UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
ESCOLA AGRÍCOLA DE JUNDIAÍ
ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

CARLA FERNANDES CURVELO
CARLAFCF@GMAIL.COM

TAD0004 - ANÁLISE E PROJETO ORIENTADO A OBJETOS

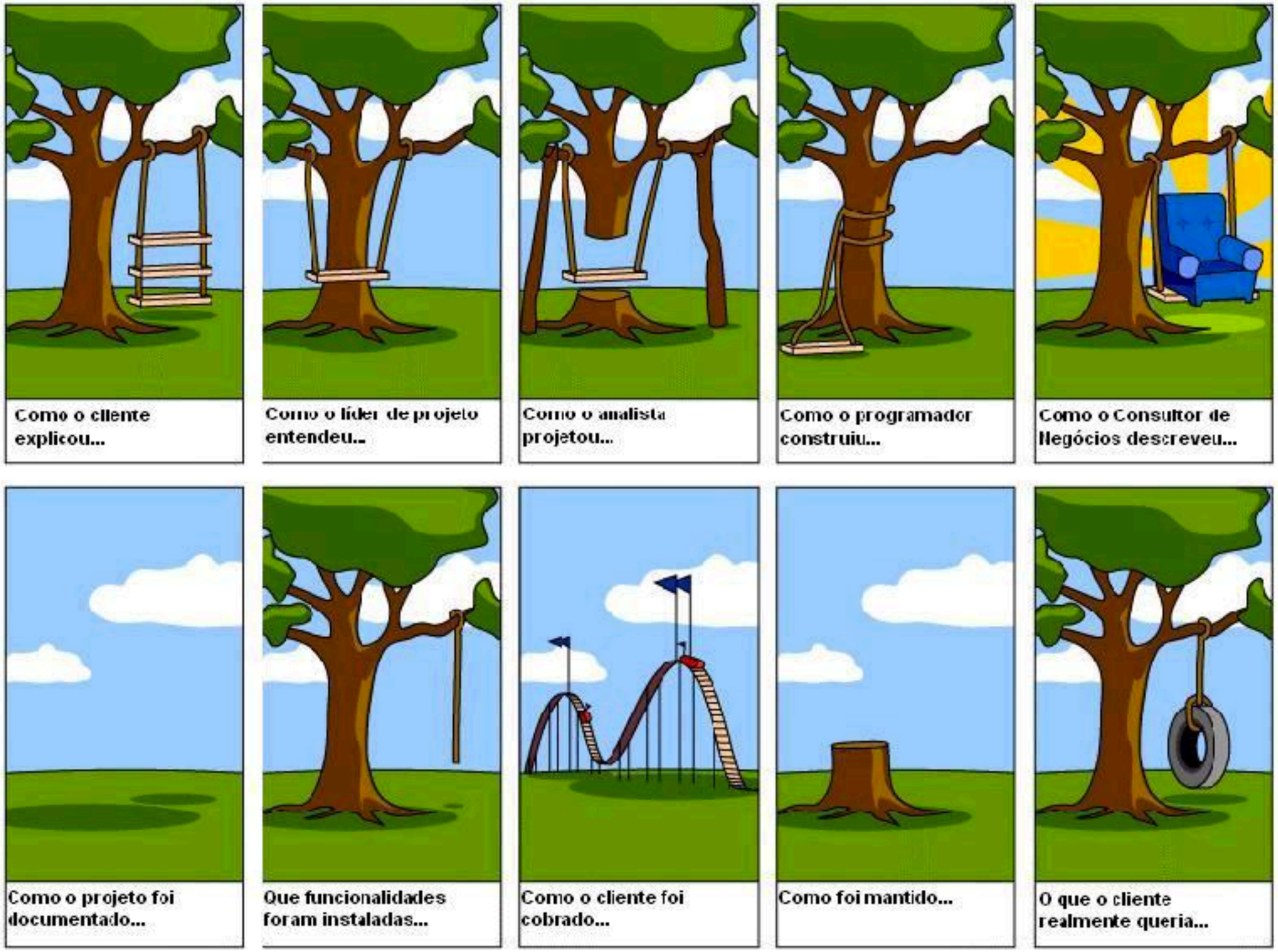
AULA 3
PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

O QUE VEREMOS NESTA AULA...

- Entender como se passa o processo de desenvolvimento de *software*
- Definir um processo iterativo e ágil
- Definir conceitos fundamentais do Processo Unificado

> Você deve utilizar o desenvolvimento iterativo apenas em projetos que você deseja que sejam bem sucedidos.
>
> **– MARTIN FOWLER**

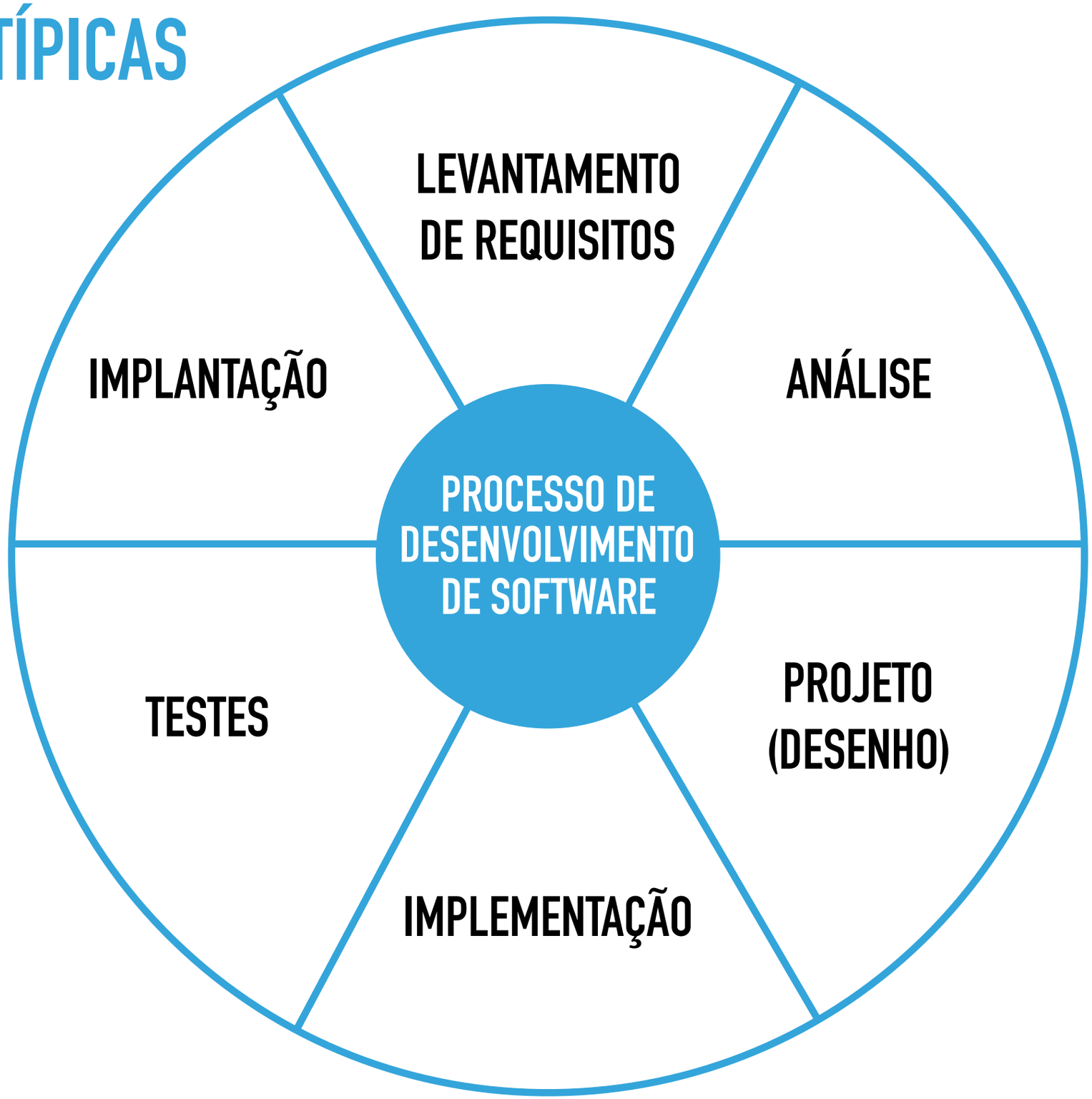
PROCESSO DE DESENVOLVIMENTO DE SOFTWARE



PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

- Compreende as atividades necessárias para definir, desenvolver, testar e manter um produto de *software*
- **Objetivos:**
  - Definir **quais** as atividades a serem executadas ao longo do projeto
  - **Quando**, **como** e por **quem** estas atividades serão executadas
  - Promover **pontos de controle** para verificar o andamento do desenvolvimento
  - **Padronizar** o modelo de desenvolvimento de *softwares* de um empresa
- Não é uma tarefa simples
- Exemplos: Processo Unificado (RUP), *Extreme Programming* (XP)

ATIVIDADES TÍPICAS



LEVANTAMENTO DE REQUISITOS

- Compreensão do problema
- Definir as necessidades dos usuários do sistema
- Feito pelos desenvolvedores juntamente com o cliente
- Gera um **documento de requisitos**
  - Ponto de referência
  - Deve ser entendido por leitores técnicos e não técnicos
  - Para que ele serve para cada leitor?
  - Não descreve como resolver o problema

COMO PODE SER FEITO

Leitura de obras de referência

Observação do ambiente do usuário

Entrevistas com o usuário

Entrevistas com especialistas

Reutilização de análises anteriores

Comparação com sistemas preexistentes

LEVANTAMENTO DE REQUISITOS

- Volatilidade dos requisitos
  - “Congelar" os requisitos
  - Adaptar a mudanças
    - Seu impacto deve ser analisado cuidadosamente
      - Cronograma e recursos financeiros
- Compreender o sistema o máximo possível antes de começas a construí-lo
- Requisitos devem ser ordenados por prioridade (**valor** e **custo**)

LEVANTAMENTO DE REQUISITOS

- **Requisitos funcionais**
  - Definem funcionalidades
- **Requisitos não funcionais**
  - Declaram características de qualidade do sistema
- **Requisitos normativos**
  - Declaram restrições impostas sobre o desenvolvimento do sistema

CLASSIFIQUE

Tempo médio entre falhas

Limitações sobre a interface com o usuário

O sistema deve permitir que cada professor lance as notas da turma

Requisitos sobre facilidade de uso

Tempo de resposta esperado pelo sistema

A plataforma tecnológica

Adequação a custos e prazos

O aluno pode realizar matrícula

LEVANTAMENTO DE REQUISITOS

- **Requisitos funcionais**
  - Definem funcionalidades
- **Requisitos não funcionais**
  - Declaram características de qualidade do sistema
- **Requisitos normativos**
  - Declaram restrições impostas sobre o desenvolvimento do sistema

CLASSIFIQUE

(2) Tempo médio entre falhas

(3) Limitações sobre a interface com o usuário

(1) O sistema deve permitir que cada professor lance as notas da turma

(2) Requisitos sobre facilidade de uso

(2) Tempo de resposta esperado pelo sistema

(3) A plataforma tecnológica

(3) Adequação a custos e prazos

(1) O aluno pode realizar matrícula

ANÁLISE

- **Engenharia de requisitos:** Levantamento de requisitos + Análise de requisitos
- Não leva em conta o ambiente tecnológico
- Criação de modelos que descrevem o sistema
- **Paralisia de análise:** estagnação da fase de análise
  - Normalmente o que acontece é o contrário; esta fase é ignorada

ANÁLISE DE DOMÍNIO

Identificar objetos do mundo real que serão processados pela aplicação

Deve-se identificar as regras de negócio e os processos de negócio

Exemplo: aluno, sala de aula, professor, turma, etc.

ANÁLISE DE APLICAÇÃO

Identificar componentes do sistema necessários para suprir as funcionalidades

Exemplo: botão para realizar a matrícula

ANÁLISE

- **Validação:** será que o *software* certo está sendo construído?
- **Diagramas UML + Protótipos**
- Protótipos: qualquer aspecto que precise ser mais bem entendido é alvo de prototipagem
- Mais fácil visualizar o sistema através de protótipos do que através de diagramas
- Incentiva a participação ativa do usuário na validação
- Protótipos não substituem a construção de modelos do sistema
  - Modelos guiam as demais fases do desenvolvimento do sistema
  - Erros identificados no protótipo devem ser utilizados para modificar e refinar os modelos
- Podem servir de base para o desenvolvimento ou serem descartados

PROJETO (DESENHO)

- Como o sistema funcionará para atender os requisitos, de acordo com os recursos tecnológicos existentes (**descrição computacional**)
- Adicionam-se **restrições de tecnologia** aos modelos da fase de análise
  - Arquitetura física do sistema, padrão de interface gráfica, algoritmos específicos, gerenciador de banco de dados
- **Verificação:** será que o *software* está sendo construído corretamente?
- Atividades de misturam com as atividades da fase de análise

PROJETO DA ARQUITETURA (PROJETO DE ALTO NÍVEL)

Identificar classes de objetos

PROJETO DETALHADO (PROJETO DE BAIXO NÍVEL)

Definir as colaborações entre os objetos

Projeto de interface com o usuário

Projeto de banco de dados

IMPLEMENTAÇÃO, TESTES E IMPLANTAÇÃO

IMPLEMENTAÇÃO

Codificação

IMPLANTAÇÃO

Sistema é distribuído e instalado no ambiente do usuário

Treinamento de usuario

Migração de dados

TESTES

Leva em consideração os dados obtidos nas fases de análise e projeto

Resulta em relatórios de testes

Testes de unidades

- Classes ou métodos
- Identificar logo erros
- Automatização

Testes de integração

- Detectar falhas na comunicação entre objetos

Testes de sistema

- Verificar se o sistema está construído de acordo com os requisitos levantados para ele

Testes de aceitação

- Realizados pelos usuários

PARTICIPANTES DO PROCESSO

GERENTE DE PROJETO

Gerenciar, coordenar e acompanhar as atividades necessárias

Fazer orçamento (tempo, custo, recursos)

Definir o processo de desenvolvimento

PROGRAMADORES

Responsáveis pela implementação

Confunde-se com analistas

ESPECIALISTAS DE DOMÍNIO

Possui conhecimento acerca da área em que o sistema estará inserido

Muitas vezes é o cliente usuário do sistema

PROJETISTAS

Avaliar as alternativas de solução do problema e gerar uma especificação da solução computacional

ANALISTAS

Responsável pelo levantamento e análise dos requisitos

Ponte de comunicação entre clientes e desenvolvedores

MODELOS DE CICLO DE VIDA – CASCATA

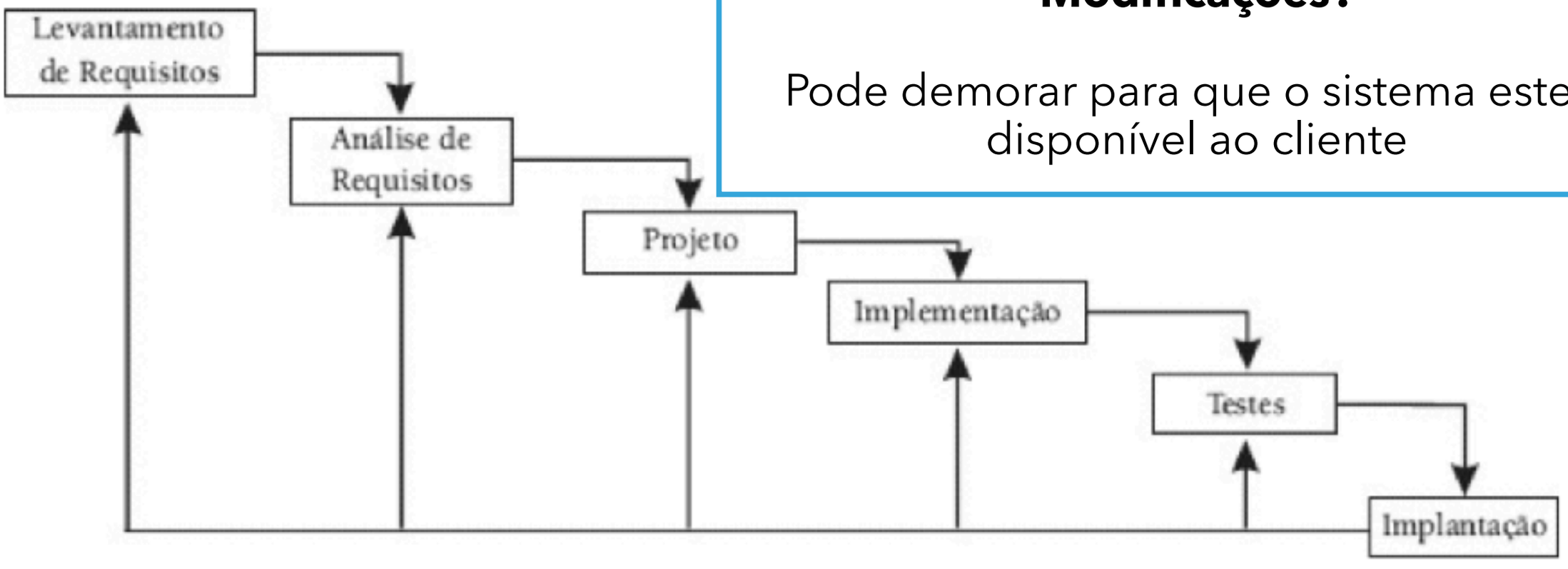
- Clássico ou linear
- Foi utilizado por muitos anos
- Abordagem sequencial

PROBLEMAS

Algumas atividades podem ser realizadas em paralelo

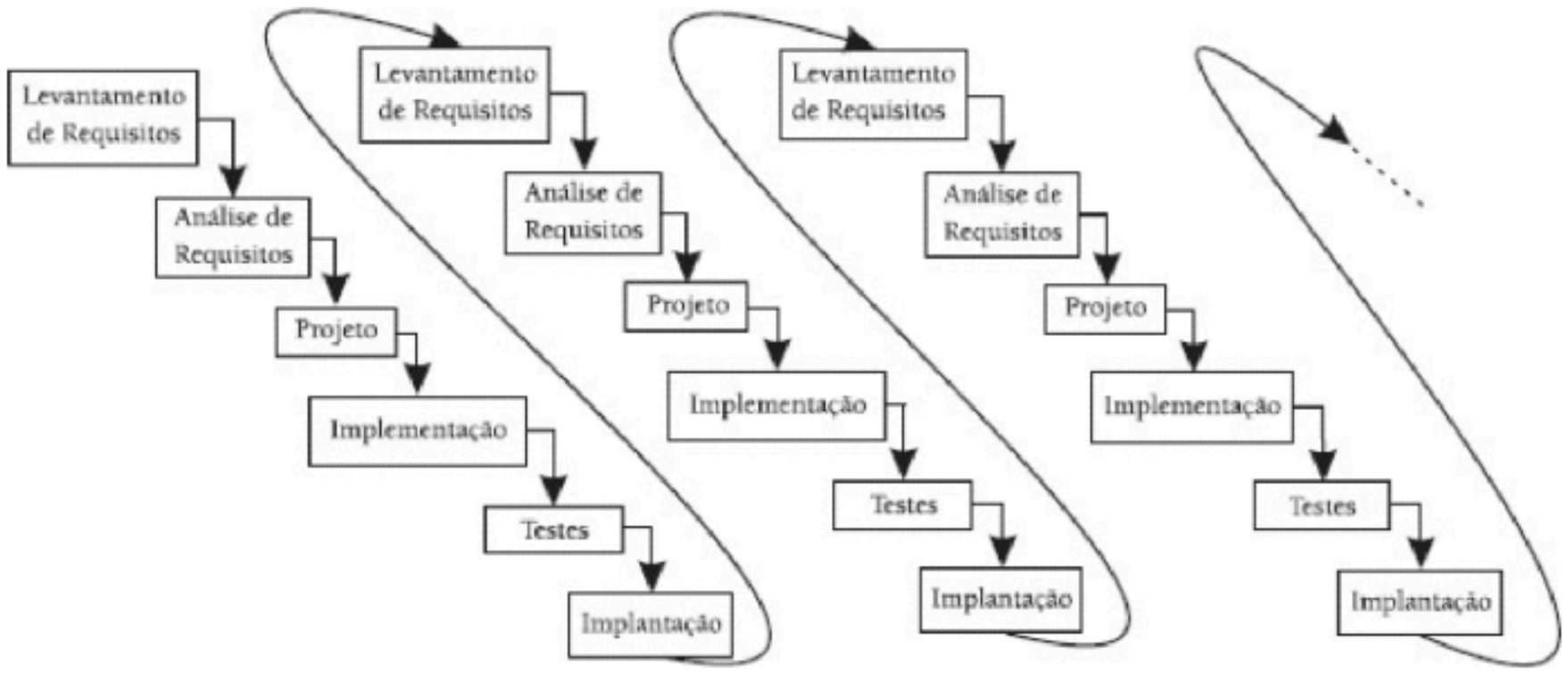
Deve-se declarar todos os requisitos no início do processo de desenvolvimento
Modificações?

Pode demorar para que o sistema esteja disponível ao cliente

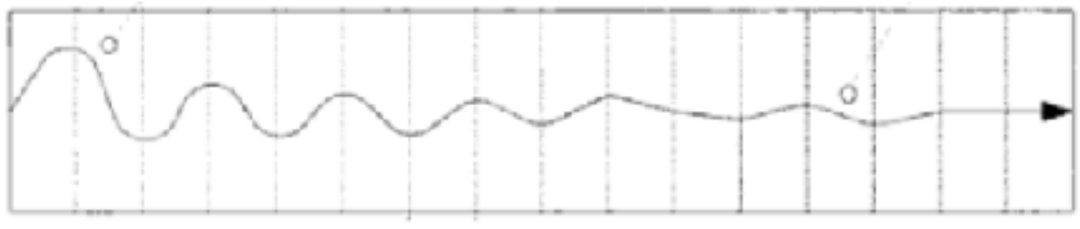


MODELOS DE CICLO DE VIDA – ITERATIVO E INCREMENTAL

- Divide o desenvolvimento em **ciclos**
- Cada ciclo produz um incremento
- Possuem melhores taxas de sucesso e produtividade
- **Iterativo:** vários passos similares
- **Incremental:** estende funcionalidade a cada passo
- Iterações curtas - prazos devem ser cumpridos



MODELOS DE CICLO DE VIDA – ITERATIVO E INCREMENTAL



- Os requisitos são definidos a cada ciclo
- Não estima a mudança descontrolada de requisitos
- Deve-se dividir os requisitos do sistema em partes (prioridade e risco)
- **Vantagem:** usuário participativo
  - Diminui a a probabilidade de requisitos mal interpretados
- **Vantagem:** melhor gerenciamento dos riscos
  - Requisitos mais arriscados são considerados o mais cedo possível
- **Problema:** cliente pode se empolgar com a primeira versão
- **Problema:** o gerenciamento é mais complexo

MODELOS DE CICLO DE VIDA – ITERATIVO E INCREMENTAL

- **Benefícios:**
  - Menos erros no projeto, maior produtividade e menor taxa de defeitos
  - Mitigação precoce de altos riscos
  - Progresso visível desde o início
  - A equipe não é sobrecarregada pela “paralisia da análise”
  - O aprendizado obtido em uma iteração pode ser utilizado para melhorar o próprio processo de desenvolvimento

PU – PROCESSO UNIFICADO

- Processo iterativo bastante popular
  - RUP (*Rational Unified Process*)
- Pode utilizar técnicas de outros processos iterativos (misto de técnicas)

EXEMPLO – ITERAÇÃO

Duração: 3 semanas

1º dia:
1h para definir as metas da iteração
Engenharia reversa p/ criar diagramas UML da última iteração
Resto do dia: reunião com a equipe (quadros-brancos, trabalhando em pares, rascunhos de diagramas UML, escrevendo pseudocódigos e notas de projeto)

Outros dias: implementação, testes, integração

EXEMPLO – ANÁLISE E PROJETO PU

Antes da iteração 1
Reunião
Identificar nomes de casos de uso e características
Escolher 10% da lista (casos de uso com alto valor de negócio, alto risco ou arquiteturalmente significativos)

Primeiros 2 dias
Especificar detalhadamente estes 10% dos casos de uso
Escolher o que fará parte da 1ª iteração (não será tudo)

1ª iteração (3 ou 4 semanas)
Modelagem
Dev.
Testes

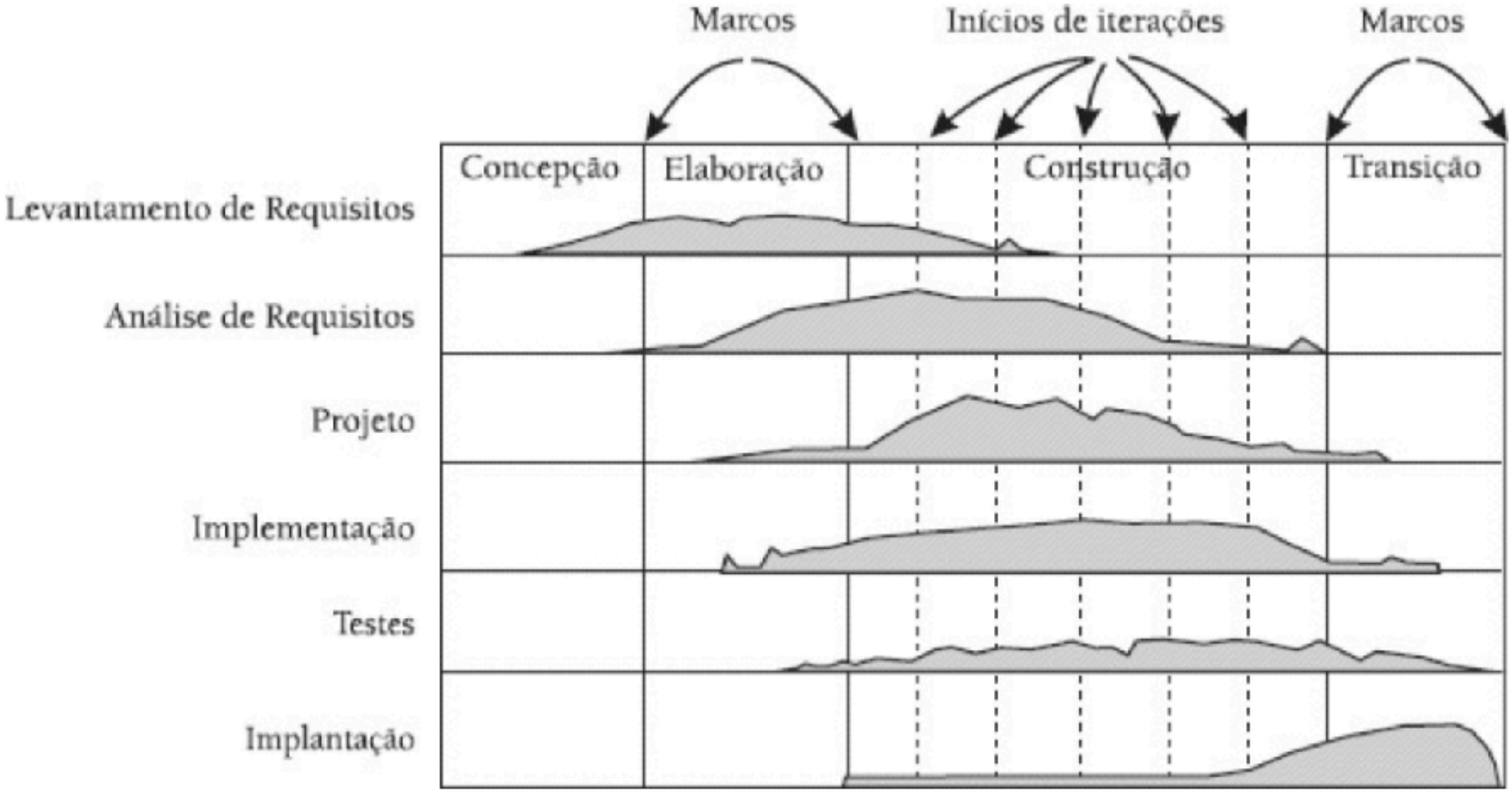
Faltando 1 semana para acabar a 1ª iteração
As metas podem ser alcançadas?
Se não, diminuir o escopo da iteração

Fim da iteração
Mostrar o resultado aos interessados

Antes da iteração 2
Reunião para descrever mais 10% dos casos de uso
Escolher o que fará parte da 2ª iteração

MODELOS DE CICLO DE VIDA – ITERATIVO E INCREMENTAL

FASES E FLUXOS DE TRABALHO



MODELOS DE CICLO DE VIDA – ITERATIVO E INCREMENTAL

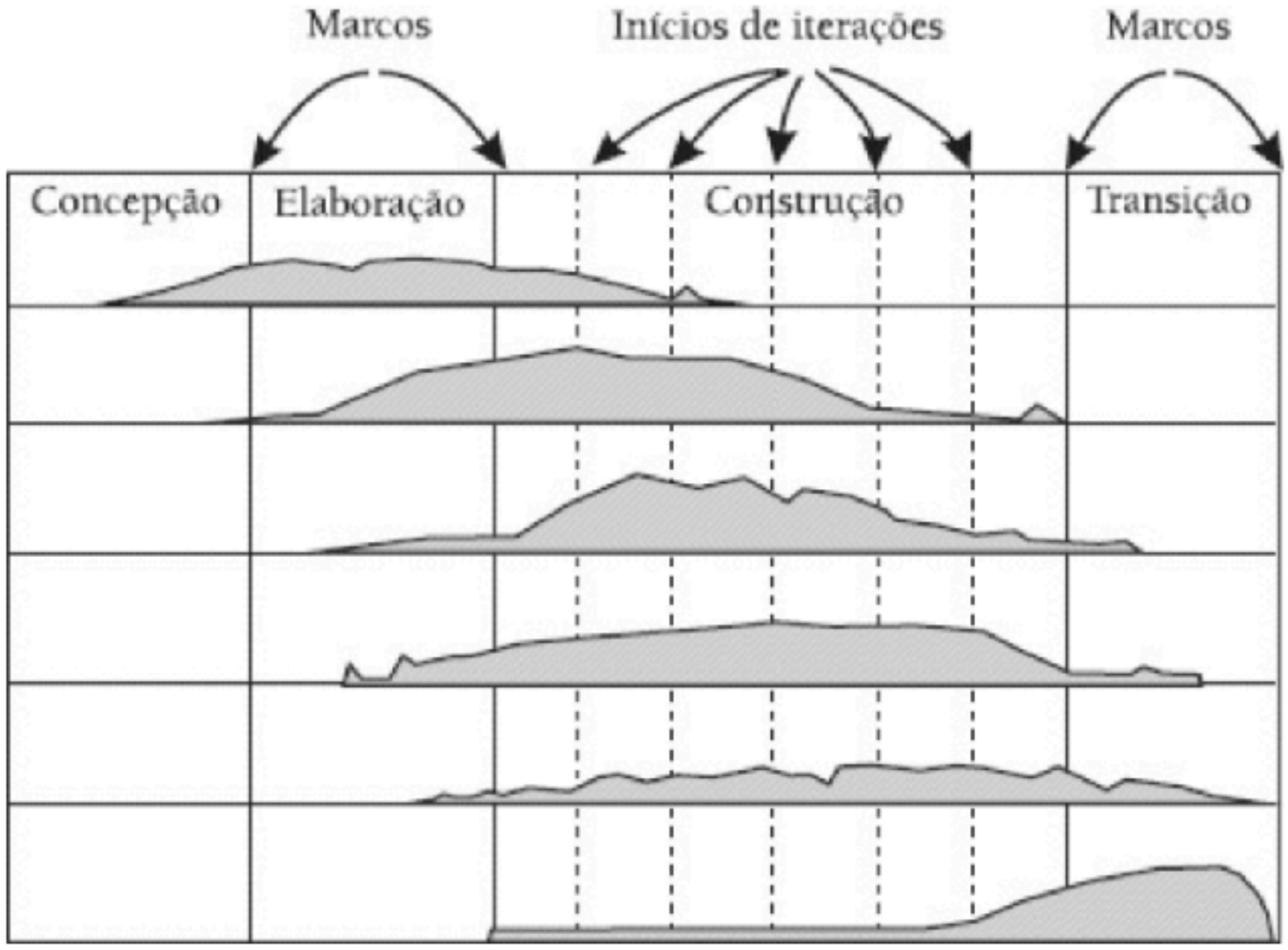
FASES

Separadas em iterações

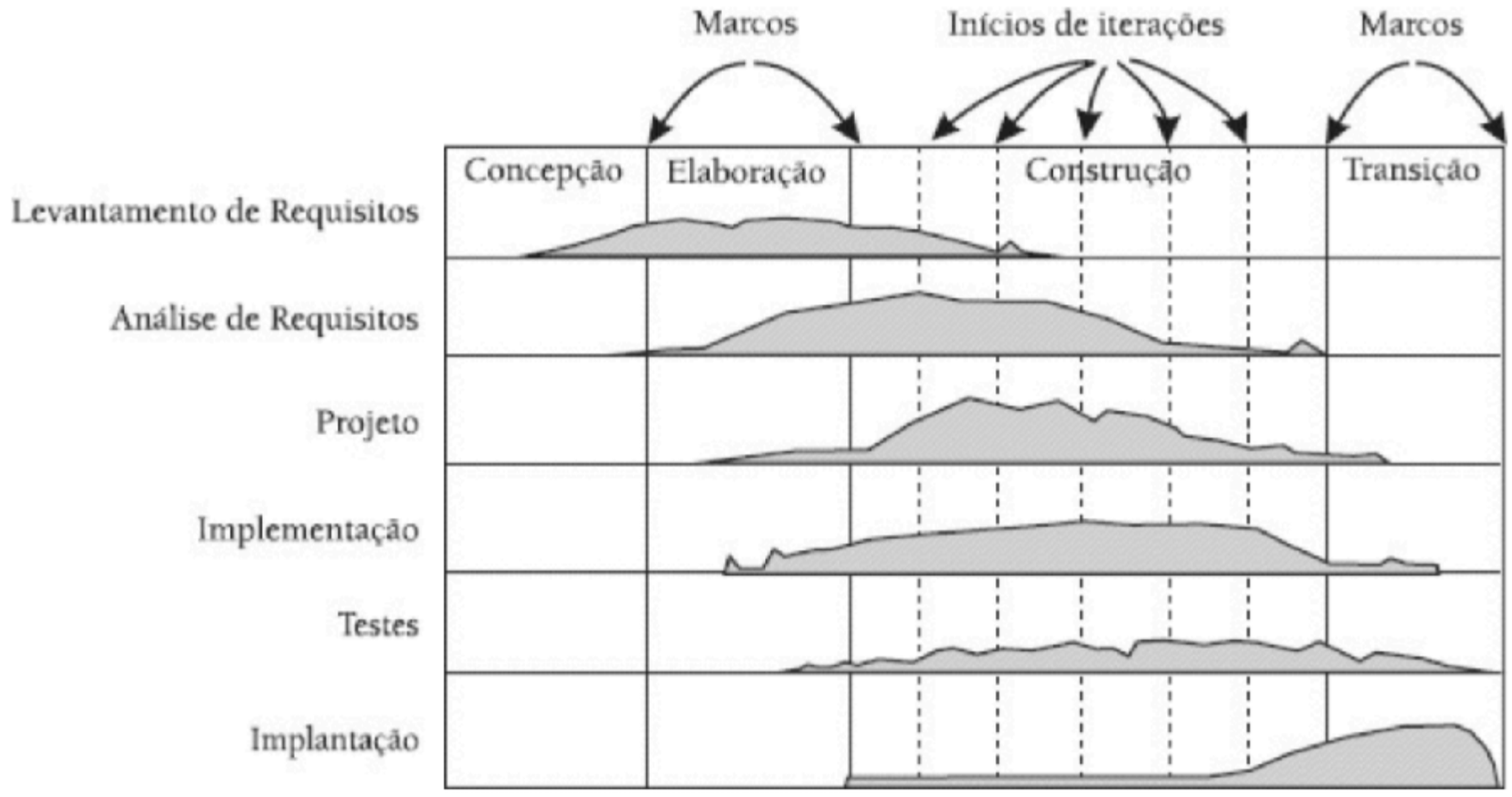
Entregam incrementos (liberados ao usuário ou internos)

Pode englobar várias atividades

Marcos: fim de uma fase
Úteis para estimar gastos e acompanhar cronograma



MODELOS DE CICLO DE VIDA – ITERATIVO E INCREMENTAL

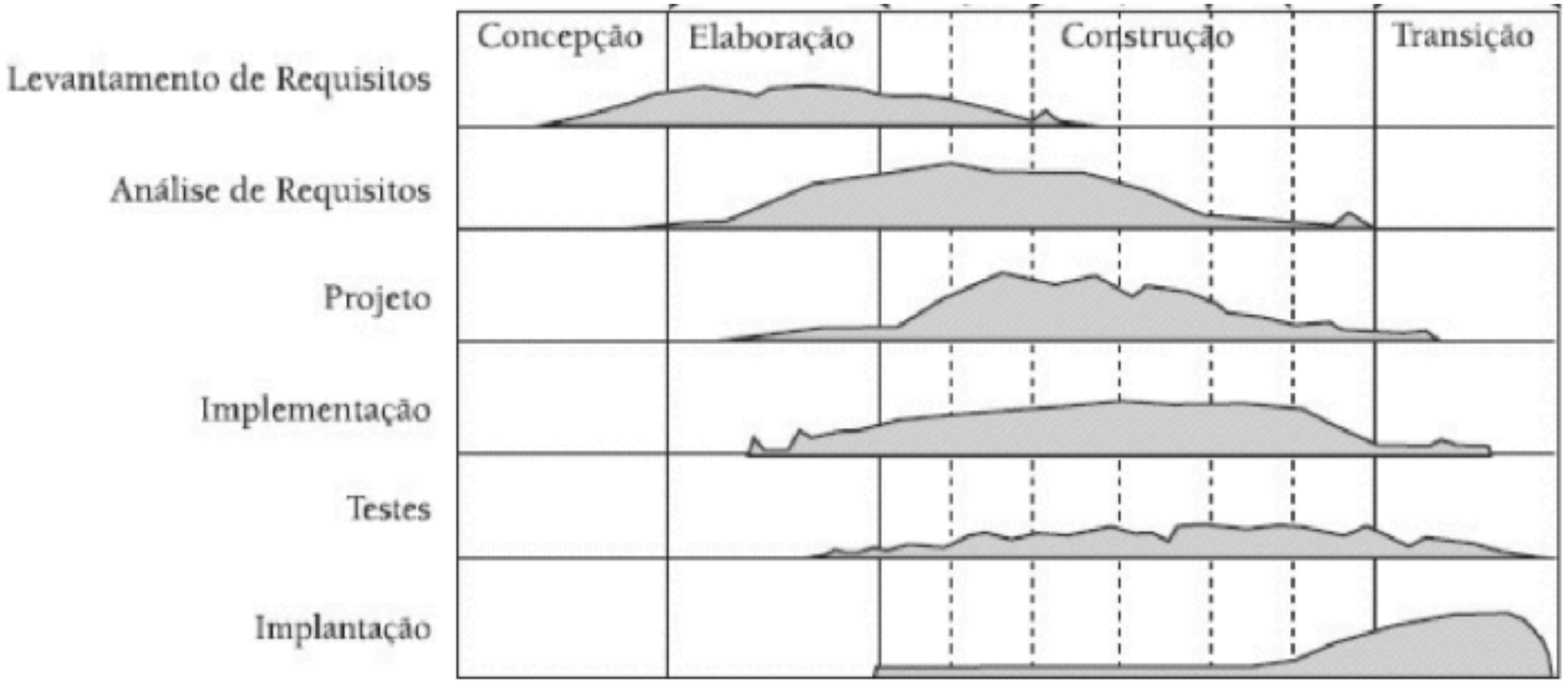


FLUXOS DE TRABALHO

Em uma iteração, o fluxo de trabalho é composto por: requisitos, análise, projeto, implementação, teste e implantação

Cada parte do fluxo de trabalho será usada em maior ou menor grau a depender da fase

MODELOS DE CICLO DE VIDA – ITERATIVO E INCREMENTAL



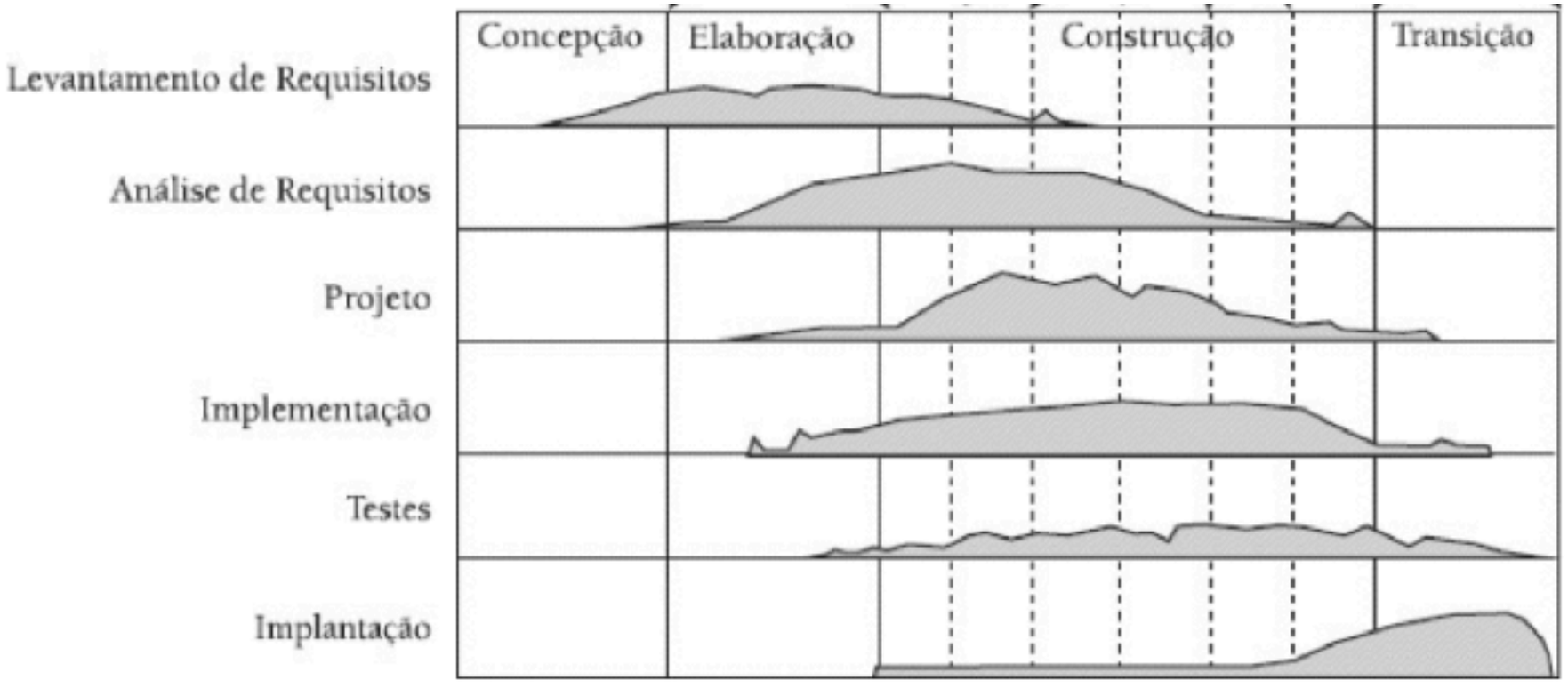
CONCEPÇÃO

- Definir a ideia geral e o escopo
- É feito um planejamento de alto nível do sistema
- Estudo de viabilidade
- Identificar maiores riscos e prazos

ELABORAÇÃO

- O planejamento inicial é completado (detalhado)
- Os requisitos são ordenados
- São planejadas as iterações da fase de **construção**

MODELOS DE CICLO DE VIDA – ITERATIVO E INCREMENTAL



CONSTRUÇÃO

- Desenvolvimento
- Aumentam as atividades de análise e projeto
- Maior número de iterações

TRANSIÇÃO

- Usuários são treinados (caso seja necessário)
- Trata-se questões de testes finais, instalação e configuração
- Avalia aceitação do cliente

MÉTODOS ÁGEIS

- Aplicam desenvolvimento iterativo com valores e práticas que encorajam **agilidade**

MANIFESTO ÁGIL

Indivíduos e iterações vem antes de processos e ferramentas

Software funcionando vem antes de documentação abrangente

Colaboração do cliente vem antes de negociação de contrato

Resposta à modificação vem antes de um plano em andamento

MODELAGEM ÁGIL

- "A finalidade de modelagem (rascunhos em UML, ...) é principalmente **entender**, não documentar" (BEZERRA, 2015, pg. 57)
- Método ágil não evita a modelagem
- Modele e aplica UML às partes incomuns, difíceis e cheias de armadilhas do projeto
- Modele utilizando quadro-branco
- Modele aos pares
- Use notações simples

BIBLIOGRAFIA

- LARMAN, Craig. **Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos e ao desenvolvimeto interativo.** 3.ed. Porto Alegre: Bookman, 2007. xiv, 695 p. [cap. 1, 2 e 3]

- BEZERRA, Eduardo. **Princípios de análise e projeto de sistemas com UML**. 3. ed. rev atual. Rio de Janeiro, RJ: Elsevier, 2015. 393 p. [cap. 1 e 2]