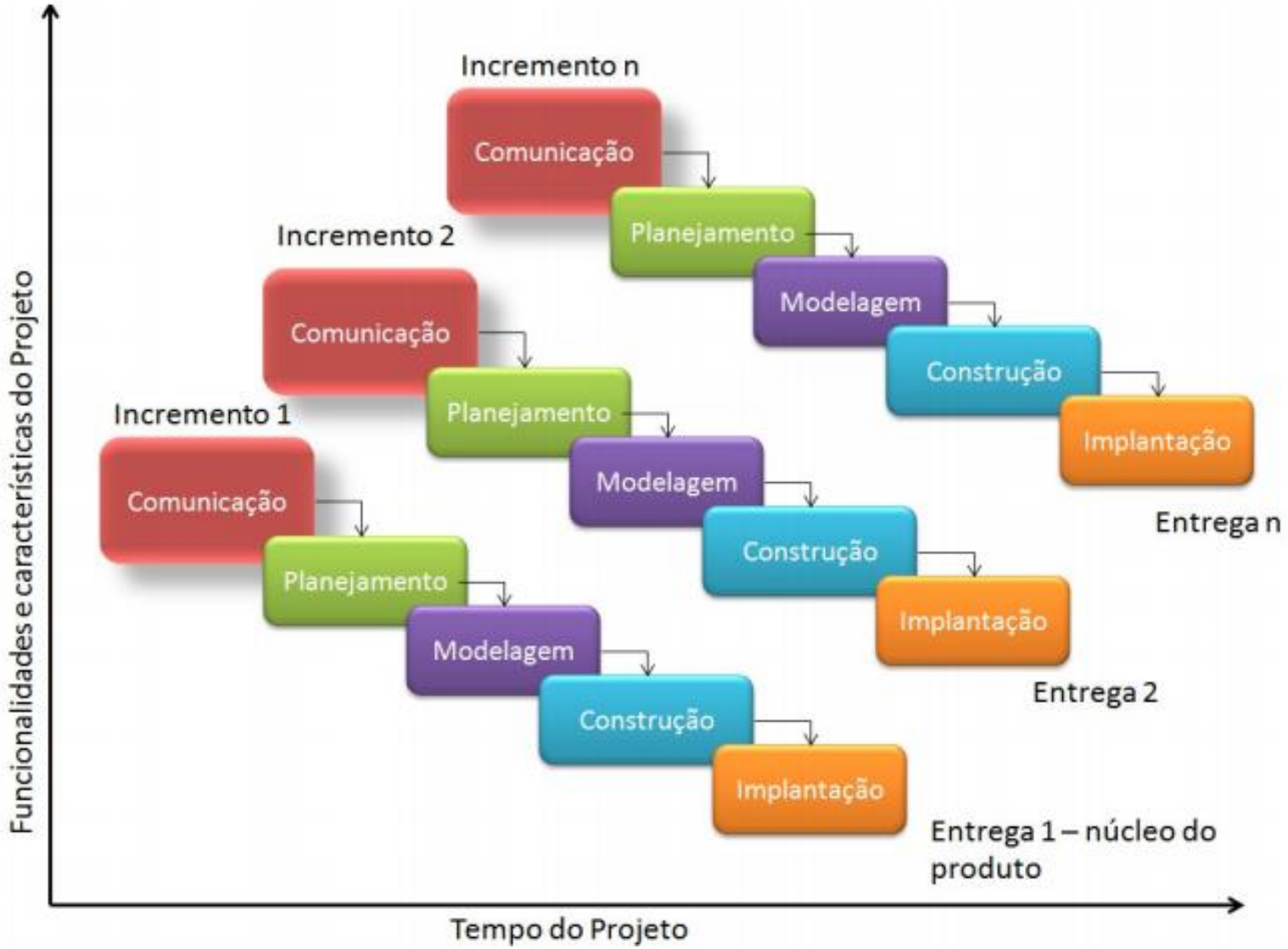
Modelo Incremental

- Situações em pode ser útil:

– Quando o software completo exige uma infraestrutura ainda não disponível e com data de entrega incerta

– Quando a equipe disponível é insuficiente para assumir o projeto completo, entregando num prazo factível para o usuário

Modelo Incremental



Processos Iterativos

Evolucionário x Incremental → uma metáfora

Evolucionário →

Incremental →

Modelo Processo Unificado

Processo Unificado

O **RUP**, abreviação de **Rational Unified Process** (ou Processo Unificado da Rational), é um processo proprietário de Engenharia de software criado pela Rational Software Corporation.

O RUP usa a abordagem da orientação a objetos em sua concepção e é projetado e documentado utilizando a notação UML (*Unified Modeling Language*) para ilustrar os processos em ação.

Processo Unificado

É um processo considerado pesado e preferencialmente aplicável a grandes equipes de desenvolvimento e a grandes projetos, porém o fato de ser amplamente customizável torna possível que seja adaptado para projetos de qualquer escala.

Para a gerência do projeto, o RUP provê uma solução disciplinada de como assinalar tarefas e responsabilidades dentro de uma organização de desenvolvimento de software.

Fases do RUP

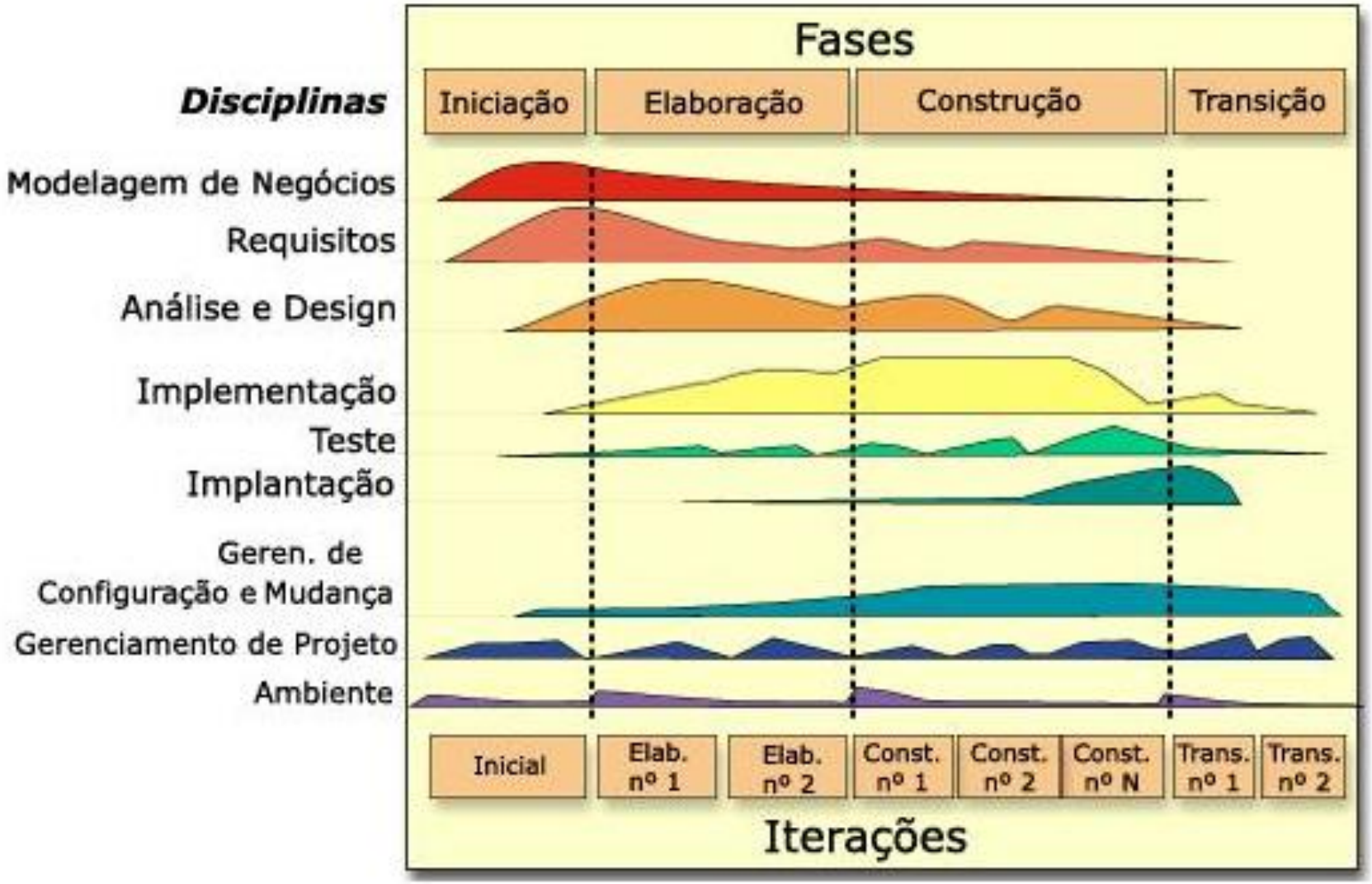
As fases indicam a ênfase que é dada no projeto em um dado instante. Para capturar a dimensão do tempo de um projeto, o RUP divide o projeto em quatro fases diferentes:

- **Concepção (Iniciação)**: fase na qual se justifica a execução de um projeto de desenvolvimento de software, do ponto de vista do negócio do cliente.
- **Elaboração**: fase na qual o produto é detalhado o suficiente para permitir um planejamento acurado da fase de construção.
- **Construção**: fase na qual é construída uma versão completamente operacional.
- **Transição**: fase na qual o produto é colocado a disposição dos usuários

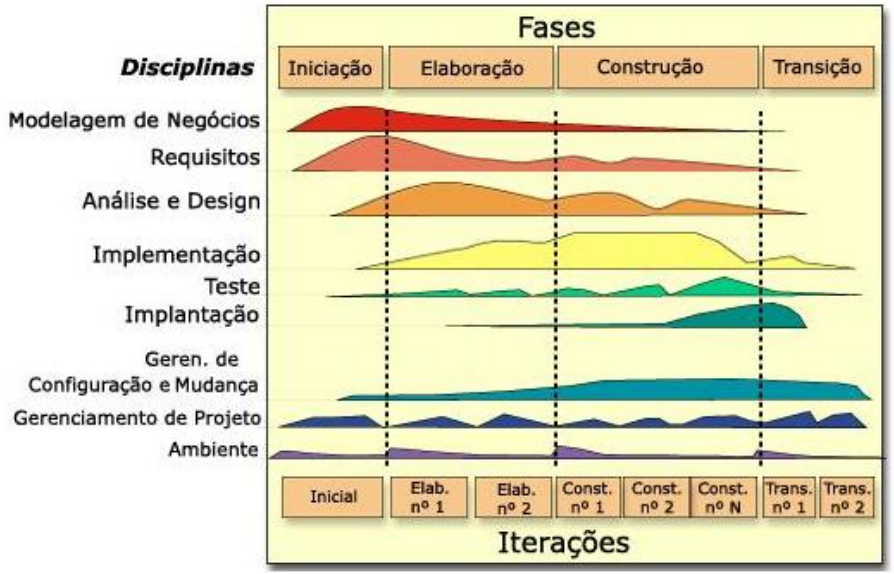
Fases do RUP

- As fases são compostas de iterações. As iterações são janelas de tempo; as iterações possuem prazo definido enquanto as fases são objetivas.
- Todas as fases geram artefatos. Estes serão utilizados nas próximas fases e documentam o projeto, além de permitir melhor acompanhamento.

Fases do RUP


Fases
Disciplinas
Iniciação
Elaboração
Construção
Transição
Modelagem de Negócios
Requisitos
Análise e Design
Implementação
Teste
Implantação
Geren. de Configuração e Mudança
Gerenciamento de Projeto
Ambiente
Inicial
Elab. nº 1
Elab. nº 2
Const. nº 1
Const. nº 2
Const. nº N
Trans. nº 1
Trans. nº 2
Iterações

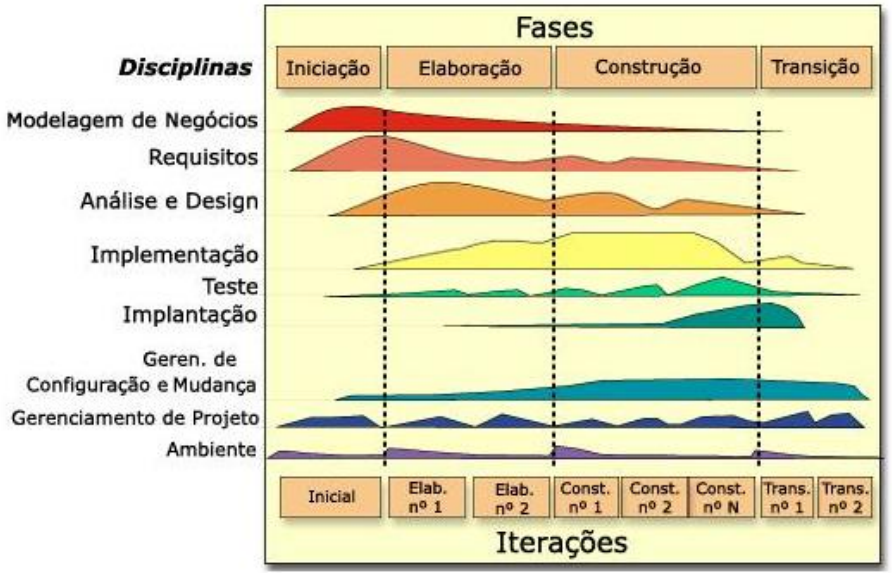
RUP – Concepção/Iniciação



<u>Ênfase no escopo do sistema:</u>

- Requisitos de negócio usando casos de uso preliminares;
- Plano de projeto, fases e iterações;
- Modelo inicial de caso de uso;
- Planejamento com recursos, riscos e cronogramas.

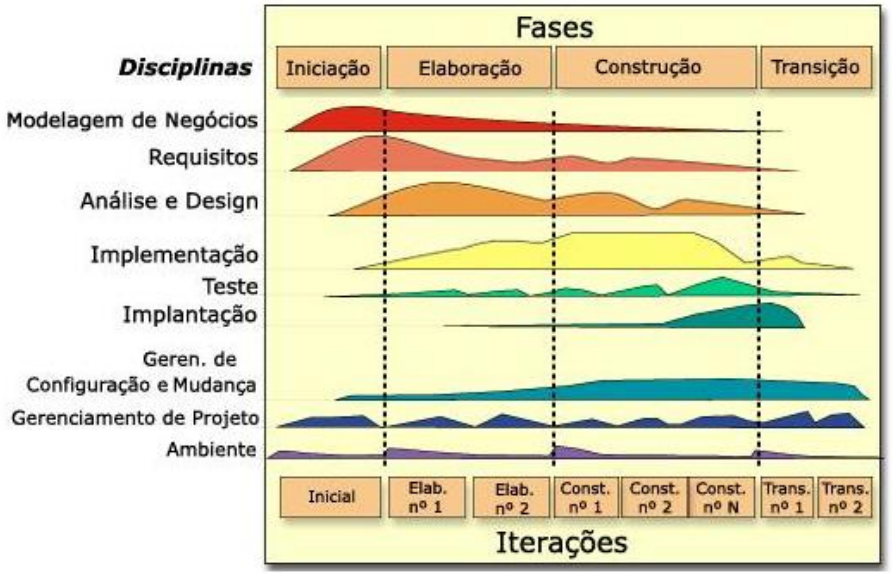
RUP - Elaboração



<u>Ênfase na arquitetura:</u>

- Refina e expande os casos de uso preliminares.
- Expande a representação arquitetural para incluir cinco visões diferentes:
  - A visão de casos de uso.
  - A visão de análise.
  - A visão de projeto.
  - A visão de implementação.
  - A visão de implantação.
- O plano é revisto e pode ser modificado.

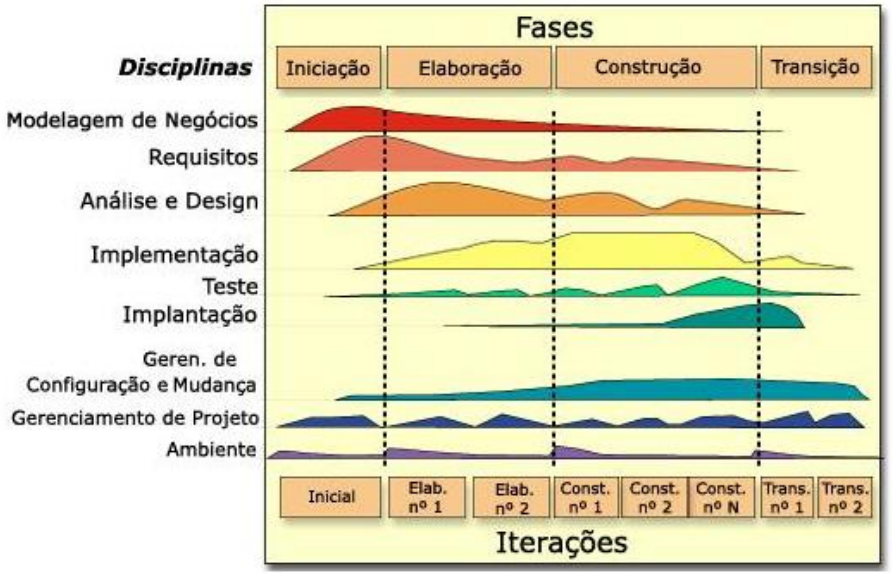
RUP - Construção



<u>Ênfase no desenvolvimento:</u>

- Usa o modelo arquitetural como entrada.
- Desenvolve ou adquire e integra componentes de software.
- Torna cada caso de uso operacional.
- Modelos de análise e projeto são completados.
- Testes são elaborados e executados.

RUP - Transição



<u>Ênfase na implantação:</u>

- O software é dado aos usuários finais para testes beta e relatórios de feedback que podem levar a modificações.
- Informações de apoio necessárias são criadas (manuais e procedimen tos de instalação).
- Na conclusão dessa fase temse uma versão utilizável do software.

RUP - Vantagens

- O usuário não espera até a conclusão do projeto para ter contato com o software, devido ao modelo incremental.

- Após o término do desenvolvimento é muito difícil encontrar novos erros.

- A complexidade é administrada; a equipe não é sobrecarregada pela “paralisia da análise” ou por passos muito longos e complexos;

RUP - Desvantagens

- Podem ocorrer divergências entre a documentação e o software.
- Pode entrar em loop devido ao modelo iterativo e incremental, dependendo do cliente para chegar ao fim do projeto.
- Aumento de gastos devido à implantação da versão a cada incremento.

Atividade de fixação

1. Cite uma característica do Modelo Incremental.
2. Cite uma característica do Modelo RUP.
3. Cite as fases do RUP. Explique uma delas.

Tarefa agendada pelo Teams.

Modelo Prototipação

Modelo Prototipação

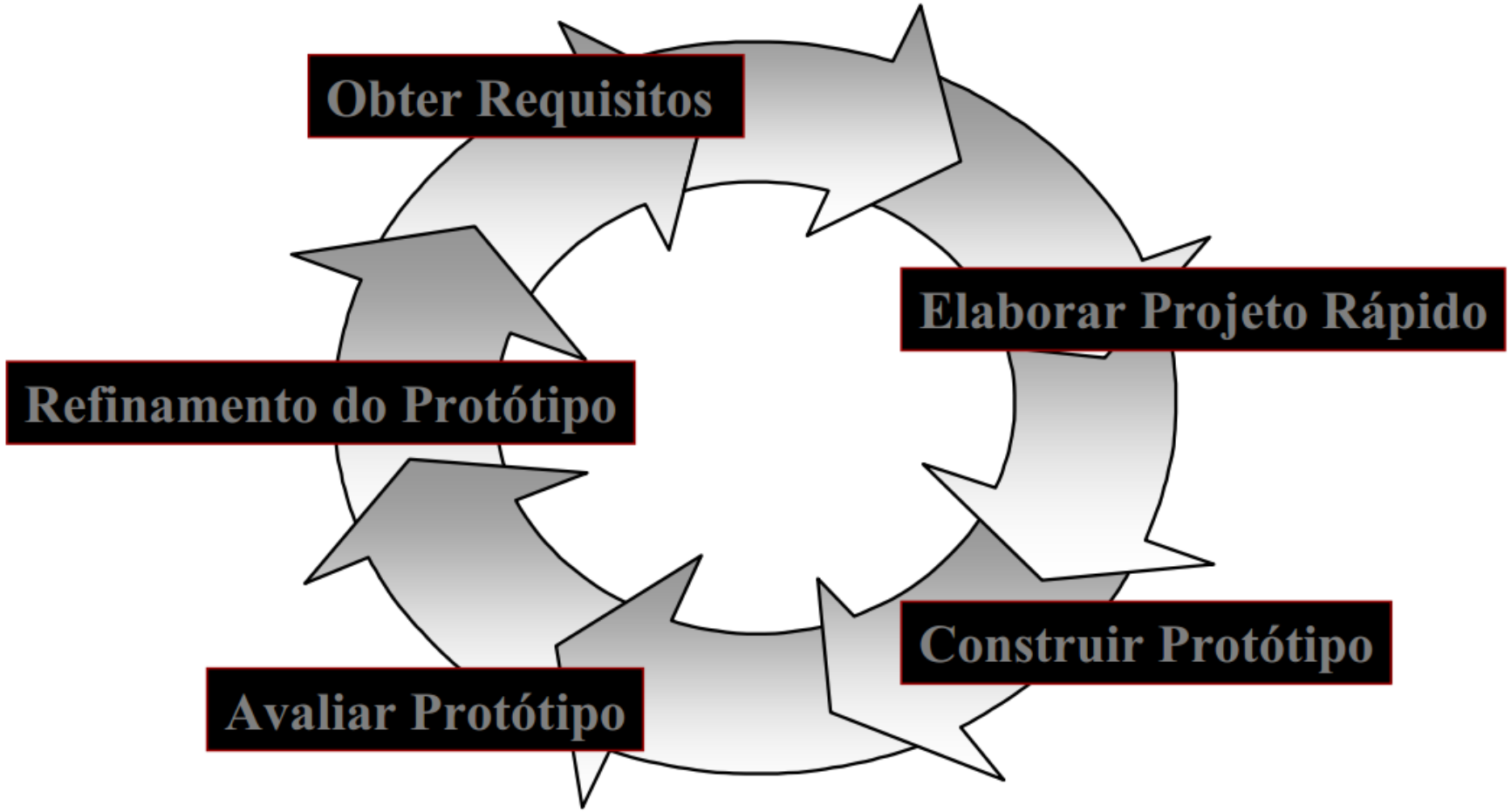
O objetivo é entender os requisitos do usuário e, assim, obter uma melhor definição dos requisitos do sistema.

Possibilita que o desenvolvedor crie um modelo (protótipo) do software que deve ser construído

É apropriado para quando o cliente não definiu detalhadamente os requisitos.

O Paradigma de Prototipação

para obtenção dos requisitos

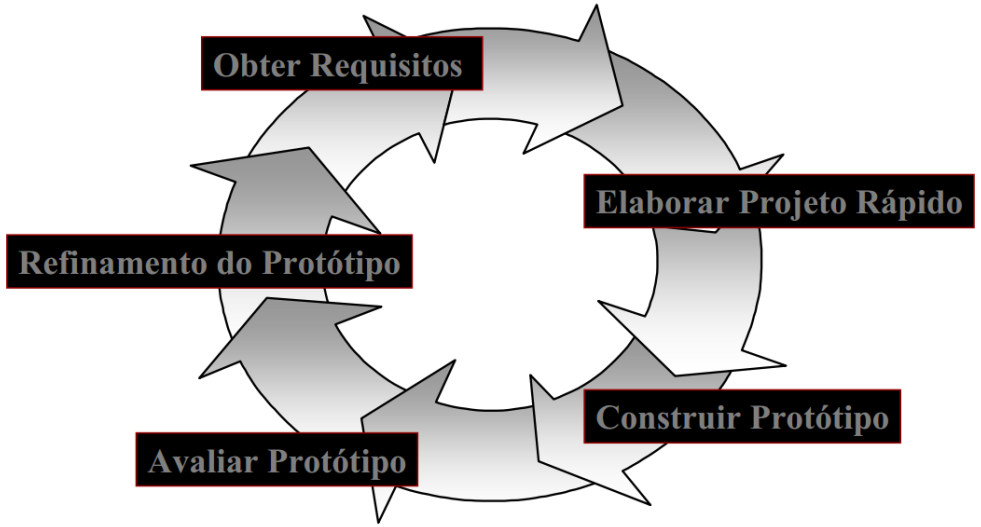


O Paradigma de Prototipação

para obtenção dos requisitos

1- OBTENÇÃO DOS REQUISITOS:

- Desenvolvedor e cliente definem os objetivos gerais do software, identificam quais requisitos são conhecidos e as áreas que necessitam de definições adicionais.

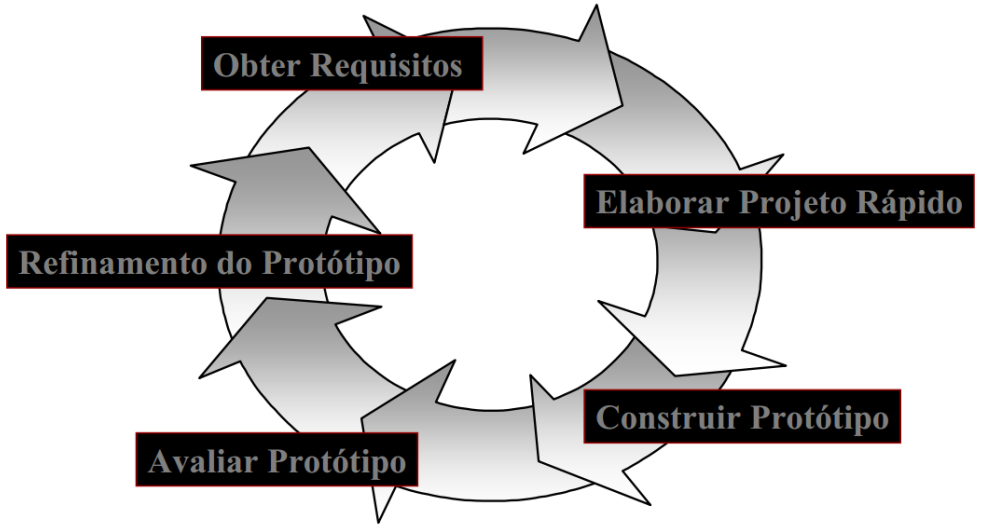


O Paradigma de Prototipação

para obtenção dos requisitos

2- PROJETO RÁPIDO:

- Representação dos aspectos do software que são visíveis ao usuário (que tipo de funcionalidade vai estar disponível para que tipo de usuário, por exemplo)

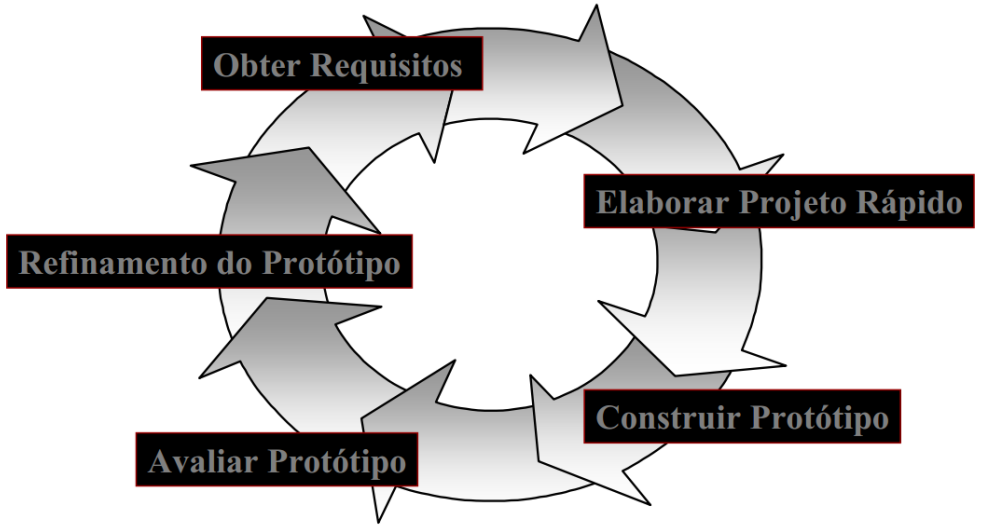


O Paradigma de Prototipação

para obtenção dos requisitos

3- CONSTRUÇÃO DO PROTÓTIPO:

- Implementação rápida do projeto

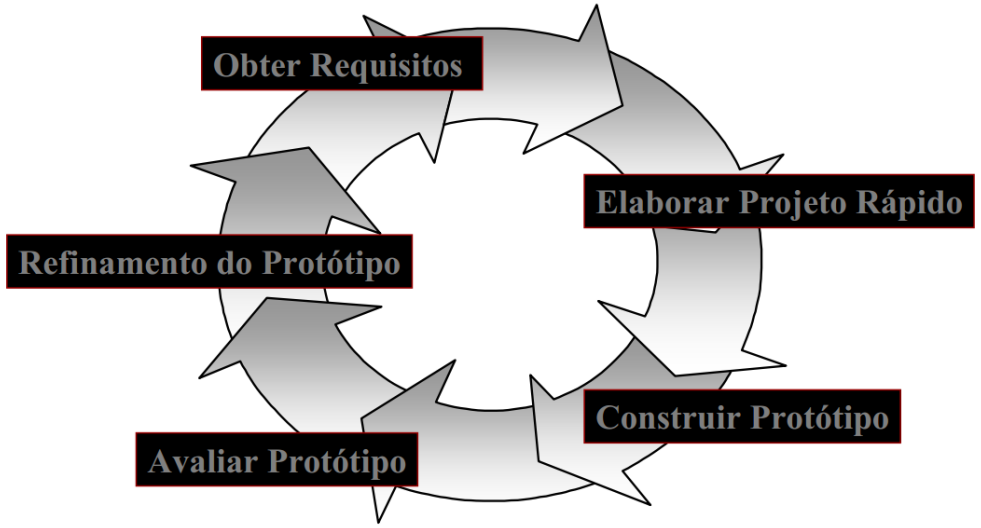


O Paradigma de Prototipação

para obtenção dos requisitos

4- AVALIAÇÃO DO PROTÓTIPO:

- Cliente e desenvolvedor avaliam o protótipo

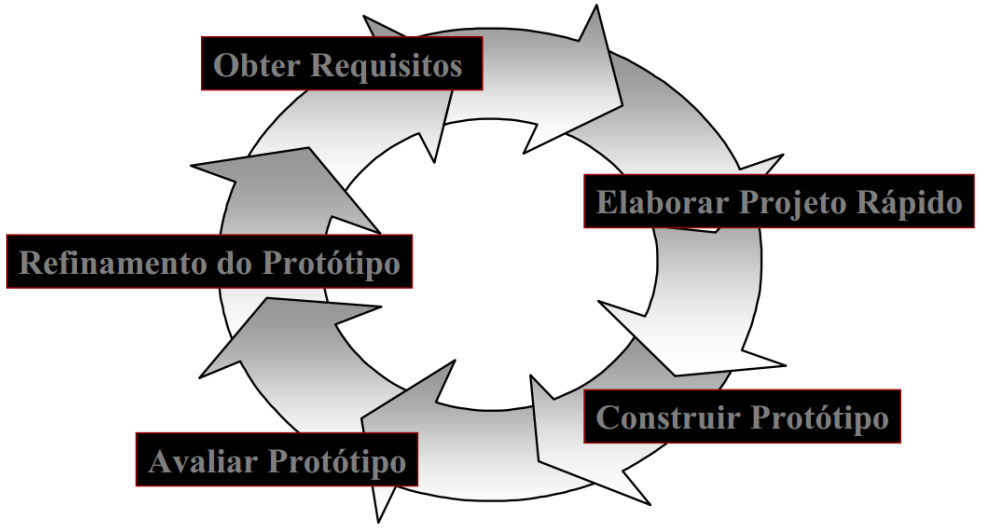


O Paradigma de Prototipação

para obtenção dos requisitos

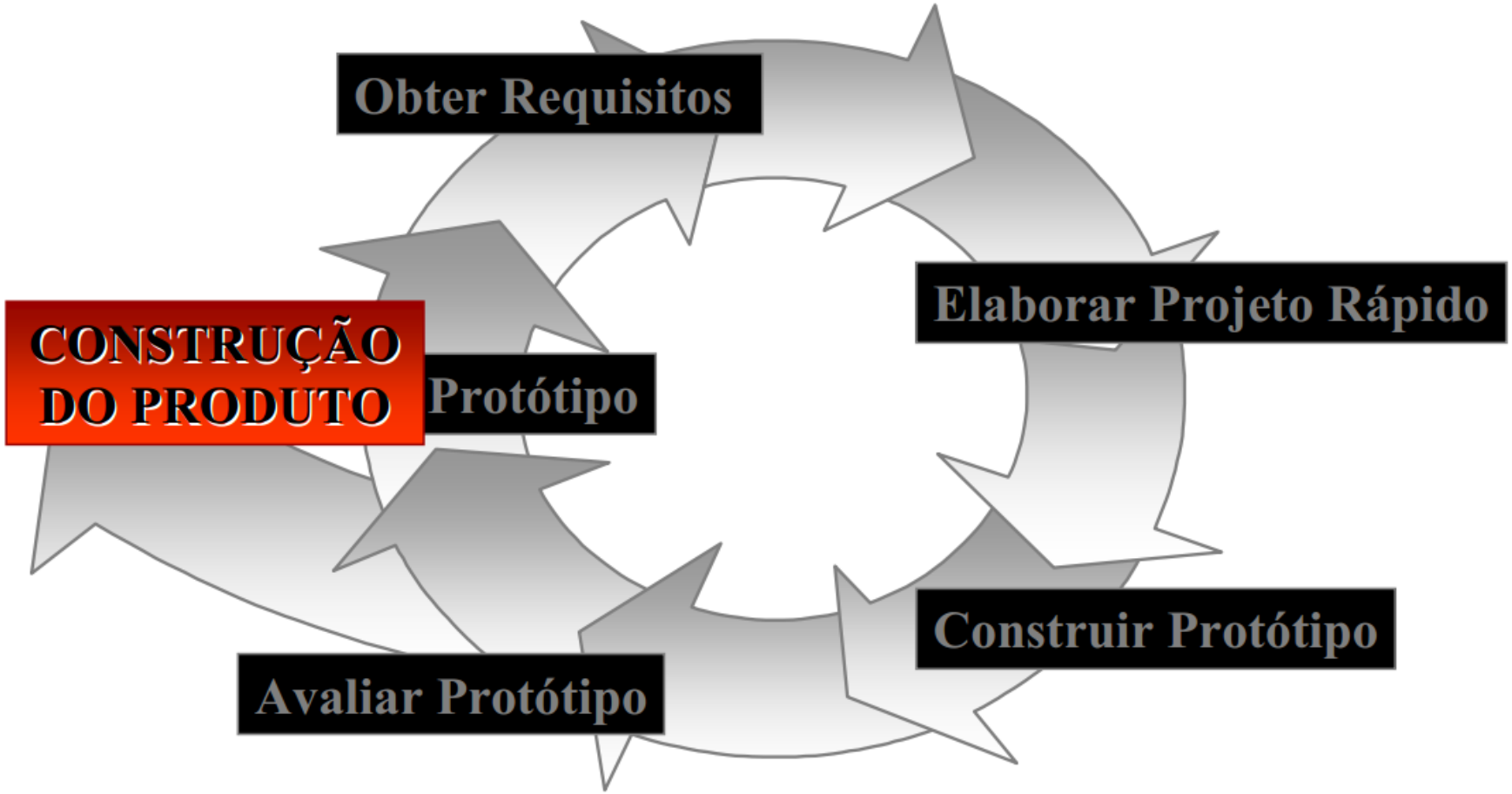
5- REFINAMENTO DO PROTÓTIPO:

- Cliente e desenvolvedor refinam os requisitos do software a ser desenvolvido.



O Paradigma de Prototipação

para obtenção dos requisitos

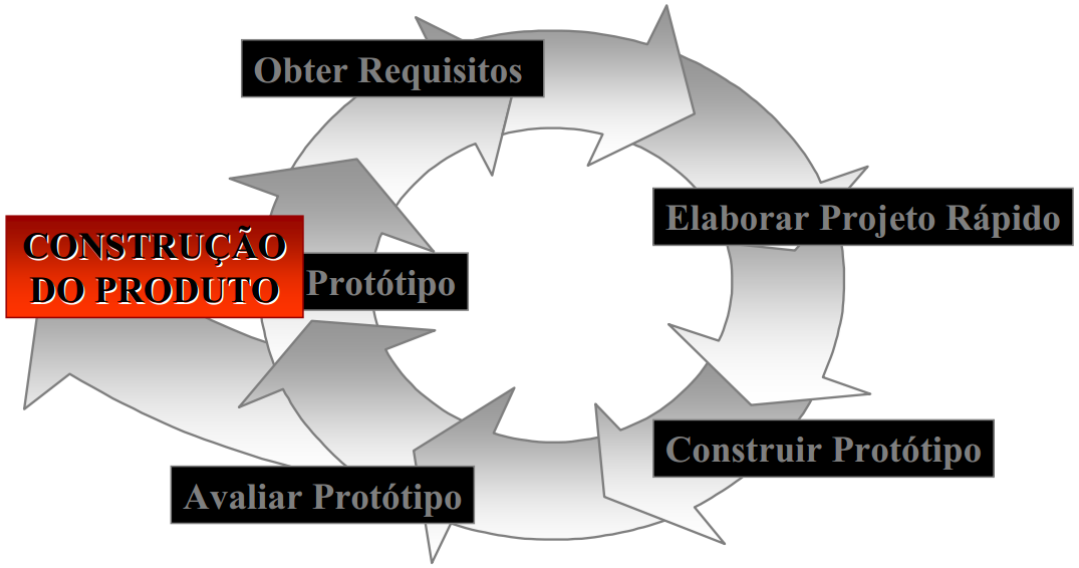


O Paradigma de Prototipação

para obtenção dos requisitos

6 - CONSTRUÇÃO PRODUTO:

- Identificados os requisitos, o protótipo deve ser descartado e a versão de produção deve ser construída considerando os critérios de qualidade.



Problemas com a Prototipação

- Cliente não sabe que o protótipo que ele vê não considerou, durante o desenvolvimento, a qualidade global e o desempenho na rede.

- Desenvolvedor frequentemente faz uma implementação comprometida (utilizando o que está disponível) com o objetivo de produzir rapidamente um protótipo

Comentários sobre o Paradigma de Prototipação

- Ainda que possam ocorrer problemas, a prototipação é um ciclo de vida eficiente.
- A chave é definir as regras do jogo logo no começo, por exemplo, saber que o protótipo ou parte dele será descartada ao final.
- O cliente e o desenvolvedor devem ambos concordar que o protótipo seja construído para servir como um mecanismo para definir os requisitos

Referências

PRESMANN, R. Engenharia de Software: uma abordagem profissional. 7. ed. Rio de Janeiro: Mc Graw Hill, 2011. Cap. 2

SOMMERVILLE, I. Engenharia de Software. 8. ed. Rio de Janeiro: Pearson, 2007. Cap. 4

Pesquisar:

Ferramentas para prototipação de software para apoiar o levantamento de requisitos

- 3 ferramentas gratuitas e 3 ferramentas pagas
- Características gerais (recursos, screenshots, preço – quando for o caso, etc)
- Site para acessar/baixar a ferramenta

Tarefa agendada pelo Teams.

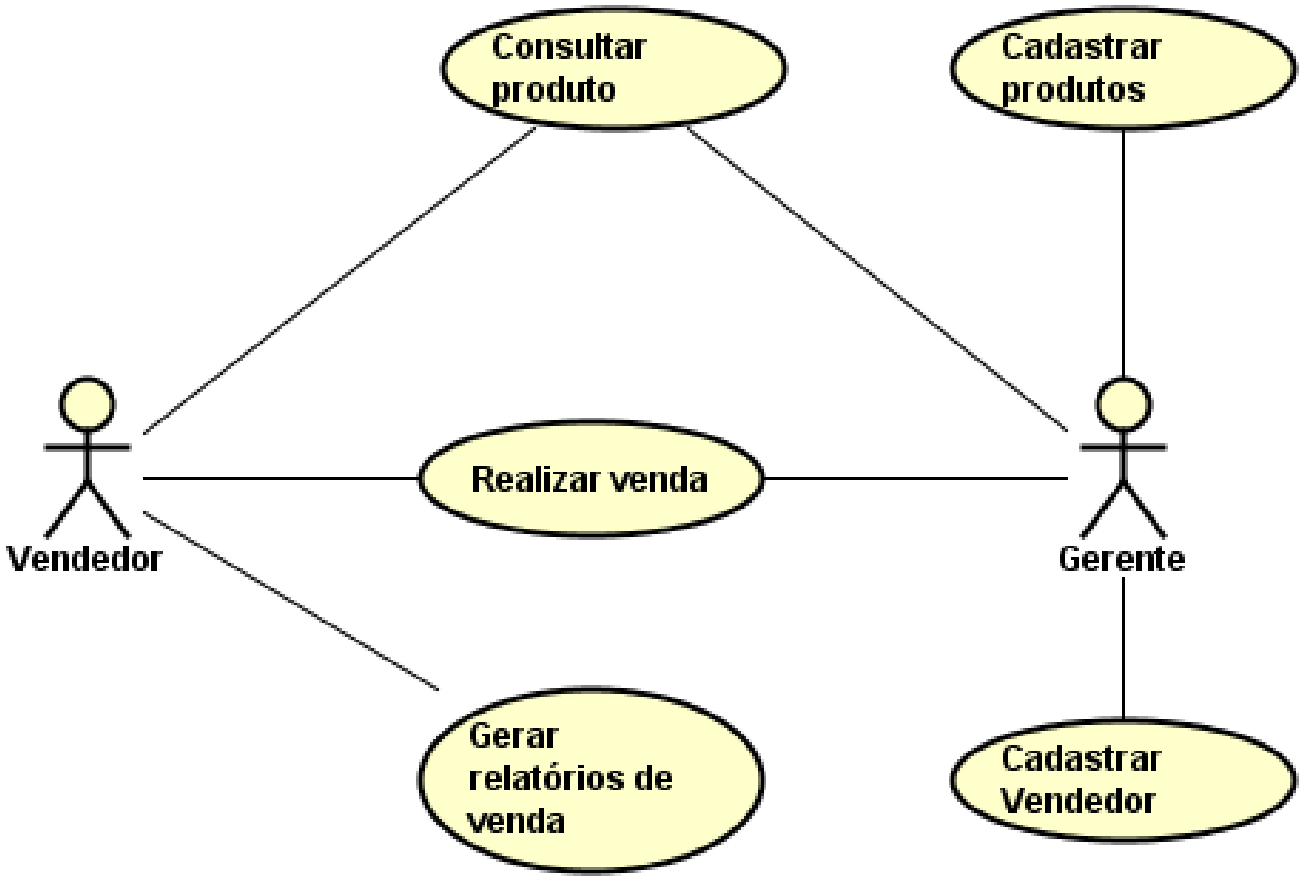
Atividade complementar

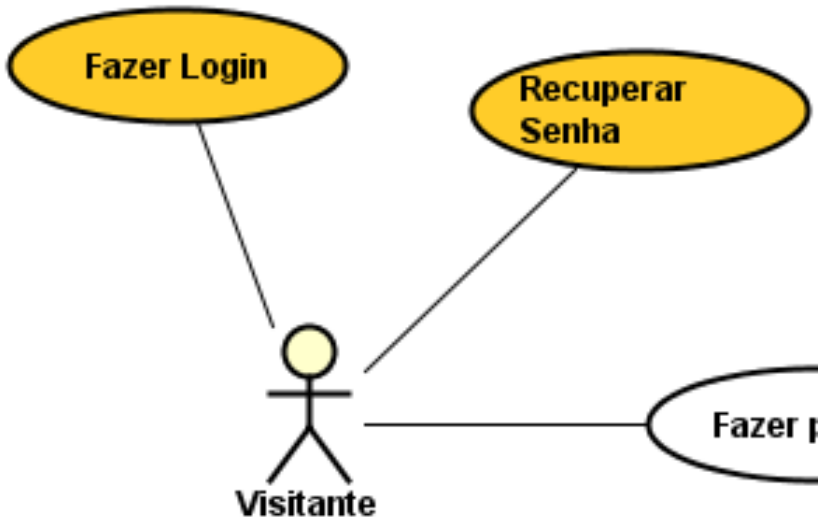
Considerando a situação atual, onde as lojas estão fechadas, escolher um comércio que você conhece e criar o protótipo de um sistema para a realização de vendas online e entregas de produtos, onde o proprietário seja capaz de gerenciar todo o processo e o cliente possa visualizar o produto, realizar compra, pagar e receber o produto em casa.

Para isso:

- Criar um diagrama de Caso de Uso com todas as funcionalidades;
- Criar um protótipo para as telas mais importantes usando uma das ferramentas pesquisadas anteriormente;
- Preparar apresentação (PowerPoint com *print* das telas) para próxima aula (20/5) e enviar para o e-mail profdemir@yahoo.com.br até o dia 19/5, com o assunto: ES1 modelo prototipação;
- Alguns serão sorteados para apresentar.

Diagrama de Casos de Uso





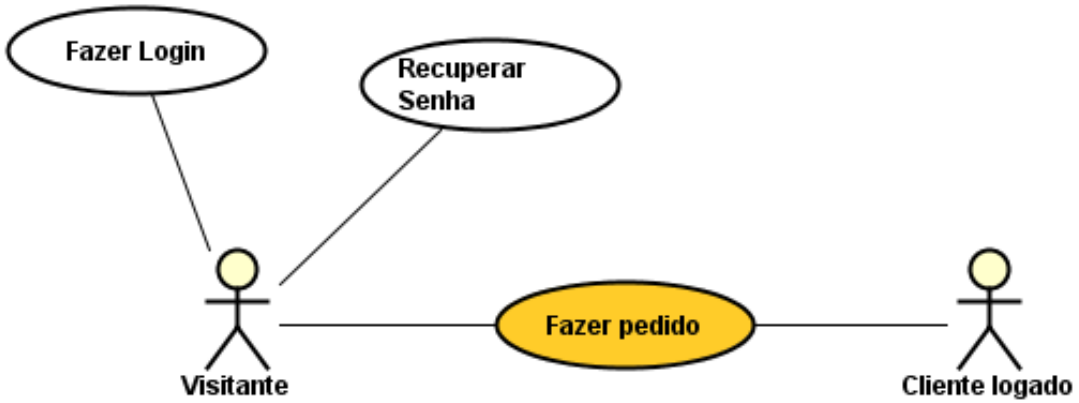
Login

Email

Senha

Esqueci a senha

Logar


Fazer Login
Recuperar Senha
Visitante
Fazer pedido
Cliente logado


Fazer login
Fazer Pedido
Home
Pizzas
Pedidos
Mussarela
Frango com Catupiry
Calabresa
Portuguesa
Baicon
Moda da casa
Brigadeiro
Banana com chocolate
Morango com chocolate
Abacaxi
Sorvete
Queijadinha
Confirmar