



# Engenharia de Software

Tecnologia de Software

## **Aula 2:** Processos de software

Fábio Levy Siqueira  
[levy.siqueira@usp.br](mailto:levy.siqueira@usp.br)

# Processo

- Definição (ISO, 2017)

Conjunto de atividades inter-relacionadas ou interagindo que transforma entradas em saídas

- Algumas características

- O processo deve ter um objetivo
- Importância do valor da saída para a organização
- Um processo tem um cliente
- Pode cruzar fronteiras organizacionais

# Processo

- Quais são os elementos de um processo?
  - Como descrevê-lo?

# Questões

- Todo projeto de software executa um ou mais *processos*?
  - Mesmo um projeto com 1 pessoa?
  
- Pensando nos processos *técnicos*
  - (Especialmente os de *desenvolvimento*)
  - Existe um processo ideal?
  - Os processos serão *sempre iguais* em uma empresa?
    - O que faz o processo ser diferente?

# Influências

- Algumas influências aos processos
  - Tamanho do projeto
  - Complexidade do software
  - Experiência da equipe
  - Uso de novas tecnologias
  - Tipo de projeto
  - Distribuição da equipe
  - *Time to market*
  - Contrato / questões legais

# Definição dos processos

- Quem decide como será o processo?
- Algumas variáveis a considerar
  - **Cultura organizacional**
  - **Modelo de ciclo de vida:** iterativo, incremental, cascata...
  - **Processos:** RUP, XP, UP...
  - **Métodos:** métodos de elicitação de requisitos, test-first, Kanban, SCRUM...
  - **Linguagens de programação:** Visual Basic .Net, Cobol, C, C++, C#, Java, Ruby...
  - **Tecnologias:** ferramentas, máquinas, bibliotecas...

# Definição dos processos

- Qual é o problema de se usar um processo inadequado?
  - Entregas fora do prazo, custo, escopo (funcionalidade) e qualidade
  - Excesso de burocracia
  - Dificuldade de manutenção
  - Insatisfação dos desenvolvedores

# **Sistema e software**



# Desenvolvimento de software

- Desenvolvimento de **software** ou de **sistemas**?
  - Sistema == Software?

- Definição de sistema

Combinação de elementos que interagem, organizados para atingir um ou mais propósitos estabelecidos (ISO, 2017).

- E o software?
  - Existe software fora de um sistema?

# Desenvolvimento de software

- Software é apenas um elemento de um sistema
  - Software não existe sem um hardware!
  - Elementos de um sistema
    - **Software**
    - Hardware
    - Dados
    - Pessoas
    - Processos
    - Procedimentos
    - Instalações
    - Serviços
    - Materiais
    - Entidades naturais (água, organismos e minerais)

# Sistema e o software

- *Exemplo*
  - Sistema de aluguel de bicicletas

# Sistema e o software

- O desenvolvimento de software é *um dos processos* do desenvolvimento de um sistema
  - Existem **sistemas de software**, em que o software é a parte *mais importante* do sistema
- Em geral o processo de desenvolvimento de sistemas é *recursivo*
  - Até mesmo o processo de desenvolvimento de software!

# Processos de software

# Processos de software

- Em uma empresa que trabalha com software
  - O desenvolvimento é o único processo?
    - E processos organizacionais?
    - E processos de apoio?
  - Quais são esses outros processos?

# ISO 12207

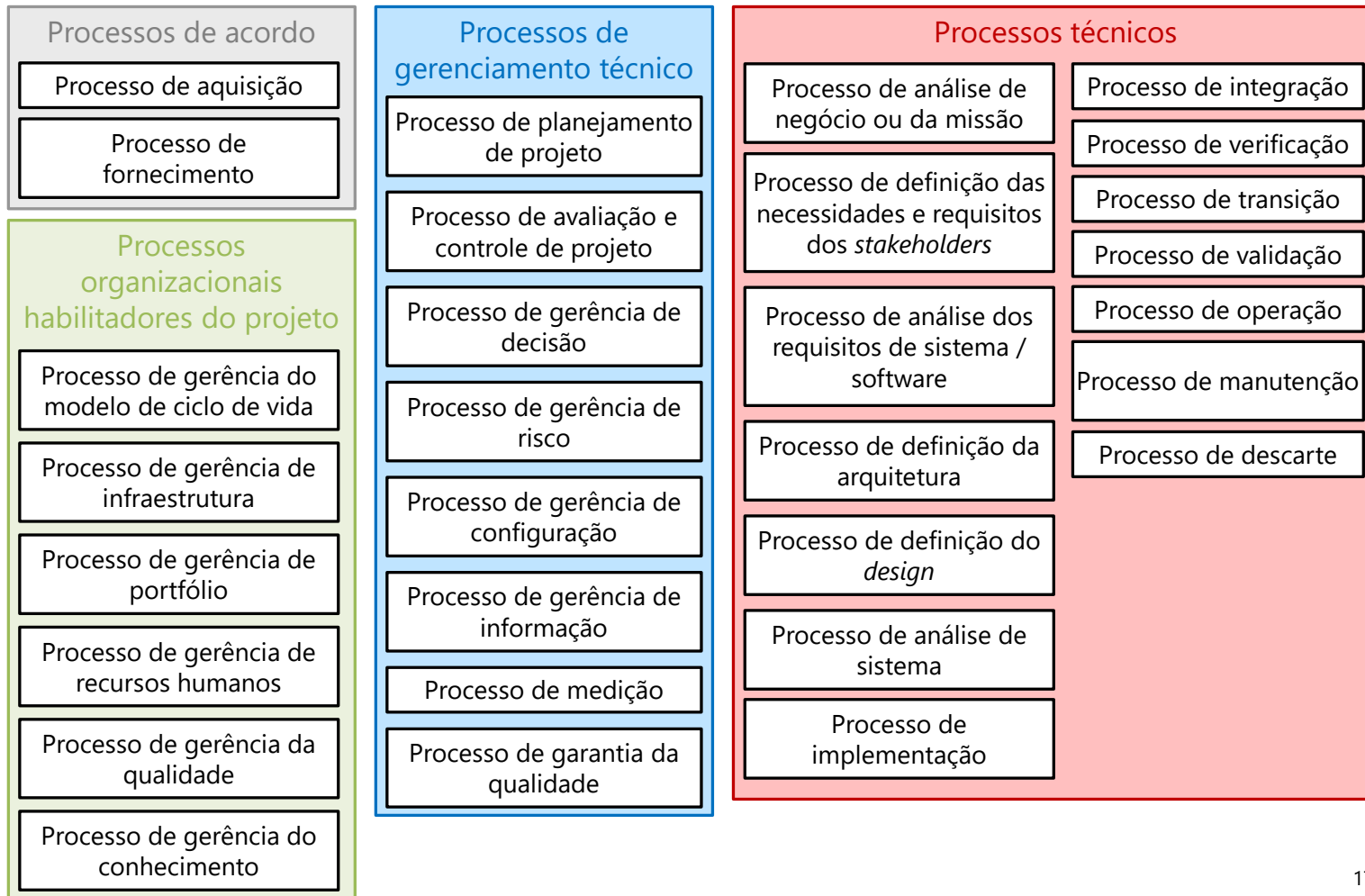
- Define um quadro de referência para os processos de software
  - Terminologia básica: **comunicação**
  - Pode ser usado para definir processos necessários
    - Adquirentes, fornecedores e outros *stakeholders*
    - Base para a melhoria de processos
- Especifica para cada processo
  - Propósito
  - Resultados
  - Atividades e tarefas

# ISO 12207

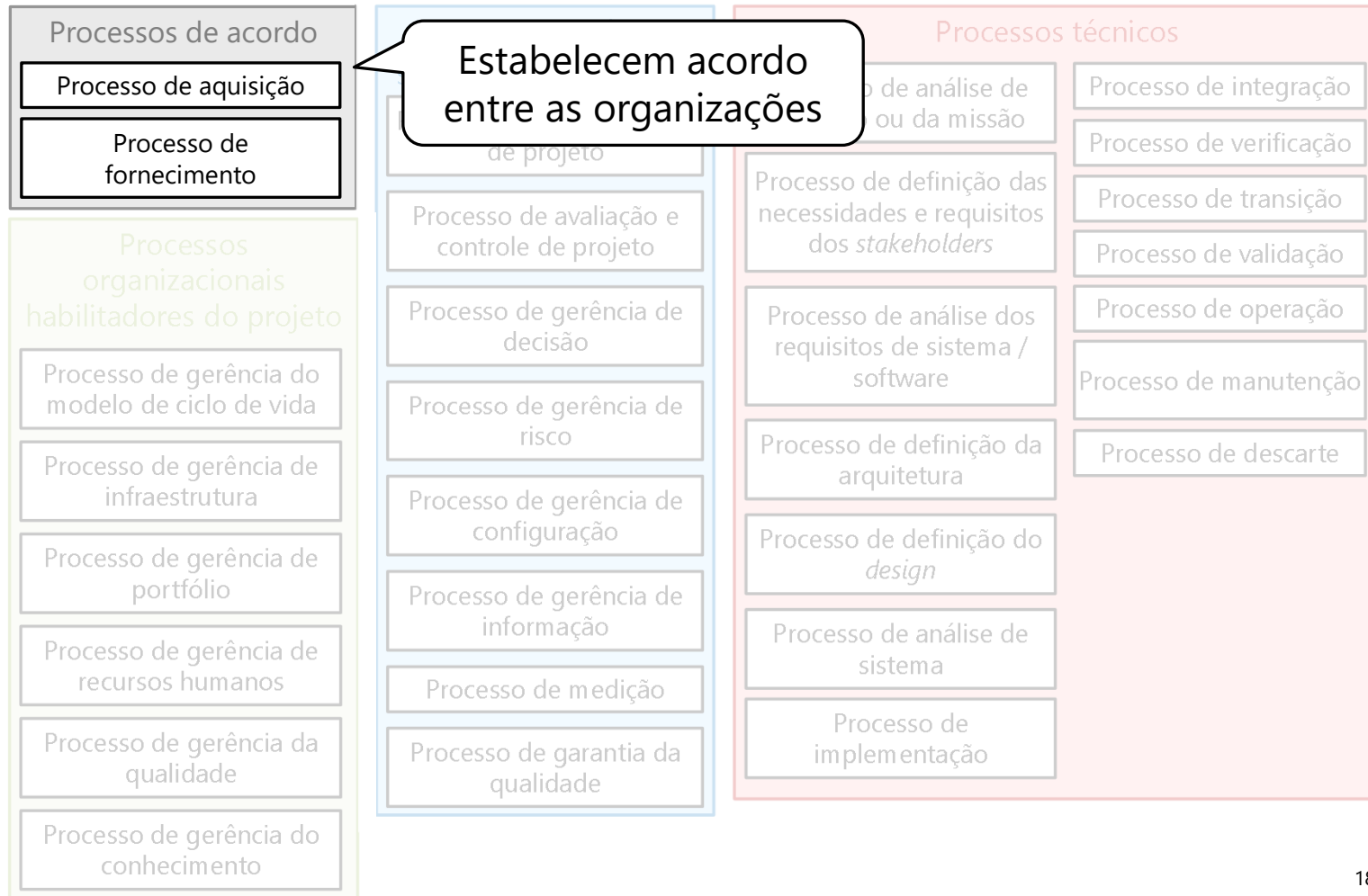
- Não especifica
  - Modelos de ciclo de vida, métodos, ferramentas, técnicas ou documentos
- Necessário instanciar os processos na organização
  - Adição e remoção de atividades e tarefas
  - Alteração da ordem definida
  - Análise da necessidade dos processos



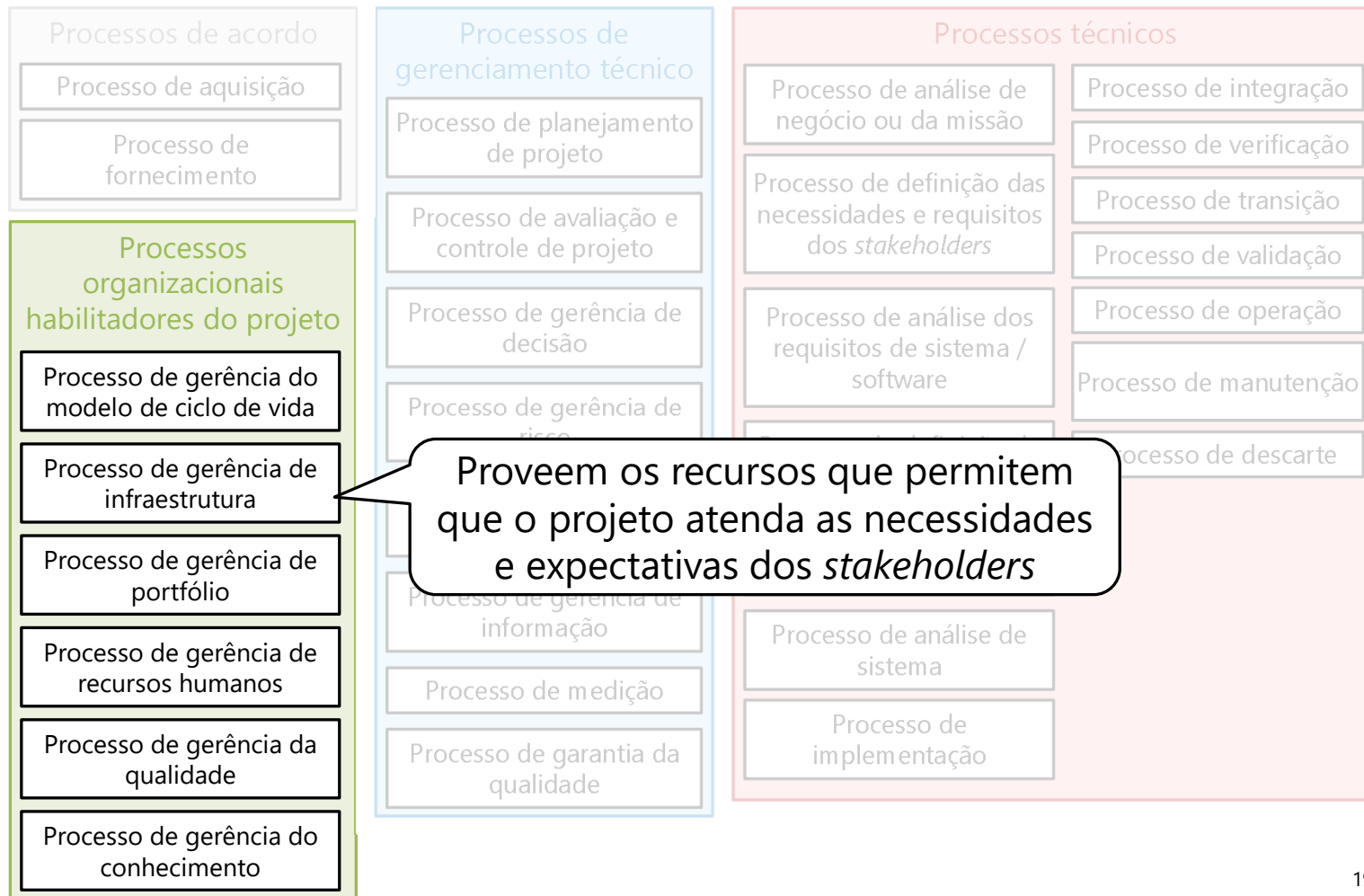
# Processos



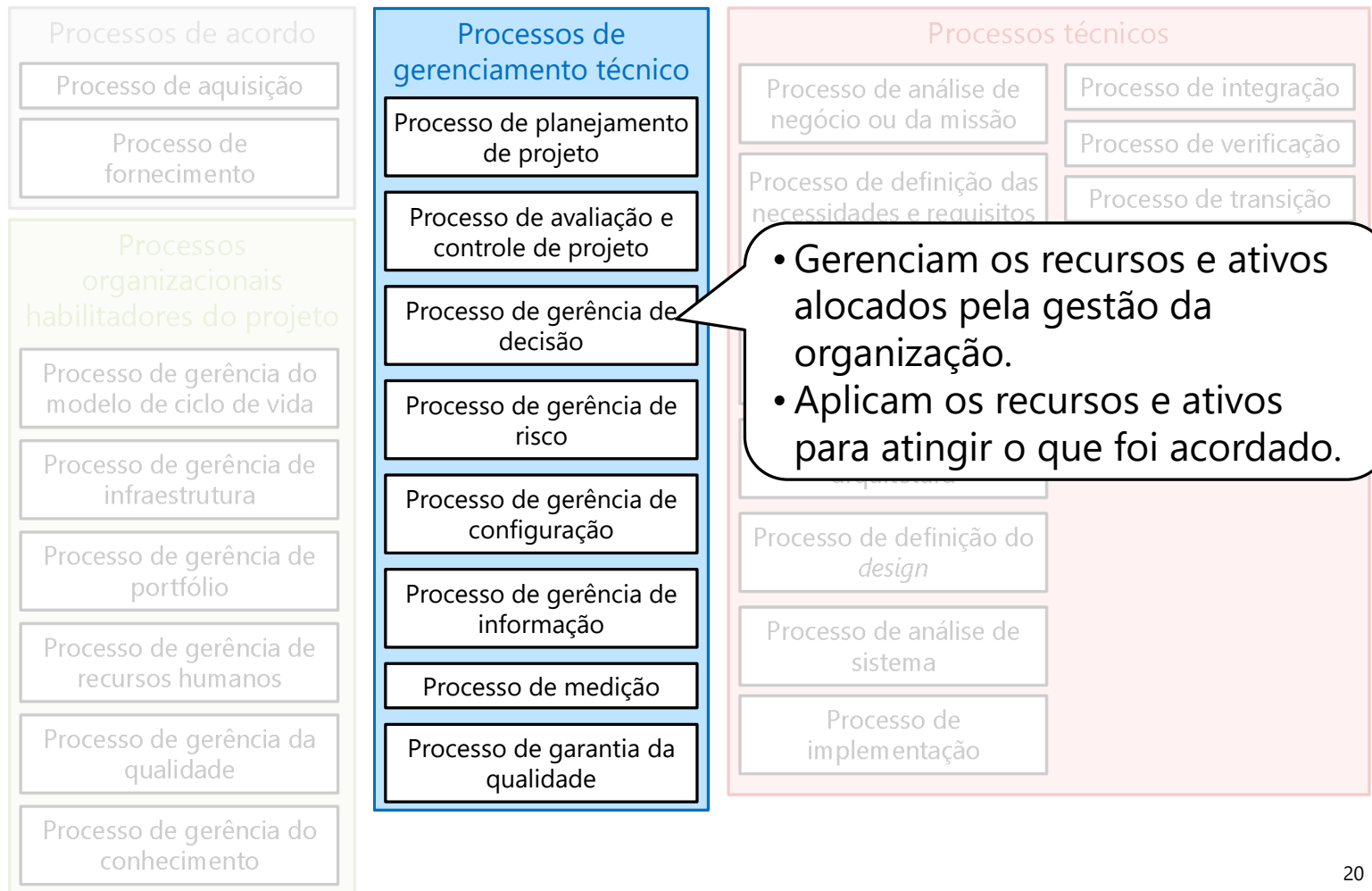
# Processos



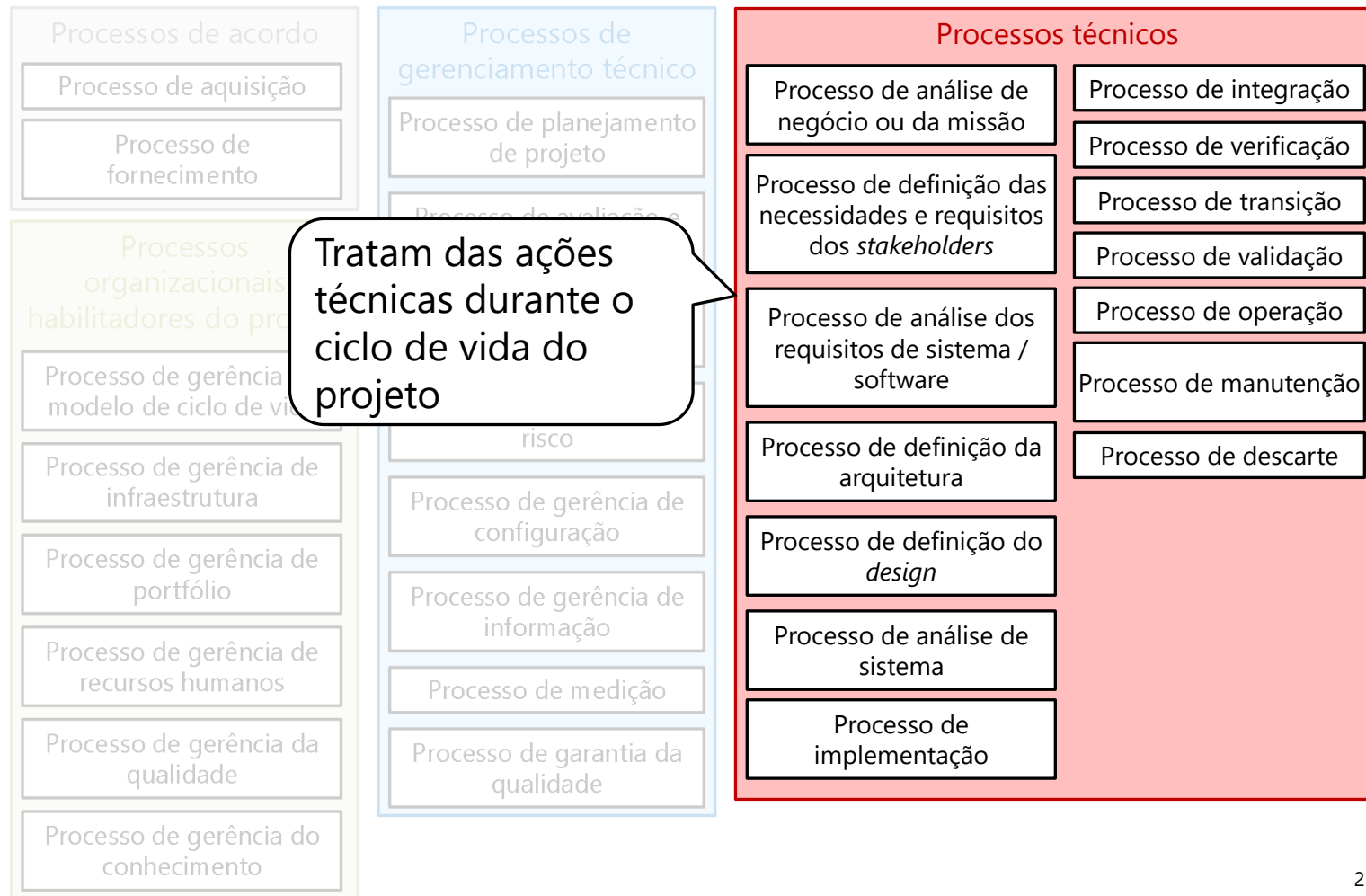
# Processos



# Processos



# Processos



# Processos de “desenvolvimento”

- Atividades “clássicas”
  - Não há um consenso nos termos

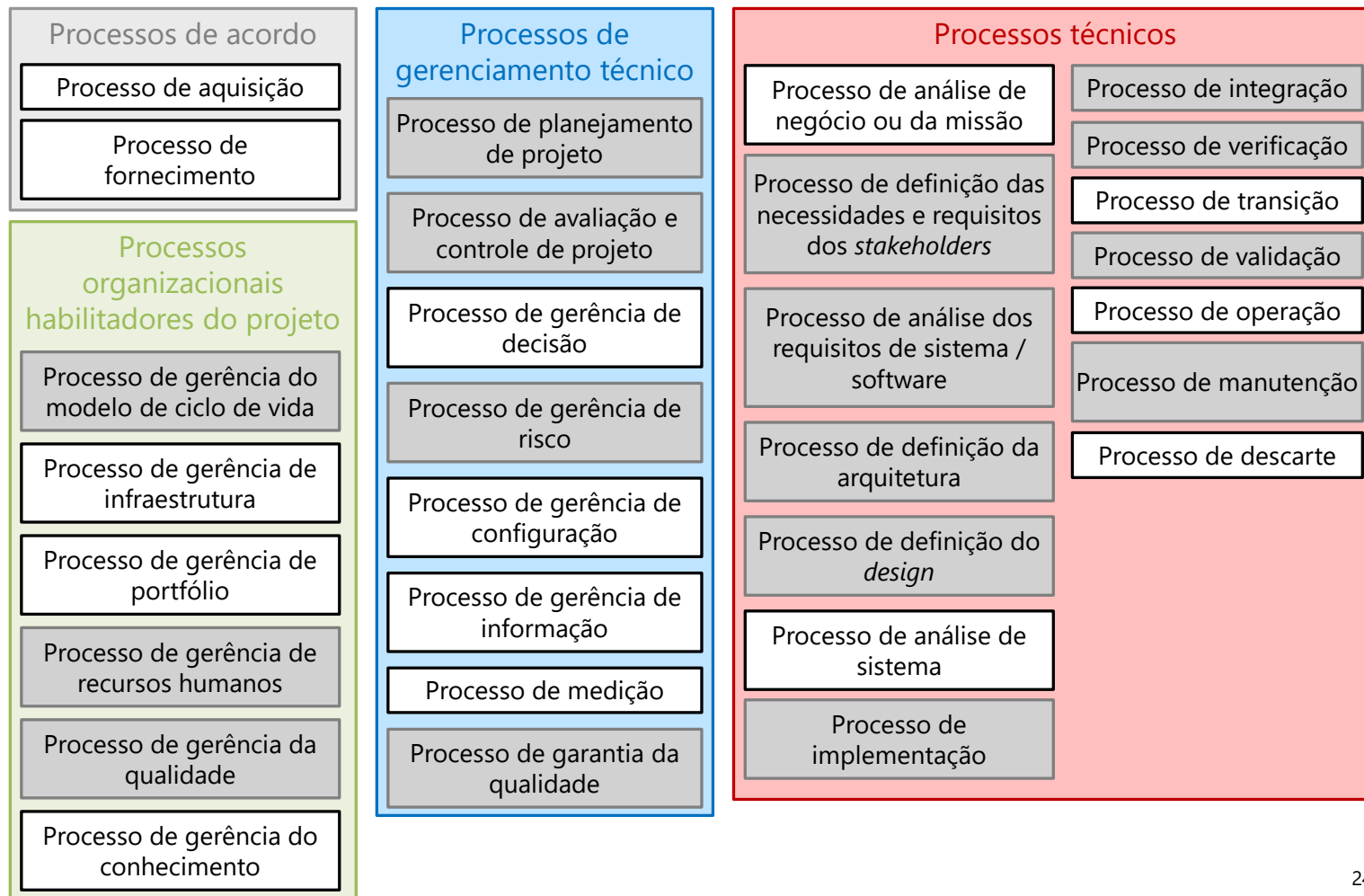
Definição e Análise de Requisitos	<ul style="list-style-type: none"><li>• Levantar os requisitos do software</li><li>• Refinar e estruturar os requisitos</li></ul>
Projeto	<ul style="list-style-type: none"><li>• Projetar como o software deve fazer o que foi requisitado</li></ul>
Implementação	<ul style="list-style-type: none"><li>• Criar o software</li><li>• Criar e executar os testes unidade</li></ul>
Teste	<ul style="list-style-type: none"><li>• Executar o software criado em busca de defeitos</li></ul>
Implantação	<ul style="list-style-type: none"><li>• Colocar o software no ambiente real</li></ul>

# Processos de “desenvolvimento”

## ▪ Atividades “clássicas” X Processos técnicos

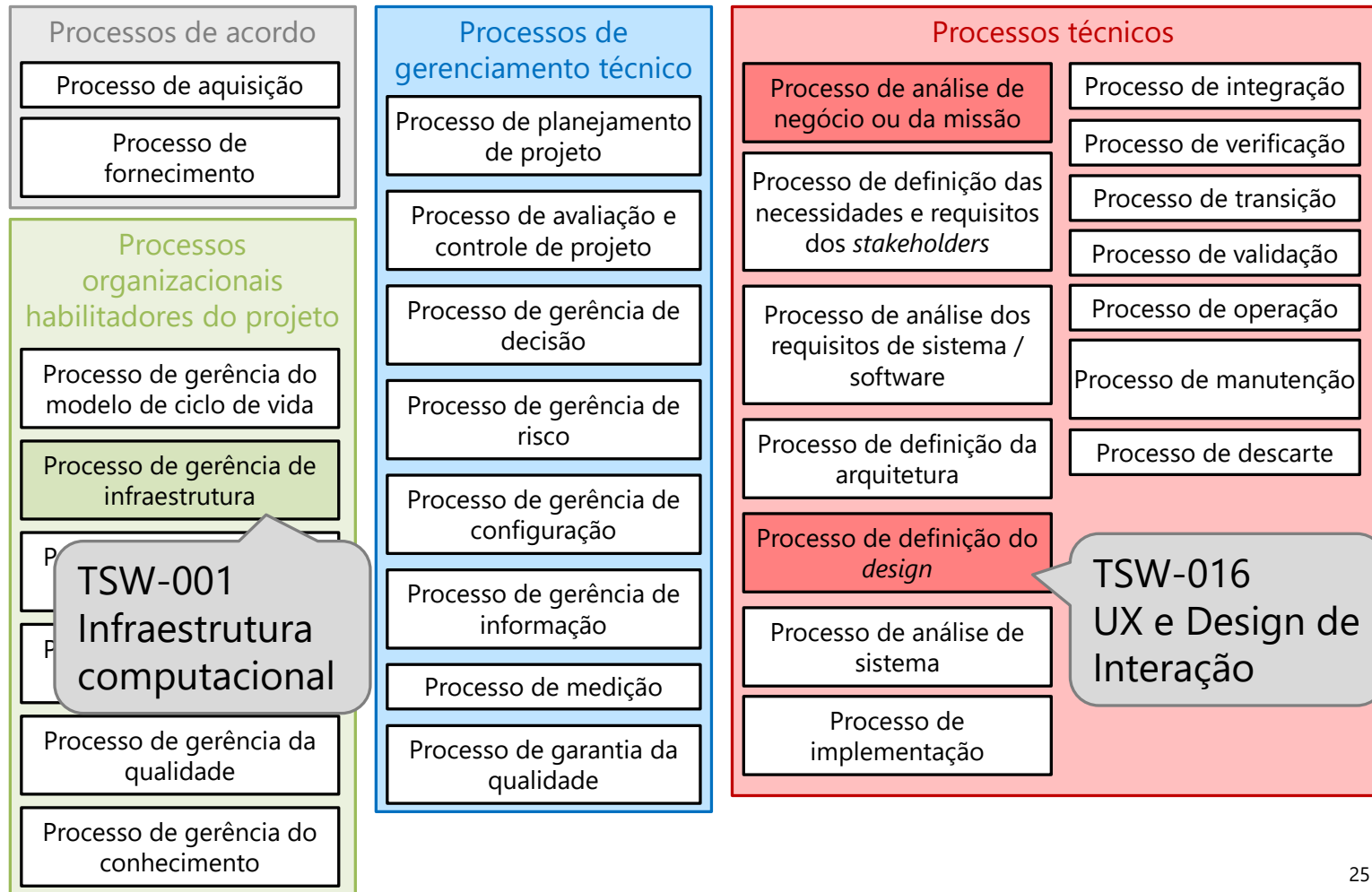
Definição e Análise de Requisitos	<ul style="list-style-type: none"><li>• Processo de análise de negócio ou da missão</li><li>• Processo de definição das necessidades e requisitos dos stakeholders</li><li>• Processo de definição dos requisitos do sistema/software</li></ul>
Projeto	<ul style="list-style-type: none"><li>• Processo de definição da arquitetura</li><li>• Processo de definição do <i>design</i></li></ul>
Implementação	<ul style="list-style-type: none"><li>• Processo de implementação</li><li>• Processo de integração (<b>também teste</b>)</li></ul>
Teste	<ul style="list-style-type: none"><li>• Processo de verificação (<b>parte dele</b>)</li><li>• Processo de validação (<b>parte dele</b>)</li></ul>
Implantação	<ul style="list-style-type: none"><li>• Processo de transição</li></ul>

# Aulas

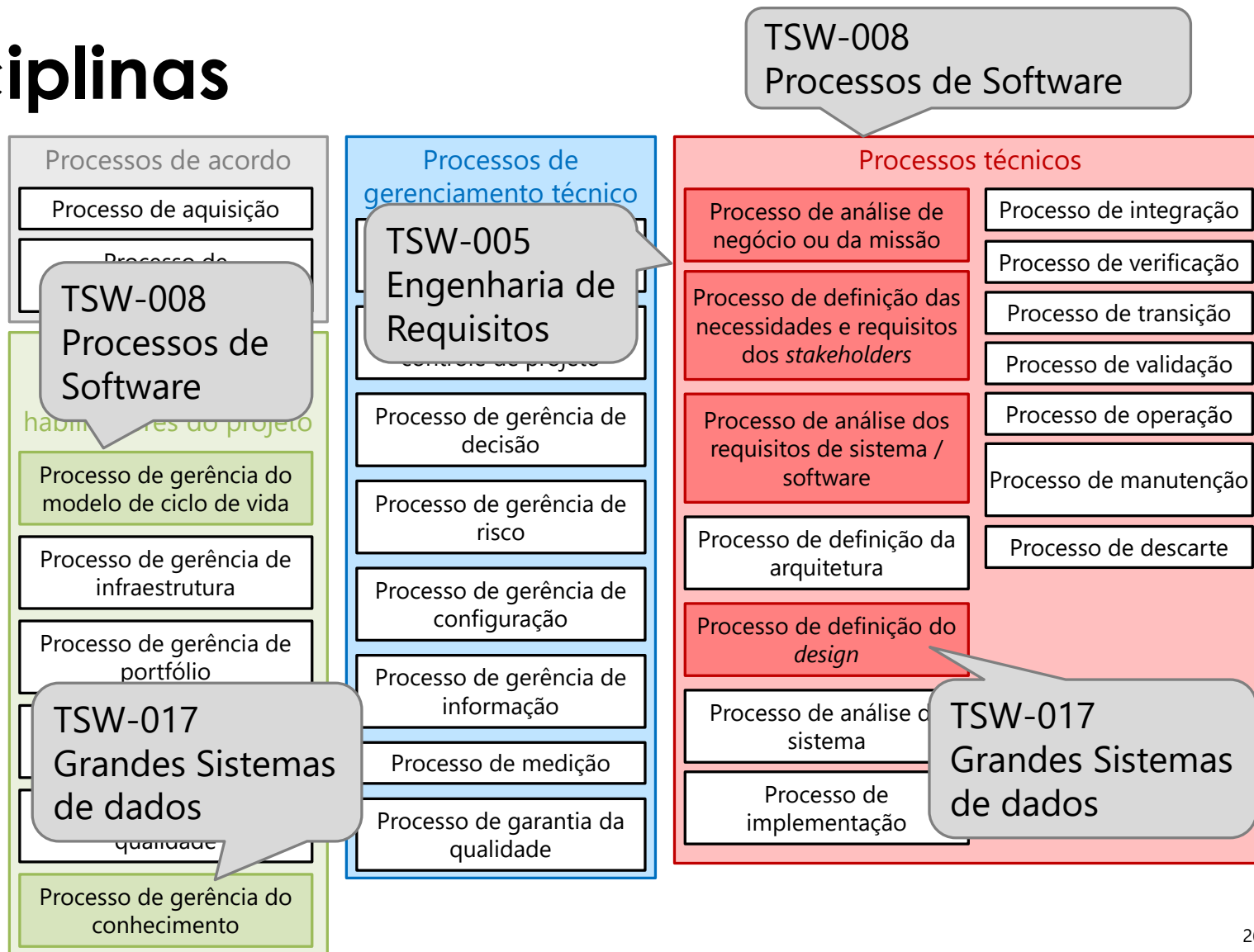




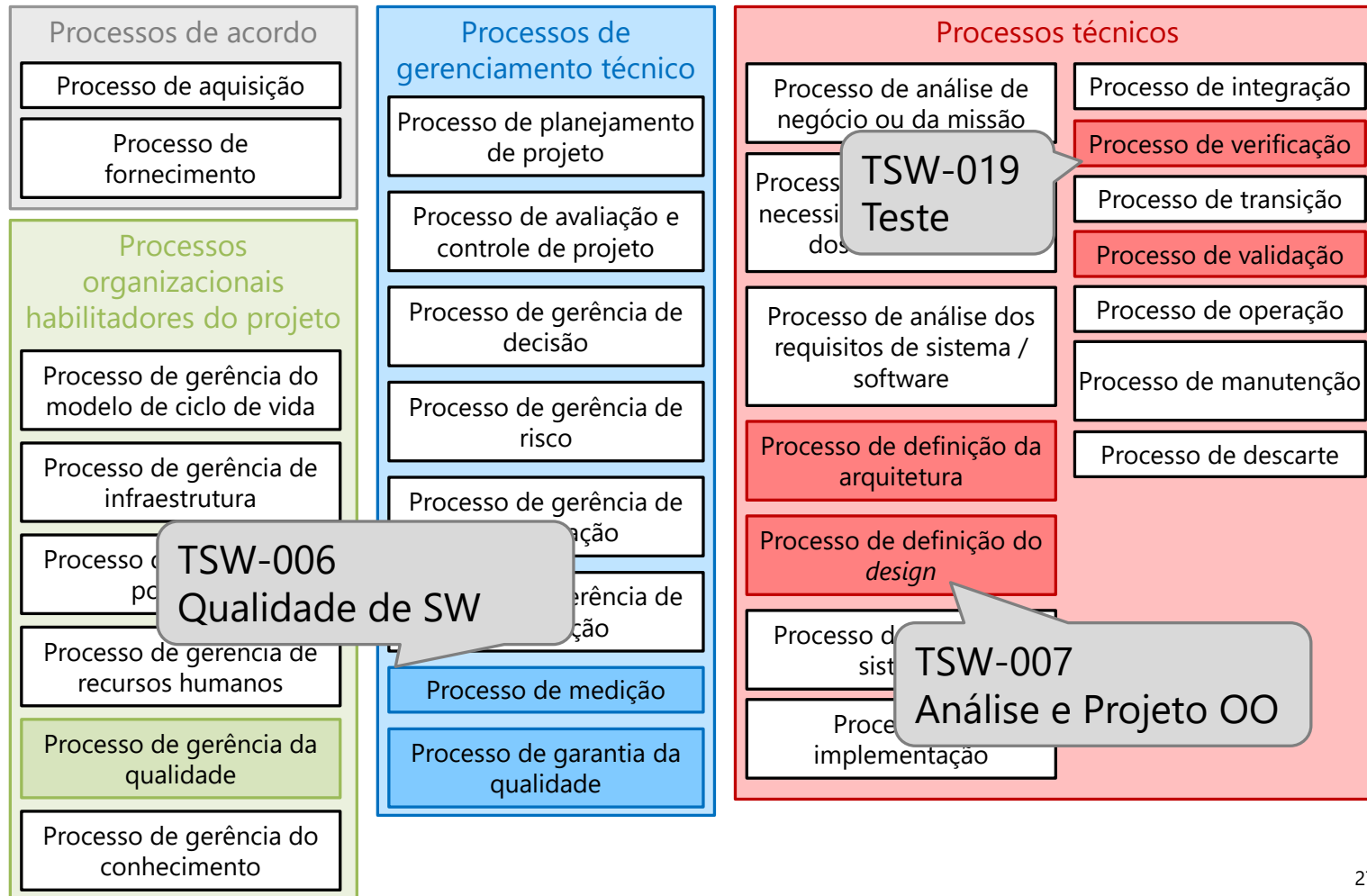
# Disciplinas



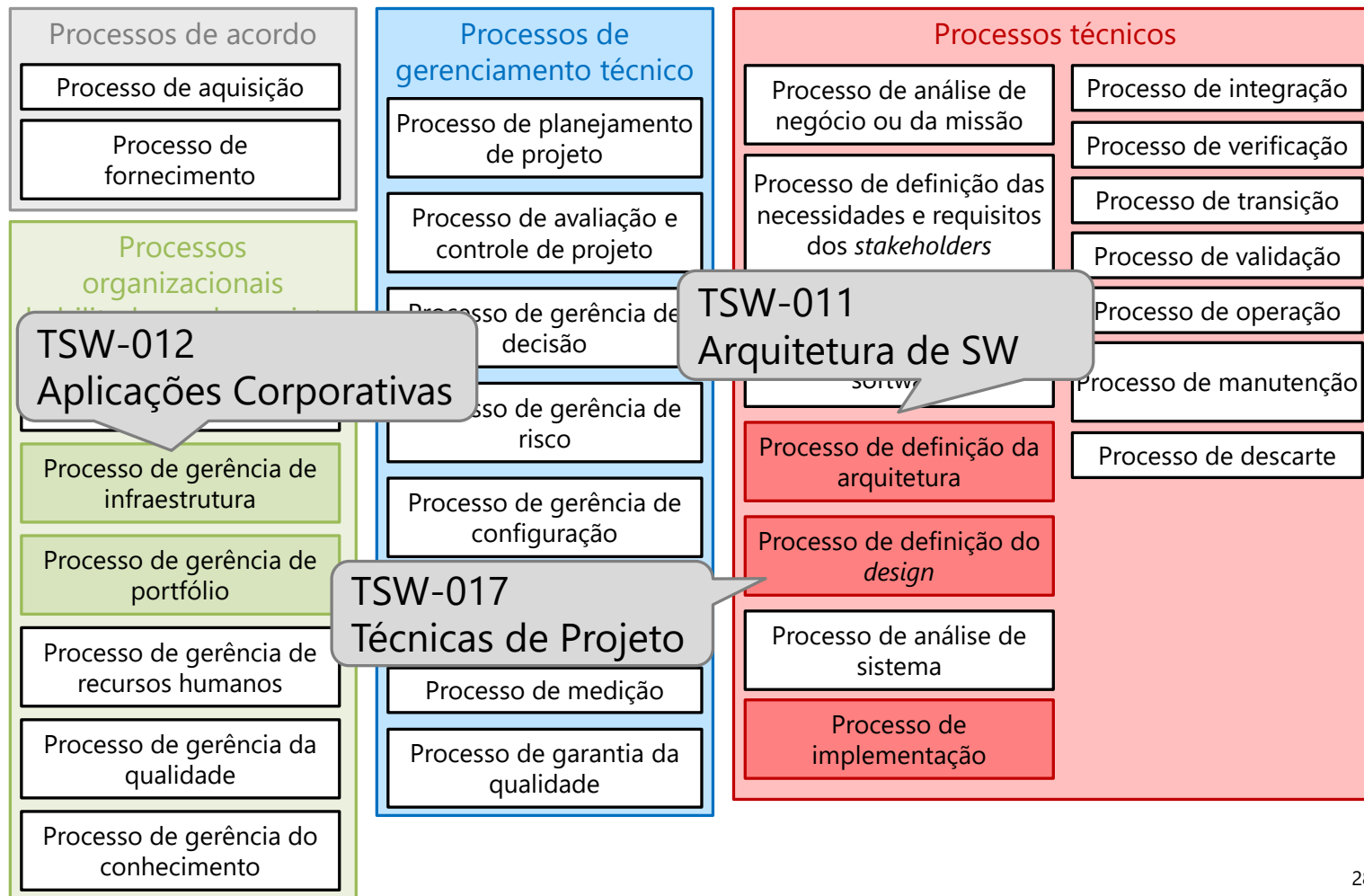
# Disciplinas



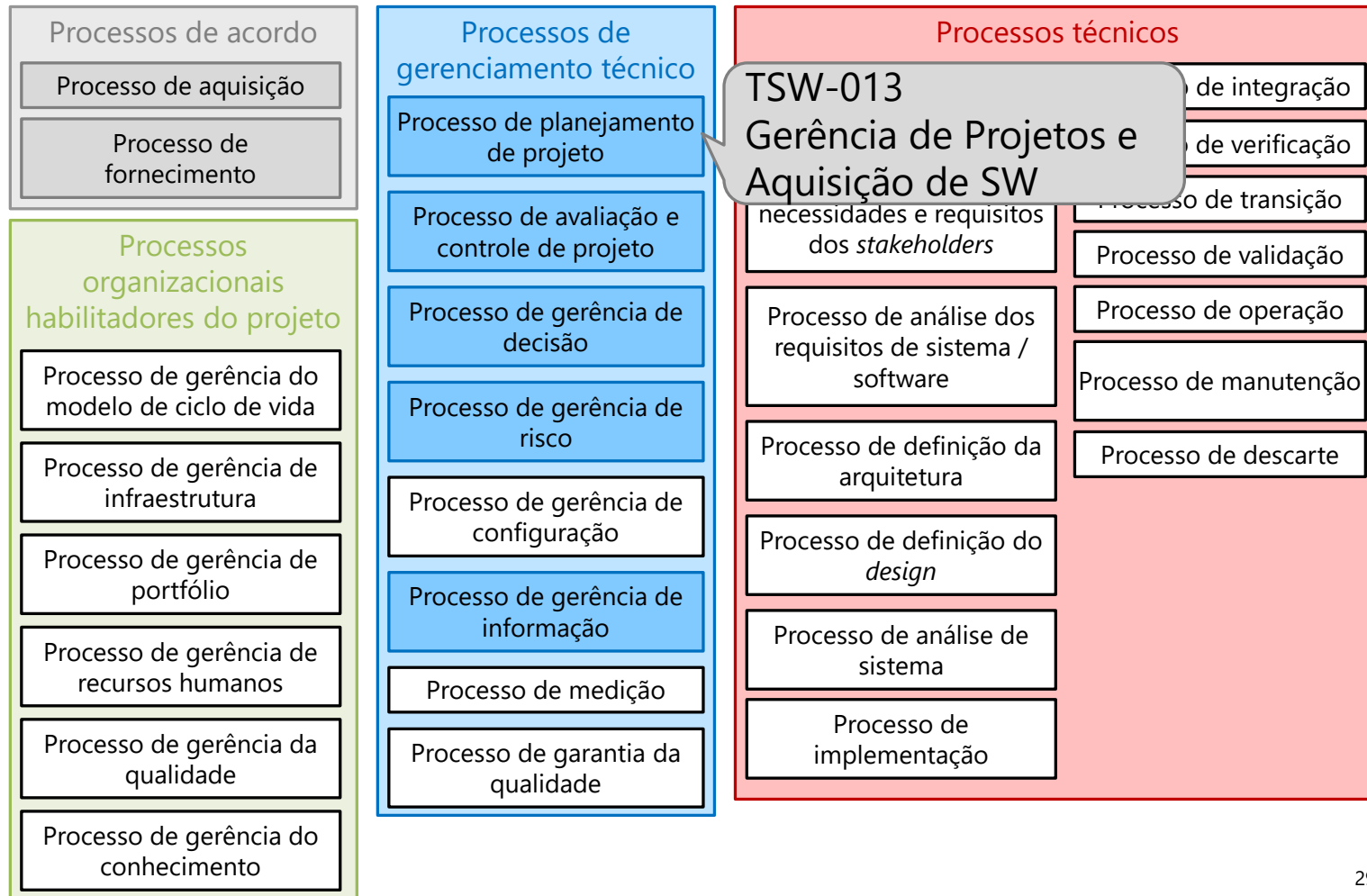
# Disciplinas



# Disciplinas



# Disciplinas

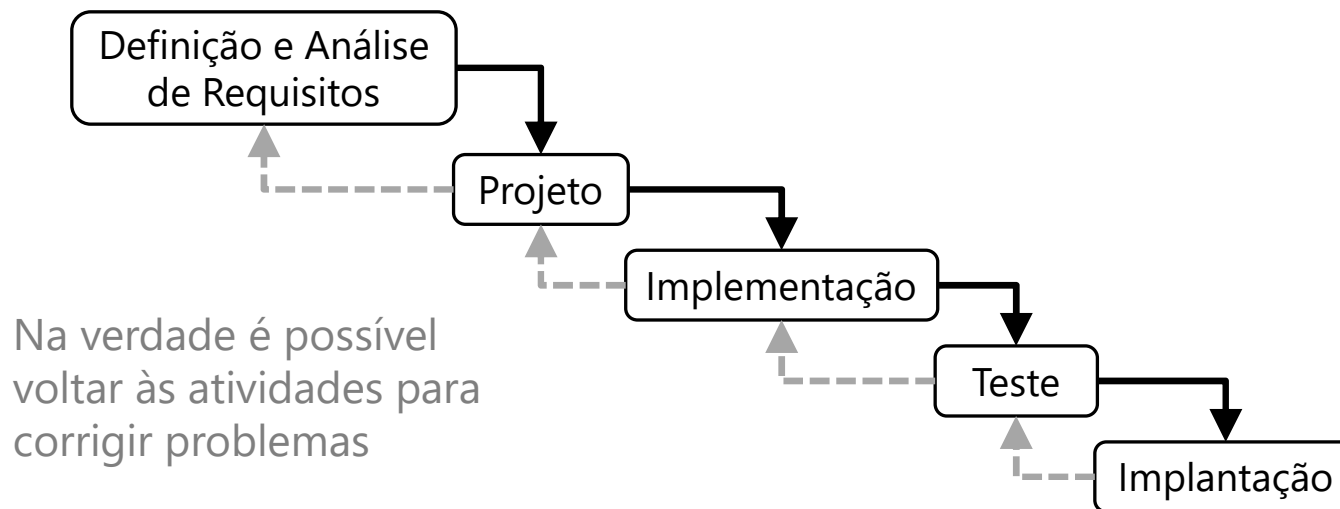


# Processos técnicos

- Não são executados necessariamente em sequência ou mesmo nessa ordem
- Processos são organizados em **modelos de ciclo de vida**
  - São como *frameworks* para processos
  - Os mais comuns
    - Cascata
    - Iterativo
    - Incremental
    - Iterativo e incremental

# Modelo cascata

- As atividades são executadas em sequência
  - Uma atividade só começa quando a anterior termina
  - O resultado de cada atividade é aprovado
- O software é entregue de uma vez, no final



# Modelo cascata

- Modelo descrito por Royce em 1970
  - Também chamado de "Modelo Clássico"
- Vantagens
  - Diminui esforço de gestão
  - Modelo *simples*
    - Fácil entender as atividades



# Modelo cascata

- Problemas
  - Comunicação entre as atividades
  - O cliente só vê o software funcionando no final
  - Defeitos demoram para aparecer
  - Custo maior para corrigir erros
    - Alteração afeta outras partes do software
    - Indisponibilidade da equipe original
  - Requisitos ficam congelados
    - ...mas o ambiente de negócio muda...
  - As últimas atividades sofrem com o atraso

# Modelos iterativo e incremental

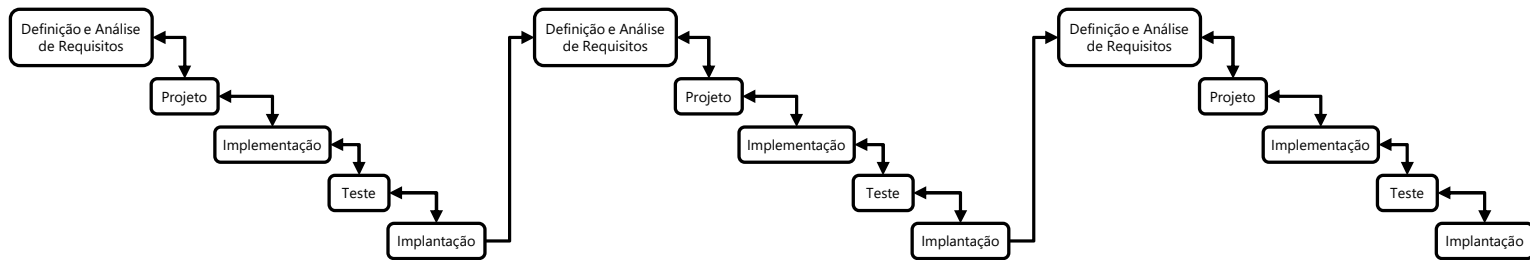
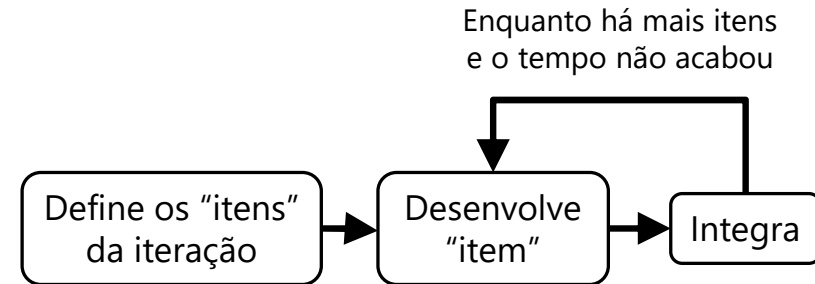
- O software é construído aos poucos

- Execuções seguidas das atividades

- As atividades não *precisam ser no estilo do cascata*

- Cada fase entrega parte do software

- Software é entregue com maior frequência

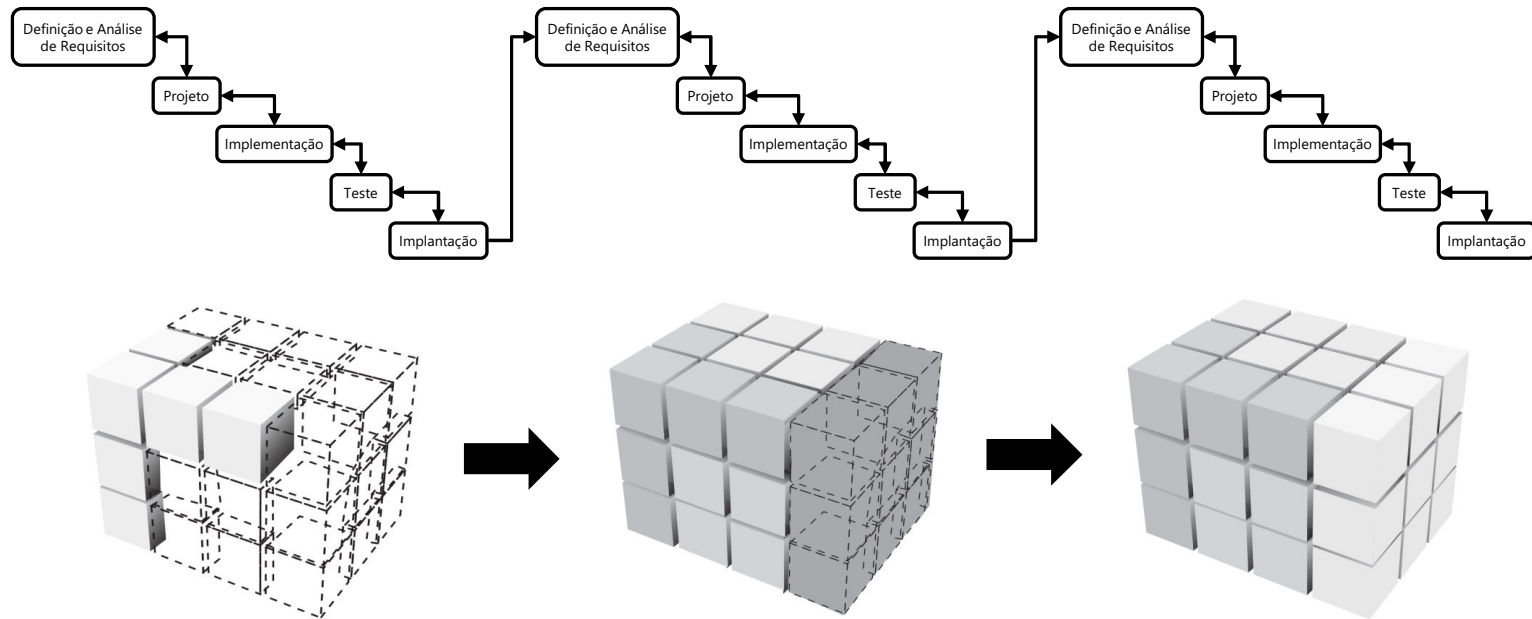


- Modelos iterativo e incremental

- Qual a diferença deles?

# Modelo incremental

- O software é construído em "pedaços"
  - Cada *incremento* entrega um conjunto de requisitos



(Cockburn, 2008)

# Modelo incremental

- 1º incremento constrói a parte central
  - Incrementos adicionam outras funcionalidades
- Vantagens
  - Prioriza o desenvolvimento
  - As últimas atividades não sofrem com o atraso
    - ...mas os últimos **incrementos** sofrem...

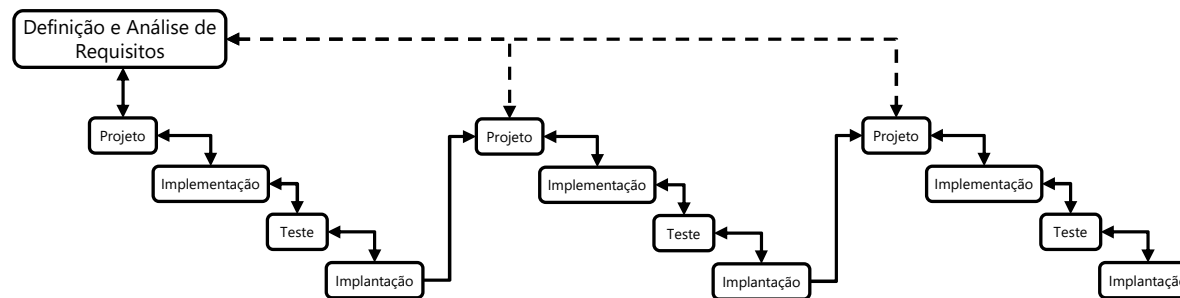
# Modelo incremental

## ■ Problemas

- Atraso nos últimos incrementos
- Não lida com mudanças nos requisitos
  - As partes já criadas não são corrigidas (= cascata)

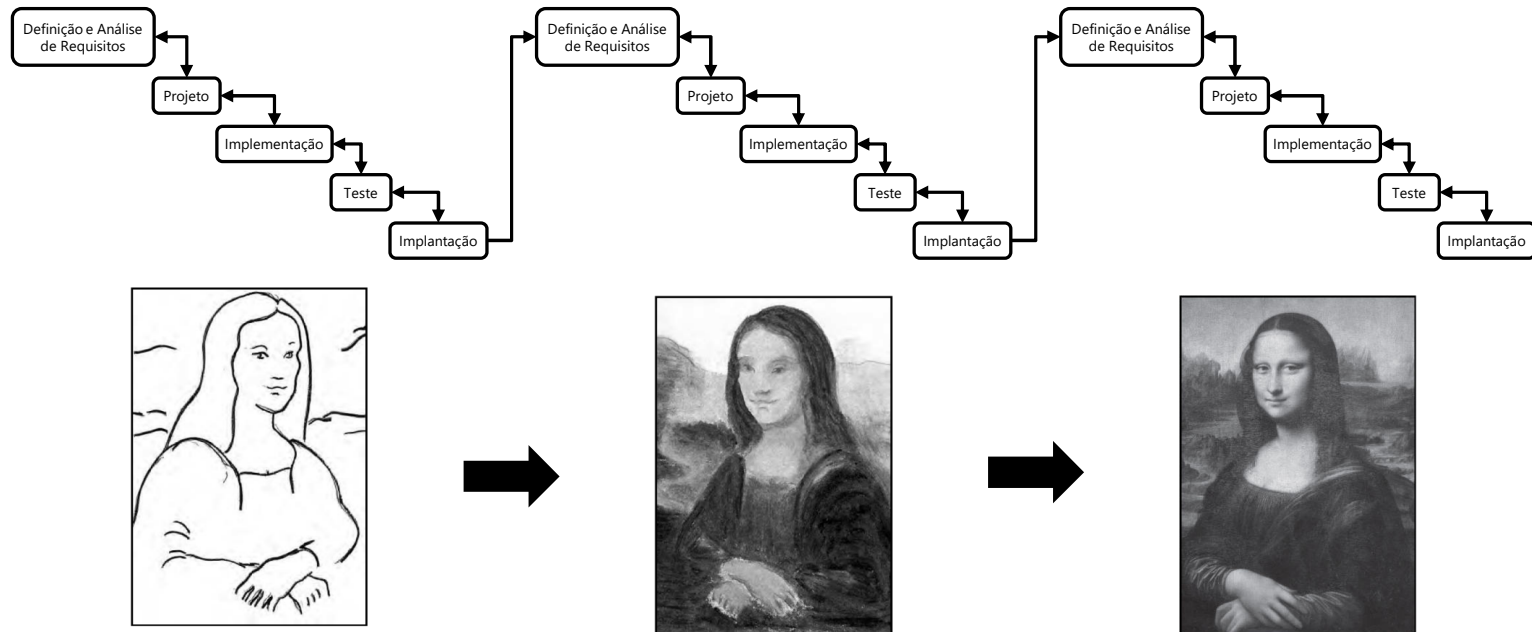
## ■ Variação

- Definição dos requisitos é feita inteira no início



# Modelo iterativo

- Cada *iteração* revisa e melhora o software
  - Participação do cliente



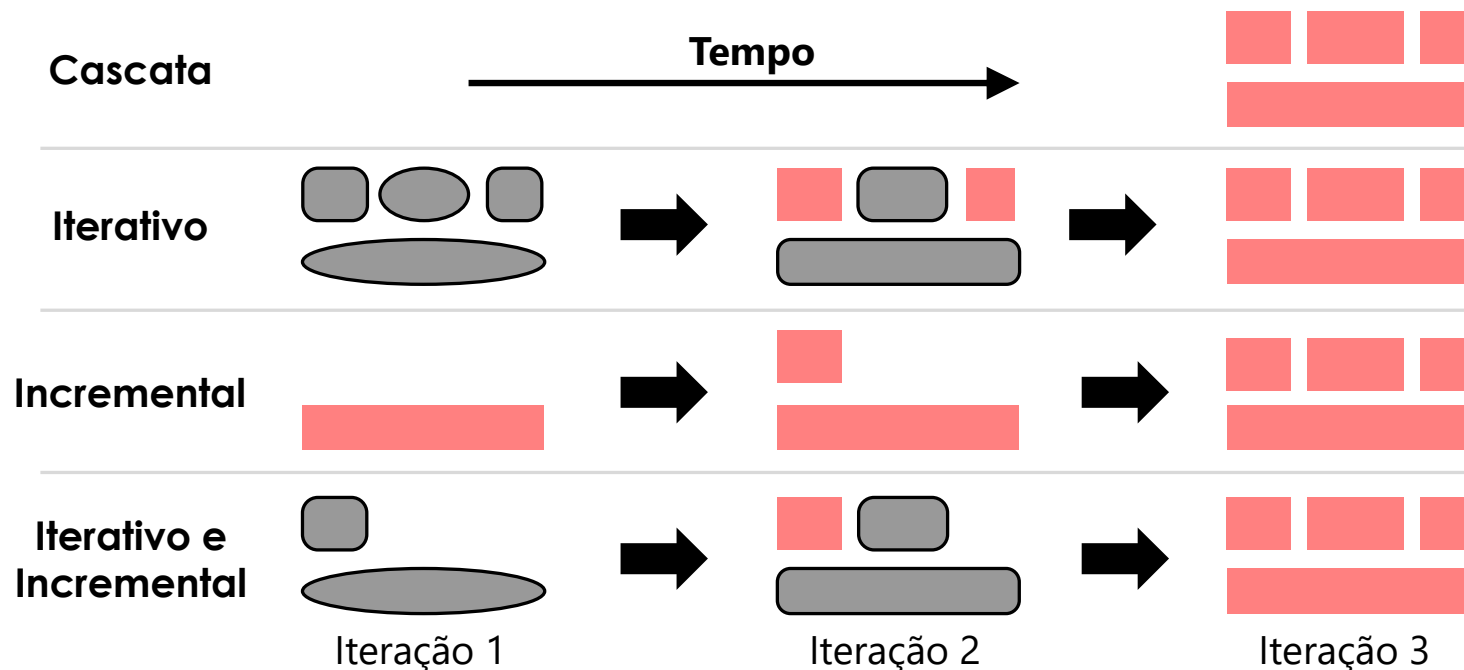
(Cockburn, 2008)

# Modelo iterativo

- Vantagens
  - Lida naturalmente com mudanças nos requisitos
    - O ambiente de negócio muda
    - O entendimento do software é melhorado
- Problemas
  - Retrabalho
  - Primeiras versões têm diversos problemas
  - Não prioriza as funcionalidades
  - Importância de uma boa arquitetura
  - Refinamento *infinito*

# Modelo iterativo e incremental

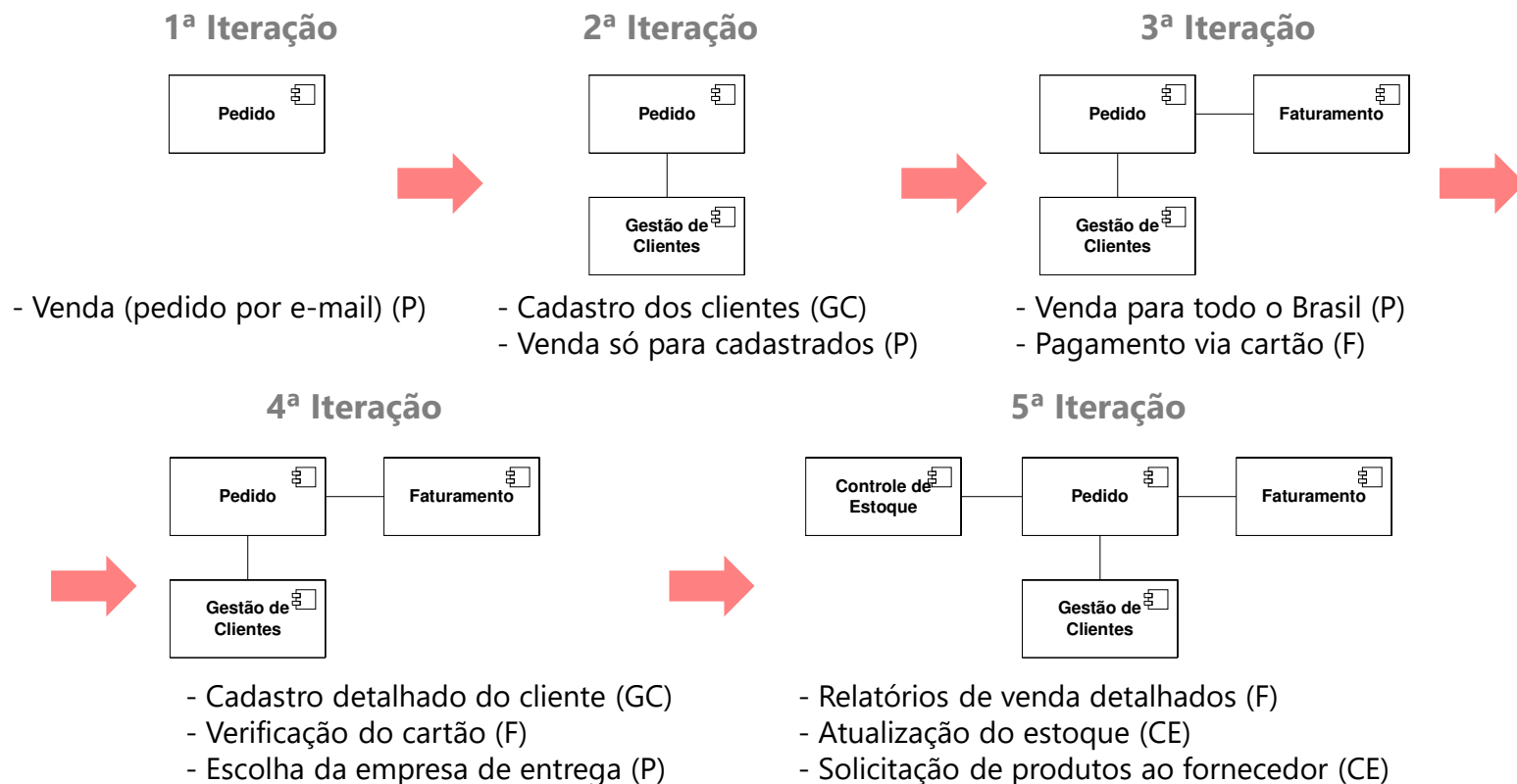
- Junta os benefícios dos dois modelos
  - Prioriza o desenvolvimento
  - Permite lidar com mudanças





# Modelo iterativo e incremental

## ▪ Exemplo: loja virtual



# Modelo iterativo e incremental

- É normal que o software entre em produção após um certo ponto
- **Problemas**
  - Planejamento
    - Quão iterativo e quão incremental?
      - *Quantas* iterações?
      - *Quantos* requisitos em cada iteração?
      - *Quais* requisitos em cada iteração?
  - Refinamento infinito

# Conclusão

- Quando usar cada modelo?
    - Cascata: ainda faz sentido?
    - Incremental
    - Iterativo
    - Iterativo e incremental
    - (Existem outros modelos)
- } Faz sentido usar separado?

# Conclusão

- “Cascata X Ágil”
  - Ágil é um conjunto de abordagens que usam o **modelo iterativo e incremental**
    - Possuem outras características (Aula 6)
  - Existem outras abordagens iterativas e incrementais
    - *Exemplo*: Processo Unificado

# Referências

- COCKBURN, A. **Using Both Incremental and Iterative Development.** CrossTalk, v.21, i.5, pp.27-30, 2008.
- ISO. **Systems and software engineering - Software life cycle processes.** ISO/IEC/IEEE 12207, 2017.
- PFLEEGER, S. L.; ATLEE, J. M. **Software Engineering: Theory and Practice.** 4. ed. Upper Saddle River: Prentice Hall, 2010.
- ROYCE, W. W. **Managing the development of large software systems.** IEEE WESCON, p.1-9, August, 1970.
- SCAMPI Upgrade Team. **Standard CMMI Appraisal Method for Process Improvement (SCAMPI SM ) A, Version 1.3: Method Definition Document.** CMU/SEI-2011-HB-001. March, 2011.