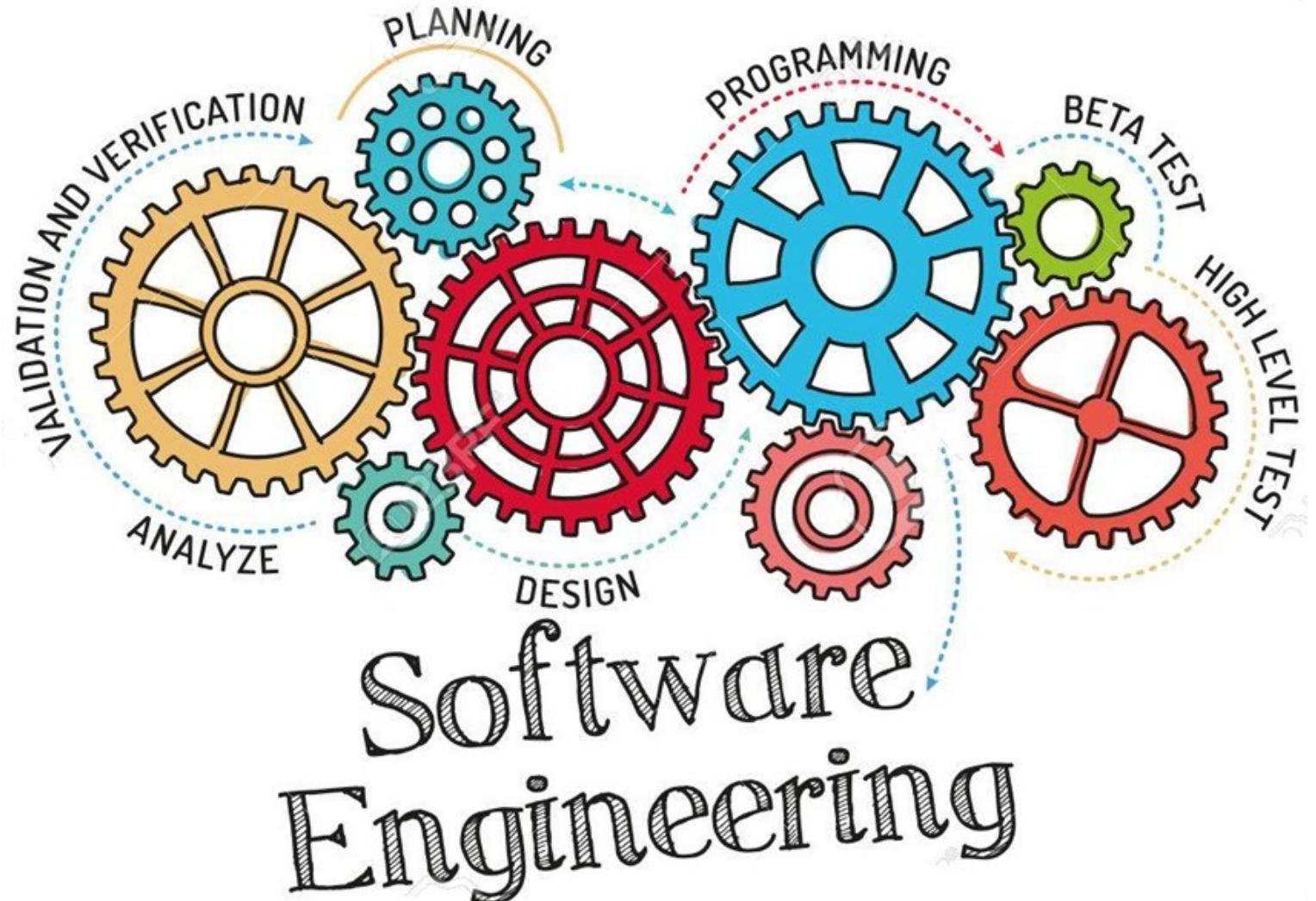


# Engenharia Software II

---

Arquitetura e Projeto de Software, Gerência de Configuração, Verificação e Validação



Source: <https://www.devante.com.br/noticias/engenharia-de-software/>

# Professor

**Prof. Dr. Rafael Queiroz Gonçalves, PMP**

## Formação acadêmica – Ciência da Computação

- Doutorado (2013-2017 - UFSC)
- Mestrado (2011-2012 - UFSC)
- Bacharelado (2006-2010 - UNIVALI)

## Formação profissional (certificações)

- Gerente de projetos profissional (PMP)
- Professional Scrum Master (PSM)
- ITIL (AXELOS)
- COBIT (ISACA)
- Arquiteto de sistemas JEE (SUN)
- Profissional UML (OMG)
- Programador JAVA (SUN/ORACLE)
- Programador PHP (Zend technologies)
- Administrador de sistemas LINUX (LPI)
- Administrador de sistemas Windows Server (MTA)
- DBA MySQL (ORACLE)
- DBA MONGODB (MongoDB Inc)
- Microsoft PowerBI Data Science
- IBM Lotus Notes Application Developer



## Atuação docente

- **2018:** Graduação e pós graduação
- **2010:** Tecnólogos e Técnicos

## Atuação profissional

- **2013 – atual:** Auditor de Controle Externo e Gestor de TI - Tribunal de Contas do Estado
  - Gerente de projetos
  - Assessor de Governança de TI
  - CTO - Diretor de TI
- **2008-2012:** Desenvolvedor de Software – Beck et al. Services GmbH
- **2006-2008:** Desenvolvedor de Software - Dualline

**Lattes:** <http://lattes.cnpq.br/212895803559843>

**LinkedIn:** <https://www.linkedin.com/in/rafaelqg>

**Youtube:** <https://www.youtube.com/@queiroz-rafael>

# Engenharia de Software II

---

## Ementa:

- Arquitetura e projeto de software
- Gerência de Configuração
- Verificação e Validação

5 ° Período

**ENGENHARIA DE SOFTWARE II - 22820**

C.H. Teórica: 36 C.H. Prática: 36 C.H. Total: 72 Créditos: 4

### Ementa:

Arquitetura e projeto de software. Gerência de Configuração. Verificação e Validação.

### Referências:

#### Básica

PRESSMAN, Roger S.; MAXIM, Bruce R. **Engenharia de software:uma abordagem profissional**. 8. ed. São Paulo, SP: AMGH Editora, 2016. xxviii, 940 p. ISBN 9780078022128. (1 Exemplar)

SOMMERVILLE, Ian. **Engenharia de software**. 9. ed. São Paulo, SP: Pearson Education do Brasil, c2011. xiii, 529 p. ISBN 9788579361081. (5 Exemplares)

WAZLAWICK, Raul Sidnei. **Engenharia de software:conceitos e práticas** . Rio de Janeiro, RJ: Elsevier, c2013. ISBN 9788535260847. (3 Exemplares)

#### Complementar

HASS, Anne Mette Jonassen. **Configuration management principles and practice**. Boston [Estados Unidos]: Addison Wesley, c2003. xlv, 370 p. (The agile software development series) ISBN 0321117662. (4 Exemplares)

LEON, Alexis. **Software configuration management handbook**. Third edition. Boston: Artech House, [2015] 1 online resource ISBN 9781608078448 (electronic bk.).

MYERS, Glenford J.; SANDLER, Corey; BADGETT, Tom. **The art of software testing**. 3rd ed. Hoboken, N.J.[EUA]: John Wiley & Sons, c2012. xi, 240 p. ISBN 9781118031964. (3 Exemplares)

PATTON, Ron. **Software testing**. 2nd. ed. Indianapolis, IN: Sams Pub., c2006. xiv, 389 p. ISBN 0672327988. (1 Exemplar)

PFLEEGER, Shari Lawrence. **Engenharia de software:teoria e prática**. 2. ed. São Paulo, SP: Prentice Hall, 2004. xix, 535 p. ISBN 8587918311. (5 Exemplares)

# Engenharia de Software I

## Ementa:

- Fundamentos da engenharia de software.
- Ciclo de vida de software.
- Métodos de desenvolvimento de software.
- Engenharia de requisitos.

Disciplina anterior...

Curso de CIÊNCIA DA COMPUTAÇÃO-KOBRASOL		
PLANO DE ENSINO SUJEITO A REVISÃO PEDAGÓGICA		
IDENTIFICAÇÃO		
Curso: CIÊNCIA DA COMPUTAÇÃO		
Disciplina: ENGENHARIA DE SOFTWARE		
Período: 4º		
CH Teórica: 45 (54 horas-aula)	Créditos: 4	
CH Prática: 15 (18 horas-aula)		
CH Total: 60 (72 horas-aula)		
OBJETIVO GERAL		
Trabalhar em todos os estágios do ciclo de vida de desenvolvimento de software, desde o entendimento do negócio e especificação de requisitos até a entrega e manutenção de produtos de software.		
EMENTA		
Fundamentos da engenharia de software. Ciclo de vida de software. Métodos de desenvolvimento de software. Engenharia de requisitos.		

# Método de aprendizagem

## Aprendizagem Baseada em Projetos

Essa metodologia ativa envolve pesquisa, trabalho em grupo e atividades práticas, que trazem significado real para os conteúdos ensinados.

“A aprendizagem baseada em projetos é um modelo de ensino que consiste em permitir que os estudantes confrontem as questões e os problemas da vida real que consideram significativos, determinando como abordá-los e, então, agindo de forma cooperativa em busca de soluções.”

Essa definição é dada pelo educador norte-americano William Bender no livro “Aprendizagem baseada em projetos: Educação diferenciada para o século XXI”. Para ele, essa metodologia “é uma das mais eficazes formas disponíveis de envolver os alunos com o conteúdo de aprendizagem”.



# Trabalho da disciplina (Projeto)

Trabalho em equipes: grupos de **3 alunos**.

- Não deve ser realizado trabalho individualmente.

Tema do projeto de desenvolvimento de software:

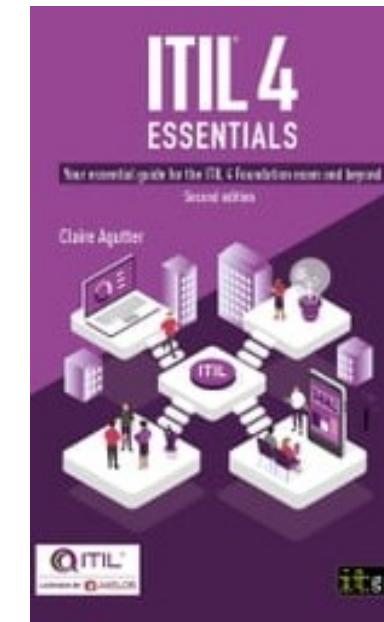
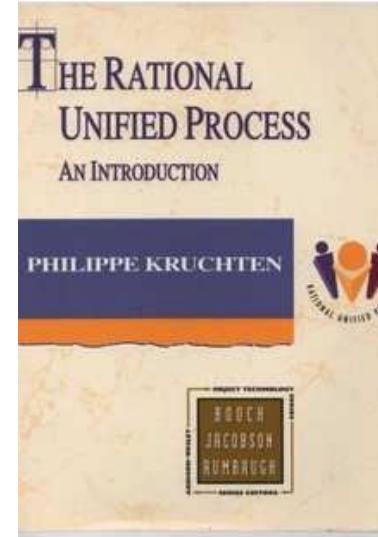
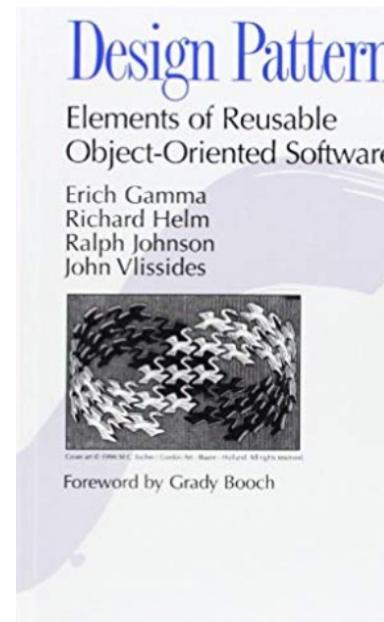
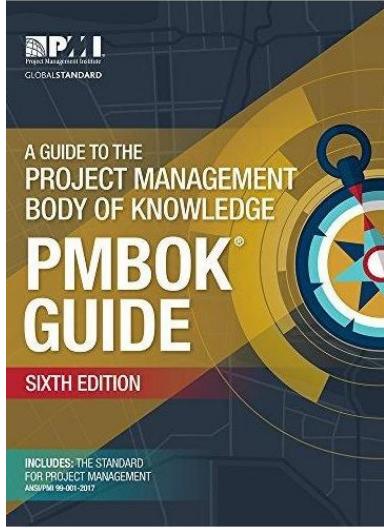
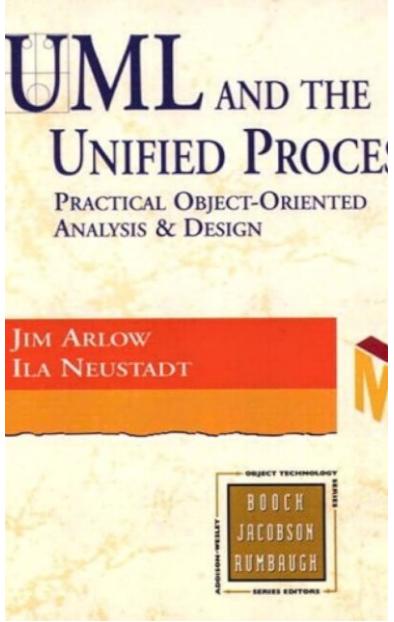
- A definir pela equipe.

Alguns requisitos técnicos mínimos são necessários a fim de colocar em prática os conceitos da disciplina de Engenharia de software II.

- Escopo mínimo do projeto:
  - Deve incluir armazenamento em SGBD.
  - Deve incluir ao menos 3 entidades fortes persistentes.
  - Deve incluir ao menos 3 casos de uso.
  - Deve incluir usuário com interação por meio de telas.
  - Deve aplicar 1 padrão de projeto GOF de cada categoria.

O projeto servirá como contexto para produção de artefatos que serão trabalhados ao longo da disciplina qual cobrem o ciclo de vida do projeto e do produto.





# Principais modelos/frameworks abordados

# Cronograma de aulas

## Arquitetura e projeto de software

- Aula 1: Visão geral da disciplina e RUP
- Aula 2: UML e Padrão de projetos (Design Patterns - GOF)
- Aula 3: RUP – Definindo arquitetura do projeto
- Aula 4: PMBOK: grupo de processos de iniciação
- Aula 5: Aula prática: elaboração dos artefatos da M1
- **Aula 6: T1 – Entrega e apresentação do trabalho M1**

## Gerência de Configuração

- Aula 7: PMBOK – Planejamento de escopo
- Aula 8: PMBOK – Planejamento de tempo
- Aula 9: PMBOK – Planejamento de riscos
- Aula 10: ITIL - Gestão de serviços e Gestão de mudanças
- Aula 11: Aula prática: elaboração dos artefatos da M2
- **Aula 12: T2 – Entrega e apresentação do trabalho M2**

## Verificação e Validação

- Aula 13: casos de teste e homologação
- Aula 14: teste unidade e automatização
- Aula 15: teste integração e automatização
- Aula 16: Teste sistema e automatização
- Aula 17: Aula prática: elaboração dos artefatos da M3
- **Aula 18: T3 – Entrega e apresentação do trabalho M3.**

Todos os trabalhos envolveram apresentação expositiva aos demais integrantes da turma.  
A aula que precede o trabalho de cada média, haverá tempo disponível em sala para execução e preparação dos artefatos de entrega.

\*Ajustes poderão ser realizados de acordo adequações necessárias ao curso do semestre.

---

# Aula 1



Visão geral da disciplina



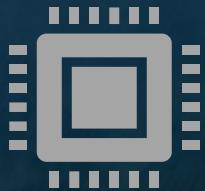
Revisão métodos ágeis

# Distinguindo conceitos



## Gerência de Projeto de Software

- Ágeis
- Backlogs
- Cerimônias
- Incrementos
- Clássico (ex. PMBOK)
  - Termo de abertura de projeto
  - Plano de tempo (cronograma)
  - Plano de custos
  - Plano de escopo
  - Plano de riscos
  - Plano de aquisição
  - Plano de comunicação
  - Plano de recursos



## Projeto Técnico de Software (Design ou Arquitetura)

- Modelagem (Diagramação) de dados
- Modelagem (Diagramação) de estruturas (estático)
- Modelagem (Diagramação) de comportamento (dinâmico)
- Prototipação



## Produto de Software

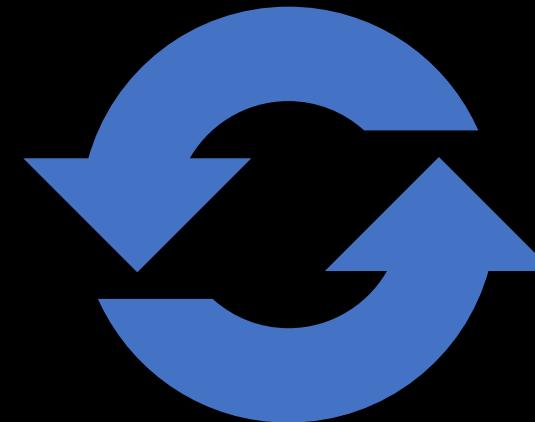
- Gerência do serviço
- Gerência de configuração
- Manutenções

# Revisão sobre métodos ágeis





Ciclo de vida  
de software



# Ciclo de vida de projeto

## Questão:

- Sobre ciclo de vida do software, é correto afirmar que:

A

A fase de teste permite identificar o que o sistema proposto deve fazer.

B

A fase de implementação concentra-se em especificar o que o sistema proposto deve fazer e como atingirá as metas definidas na fase de análise.

C

Se o sistema for um produto genérico, vendido em um mercado competitivo, a fase de implementação envolve uma extensa investigação para identificar as necessidades dos consumidores em potencial.



A fase de implementação envolve a elaboração de programas propriamente ditos, a criação dos arquivos de dados e o desenvolvimento do banco de dados.

E

A fase de projeto está intimamente ligada a fase de implementação, pois cada módulo do sistema é testado de acordo com o progresso de sua implementação.

# Ciclo de vida de projeto - Questão

É correto afirmar que a modelagem ágil é uma abordagem de desenvolvimento de software que enfatiza a entrega contínua de software funcional e:

A adota uma abordagem sequencial, na qual o modelo é desenvolvido em fases distintas, tais como: análise de requisitos, projeto, implementação, testes e manutenção.

B favorece a criação de modelos em partes incrementais, investindo semanas ou até meses na criação de modelos e documentos para cada entrega.

 C incentiva a comunicação constante e a colaboração entre membros da equipe, bem como com os stakeholders do projeto.

D valoriza a documentação em detalhes, mesmo quando os requisitos do software não estão claramente definidos.

A dynamic photograph capturing a scrum in American football. Several players from two teams are crouched low to the ground, their bodies angled towards each other. The player in the foreground, wearing a white jersey with red stripes, has his hands firmly gripping a brown leather football with white stripes. The ball is positioned centrally between the players' legs. The background is a soft-focus green field, suggesting a grassy stadium. The lighting is bright, casting sharp shadows on the players' uniforms.

SCRUM

# SCRUM - Questão

No contexto das metodologias ágeis, o que é o artefato Backlog?

- A) Um conjunto de requisitos do projeto a serem definidos.
- B) Uma lista de atividades planejadas a serem realizadas em uma sprint.
- C) Um documento que contém todos os testes a serem realizados.
- D) Uma lista de itens do produto a serem desenvolvidos e priorizados.**

# SCRUM - Questão

Segundo “O Guia do Scrum”, versão 2020, em português, disponível no site <https://scrumguides.org>, a Sprint \_\_\_\_\_ conclui uma Sprint. Assinale a alternativa que preenche corretamente a lacuna do trecho acima.

- A Planning
- B Review
- C Retrospective
- D Backlog
- E Final

**Sprint Review** focuses on the product, while **Sprint Retrospective** focuses on the process.

	Sprint review	Sprint retrospective
When does it take place?	On the last day of the Sprint.	After the Sprint Review, and typically on the same day.
Who participates?	Product Owner Scrum Master Development Team Stakeholders	Product Owner Scrum Master Development Team
What are the goals?	Review delivered increment Update the product backlog Prep for the next sprint planning	Identify what went right Identify what can be improved Improve culture Re-energize the team

Source: <https://www.wrike.com/scrum-guide/faq/what-is-the-difference-between-sprint-review-and-retrospective/>

# SCRUM - Questão

Segundo “O Guia do Scrum”, versão 2020, em português, disponível no site <https://scrumguides.org>, um Scrum Team consiste em:



A Um Scrum Master, um Product Owner e Developers.

B

Um Scrum Manager, um Product Master, Developers e Testers.

C

Um Product Master, um Scrum Owner e Developers.

D

Um Product Manager, um Scrum Master e Developers.

E

Um Scrum Manager, um Product Owner, Developers e um Tester.

# SCRUM – Questão

Na metodologia ágil Scrum, os sprints são eventos de duração fixa de um mês ou menos que podem ser cancelados se a Meta do sprint se tornar obsoleta. O único integrante do Scrum Team que tem autoridade para cancelar o sprint é o:

A Stakeholder;

B Desenvolvedor;

C SCRUM master

 Product Owner;

E Analista de Teste.

# SCRUM - Questão

O Scrum é um Ciclo de Vida Ágil, proposto para o desenvolvimento de software, baseado nos princípios da transparência, inspeção e adaptação, que emprega uma abordagem para que o desenvolvimento dos entregáveis aconteça de forma:

A

incremental e interativa

C

interativa e iterativa

D

linear e incremental

E

linear e interativa

# SCRUM - Questão

Sobre o método de desenvolvimento ágil de software Scrum, é correto afirmar:

- A O *backlog* do produto é uma lista priorizada de requisitos ou características que não muda.
- B O *backlog* do produto é um subconjunto dos itens do *backlog* da *sprint*.
- C O *product owner* atua como facilitador para os membros da equipe *scrum* e comanda a reunião diária.
-  D As tarefas são realizadas em um período de tempo chamado de *sprint*.
- E O *scrum master* define o modo como a equipe de desenvolvimento completa o incremento de código.

# SCRUM – Questão

O Scrum é um framework ágil para gerenciamento de projetos, amplamente utilizado no desenvolvimento de software e possui 3 papéis principais: o Product Owner, o Scrum Master e a Equipe de Desenvolvimento. O Product Owner é responsável por:



desenvolver e gerenciar o *Product Backlog*, que é a lista de funcionalidades e requisitos do produto.

B fornecer soluções técnicas e resolver impedimentos que possam surgir durante o processo de desenvolvimento.

C garantir que a equipe de desenvolvimento esteja seguindo o processo Scrum corretamente e que esteja cumprindo as responsabilidades atribuídas a ela.

D liderar e facilitar as reuniões do *Scrum*, como o *Sprint Planning*, a *Daily Scrum*, a *Sprint Review* e a *Sprint Retrospective*.

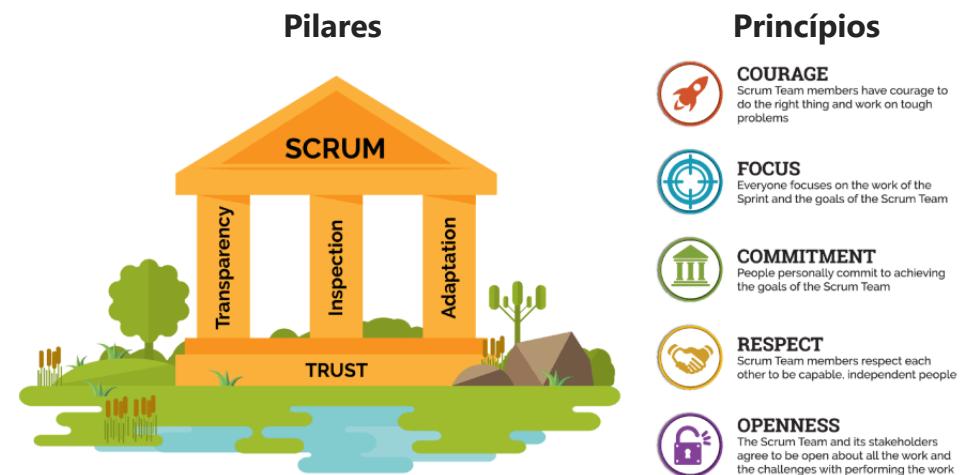
# SCRUM - Questão

A utilização de metodologias ágeis em projetos de desenvolvimento de sistemas da informação é uma boa prática recomendada e o framework Scrum é uma opção que, por meio de facilitadores adaptativos, dentro da uma perspectiva de abordagem iterativa e incremental, permite o atingimento de metas para soluções de problemas complexos. Acerca do conceito dos pilares empíricos do Scrum. Neste contexto, analise os itens a seguir:

- I - O processo emergente e o trabalho devem ser visíveis tanto para quem executa o trabalho quanto para quem recebe o trabalho é um conceito identificado no pilar da **transparência**.
- II - Os artefatos e o progresso em direção às metas acordadas devem ser inspecionados com frequência e diligência para detectar variações ou problemas potencialmente indesejáveis são conceitos ligados ao pilar da **inspeção**.
- III - Se algum aspecto de um processo se desviar fora dos limites aceitáveis ou se o produto resultante for inaceitável, o processo que está sendo aplicado ou os materiais que estão sendo produzidos devem ser ajustados é uma orientação contida no conceito do pilar da **adaptação**.

Em relação aos itens apresentados, pode-se afirmar que:

- A) somente os itens II e III são verdadeiros.
- B) todos os itens são verdadeiros.
- C) somente os itens I e II são verdadeiros.
- D) somente o item III é verdadeiro.
- E) somente o item I é verdadeiro



Credit: ABN AMRO Bank N.V.

Source: <https://medium.com/@anwarhermuche/introdu%C3%A7%C3%A3o-ao-scrum-8ad0a5fd483a>



# Kanban



# Kanban - Questão

Muitas organizações aplicam os princípios e práticas de gestão com Kanban para alcançar os objetivos de seus projetos de desenvolvimento de software.

Sobre Kanban, assinale a afirmativa correta.

A

Incentiva ciclos anuais de feedbacks porque cadências de reuniões e revisões recorrentes criam muitas interrupções e impactam o tempo de entrega.

B

É um método de gestão baseado no Kaizen e que valoriza entrega de *software* funcional completo, sem entregas parciais, trabalhos inacabados e débitos técnico.

C

Fomenta uma cultura organizacional que privilegia o indivíduo em relação à coletividade, orientada ao cliente e sem compartilhar propósitos e metas.

D

Pressupõe que o escopo do projeto é sempre estável e o desenvolvimento de *software* precisa seguir de modo linear para garantir a produtividade.



Permite visualizar o processo e o fluxo de trabalho de projetos, programas e portfólios por meio de um conjunto de quadros interconectados.

**Kaizen** é uma filosofia japonesa que acredita na melhoria contínua, onde tudo pode ser aprimorado através de pequenos ajustes que se transformarão em grandes mudanças. A palavra Kaizen, em sua origem, significa melhoria e/ou mudança.

A large industrial steel structure, possibly a reactor vessel or pressure vessel, is being lowered by a crane into a factory floor. The structure is massive, with a yellow cylindrical base and a grey upper section. The factory is filled with industrial equipment, steel beams, and workers in safety gear. The lighting is dramatic, with bright overhead lights and deep shadows.

# Processo Lean Manufacturing

# Lean - Questão

O método Lean é um dos principais métodos ágeis utilizados na gestão de projetos. Sobre essa metodologia, assinalar a alternativa CORRETA:

A É utilizada na gestão de projetos que não tenham prazo de entrega, utilizando intervalos de tempo para desenvolver cada etapa.

B É composta por *checklists*, oferecendo uma visão global do projeto. É específica para alguns tipos de negócio, pois busca a revolução dos processos.

C É uma boa alternativa para criar objetivos realistas e possíveis para a situação de cada empresa, baseando-se em cinco primórdios, os quais são indicados pelas letras do seu nome.

 É utilizada para projetos mais objetivos ou reduzidos, e identifica eficientemente os desperdícios.

**Processo Lean** é todo fluxo de trabalho que se estrutura conforme os conceitos e princípios da Manufatura Enxuta, a Lean Manufacturing. Criada no Japão, no período pós-guerra, ela é toda baseada na eliminação do desperdício e na otimização dos recursos. É também a base do modelo de produção Toyotista, no qual **a produção é puxada pela demanda**.

Source: <https://www.escolaedti.com.br/processo-lean>

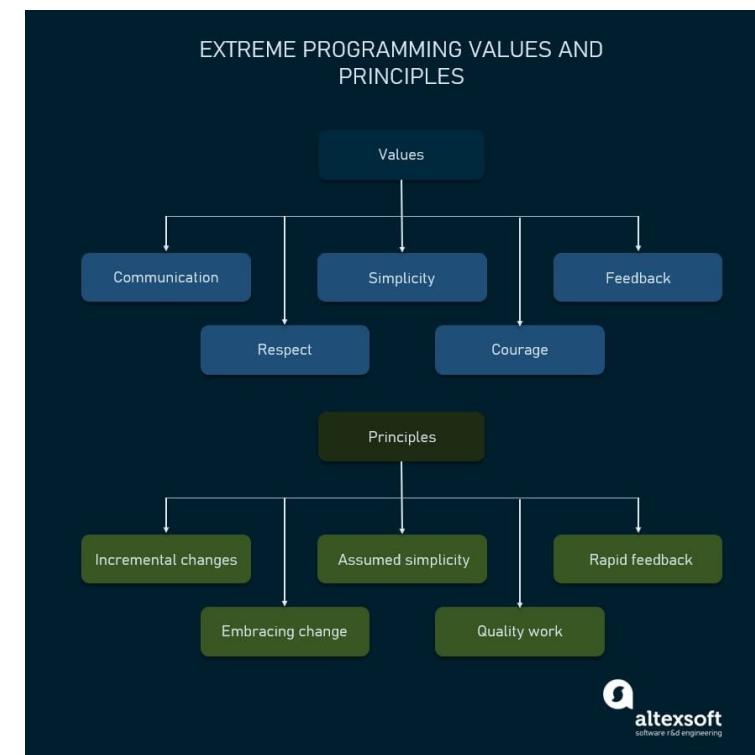
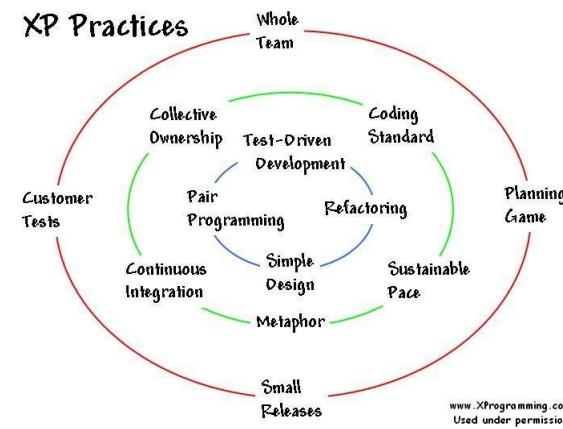
# XP – Extreme Programming

```
types.Operator:  
    X mirror to the selected  
    object.mirror_mirror_x  
    for X
```

# eXtreme Programming - Questão

Assinale a alternativa que apresenta uma característica da metodologia ágil Extreme Programming (XP).

- A Reuniões diárias de 15 minutos.
- B Modelos como artefato primário do processo de desenvolvimento.
- ✓ Programação em dupla.
- D Ciclos adaptáveis.
- E Desenvolvimento dirigido a funcionalidades.



Source:

<https://www.altexsoft.com/blog/extreme-programming-values-principles-and-practices/>  
<https://www.cprime.com/resources/blog/key-xp-practices/>

# eXtreme Programming - Questão

- A metodologia Extreme Programming (XP) é uma abordagem ágil de desenvolvimento de software que se concentra em valores como comunicação, feedback, simplicidade e coragem. Essa metodologia

A

é considerada adequada para projetos de grande porte com requisitos de software bem definidos.



é frequentemente combinada com outras práticas ágeis, como o *Agile Modeling*, para melhorar ainda mais a eficácia do desenvolvimento de software.

C

enfatiza o processo de desenvolvimento de software em cascata, em que as etapas são realizadas em sequência, seguindo uma ordem linear diariamente.

D

propõe a prática da codificação em pares, em que dois programadores trabalham juntos em um único computador, evitando os frequentes testes de integração e minimizando o tempo necessário para resolvê-los.

# Aula 2

## UML e Padrões de projetos (Design Patterns - GOF)

# UML

UML, ou Unified Modeling Language, é uma linguagem de notação destinada à modelação e documentação das fases de desenvolvimento de softwares orientados a objetos.

- A criação da UML foi motivada pela problemática da disparidade entre diversas formas de notação aplicadas em determinada momento. Então a organização Rational Software (posteriormente adquirida pela IBM) desenvolveu a primeira versão da UML entre 1994-1996.
- Em 1997 a UML passou a ser mantida pela Object Management Group (OMG), e em 2005 obteve reconhecimento de ISO (International Organization for Standardization).
  - **ISO/IEC 19501:** <https://www.iso.org/standard/32620.html>



# UML

No final da década de 1980 haviam muitos conflitos de definições e nomenclaturas na área de modelagem. Neste contexto 3 autores das principais metodologias adotadas decidiram trabalhar na definição de um modelo único que veio a ser a UML:

- Ivar Jacobson (OOSE – Object Oriented Software Engineering)
- Grady Booch (The Booch Method)
- James Rumbaugh (OMT –Object Modeling Technique)

**A UML permite que você “desenhe” uma “planta” do seu sistema (ex. arquitetura ou do projeto técnico).**

- **Analogia com a engenharia civil:**

- É possível um construtor executar um projeto sem ter a planta (projeto básico) definida. Essa construção será feita com base nas experiências prévias do profissional. Em projetos pequenos e simples as chances de sucesso são grandes. Em projetos maiores e complexos, as chances de sucesso são menores.
- Cada projeto tem um contexto (ambiente, terreno, licenciamentos, regulamentação específica que varia de acordo com a área e tempo). O estudo prévio adequado pode direcionar o projeto para o caminho adequado.
- A elaboração do projeto técnico realiza o “cálculo estrutural”, o desenho da planta, aprova a estrutura, elevando as possibilidades de sucesso do projeto e do produto resultante.

**A experiência do desenvolvedor ou analista não pode substituir a necessidade de um projeto que defina uma “planta” da solução.**

- Esta “planta” garante, em todas as fases do projeto (definição, desenvolvimento, homologação, distribuição, utilização e manutenção ), uma maior clareza e objetividade para execução de cada ação, e, com certeza, quanto maior a solução, maior a necessidade de um projeto técnico.
- A UML independe da tecnologia em que o sistema será desenvolvido.
- A ideia é prover uma visão lógica do projeto de forma a facilitar sua implementação física (construção do software).

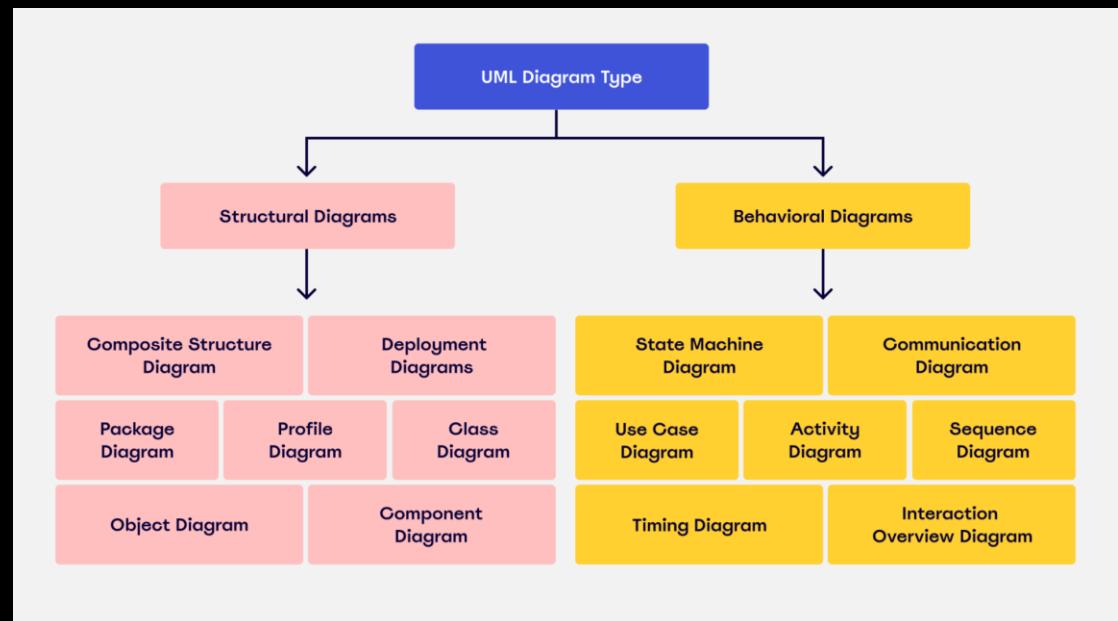
# UML - Diagramas

## Diagramas estruturais:

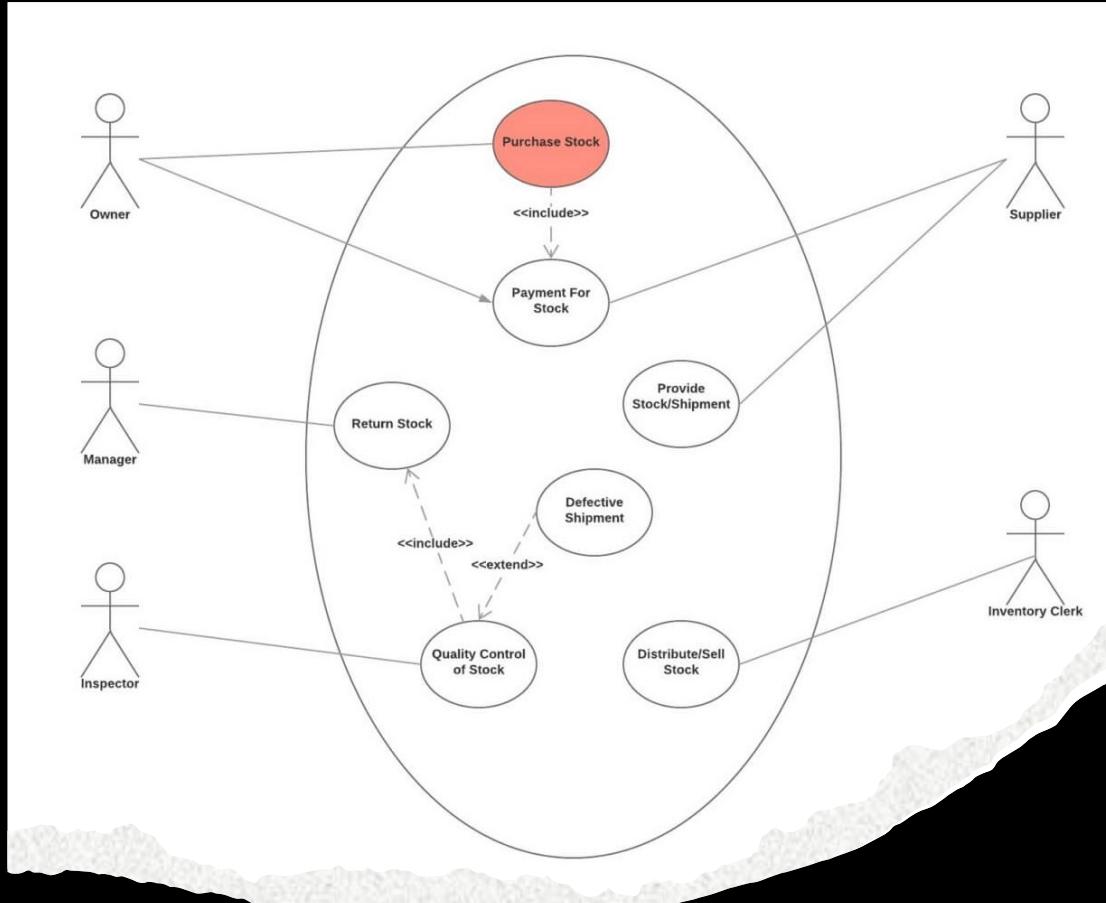
- Representam as estruturas estáticas do sistema. Representa as características das estruturas (classes, componentes, módulos) e seus relacionamentos.

## Diagramas comportamentais:

- Representam como o sistema se comporta quanto está em execução. São modeladas a execução de software em momento cenários particulares de uso.



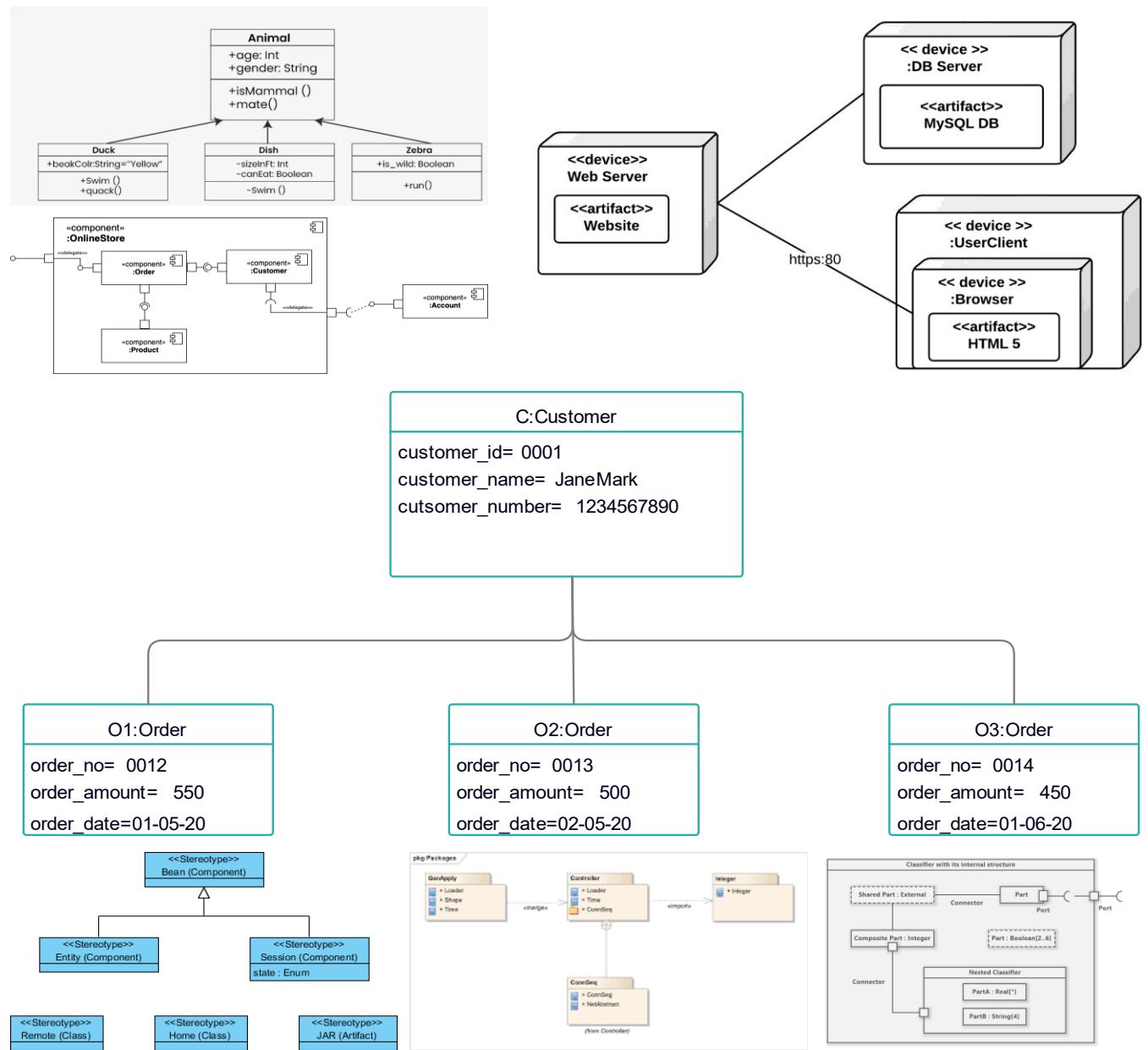
# UML diagrams

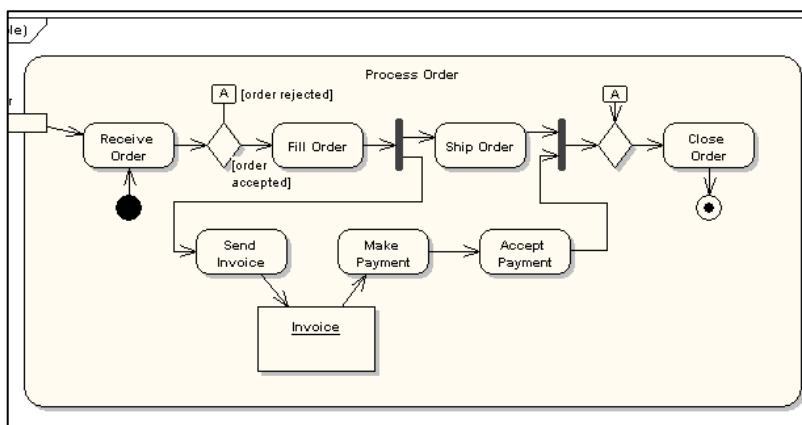
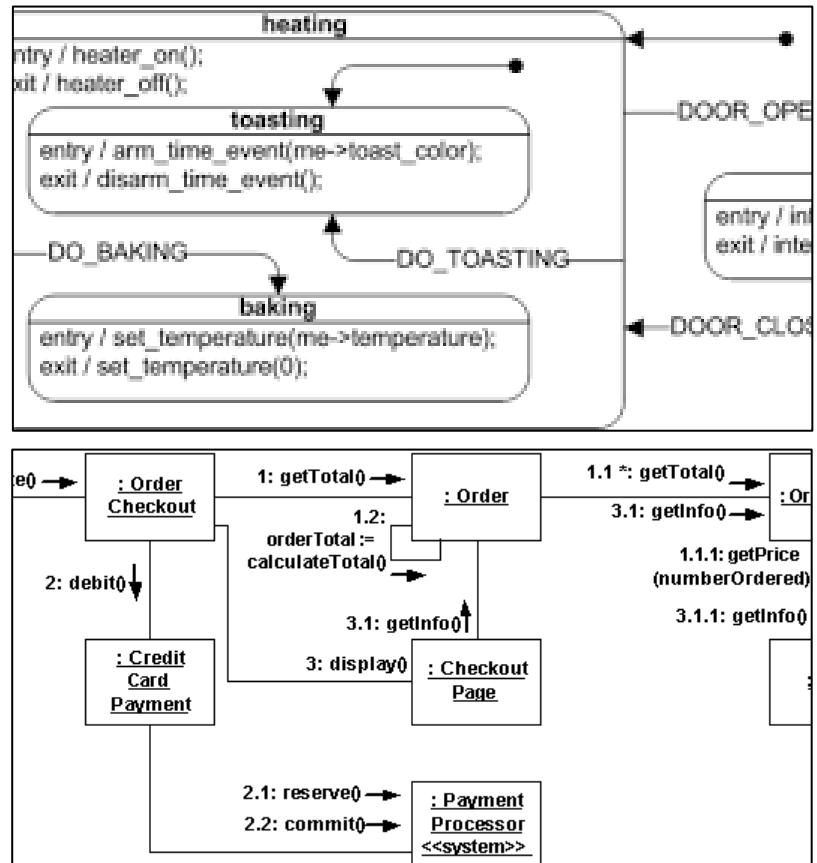
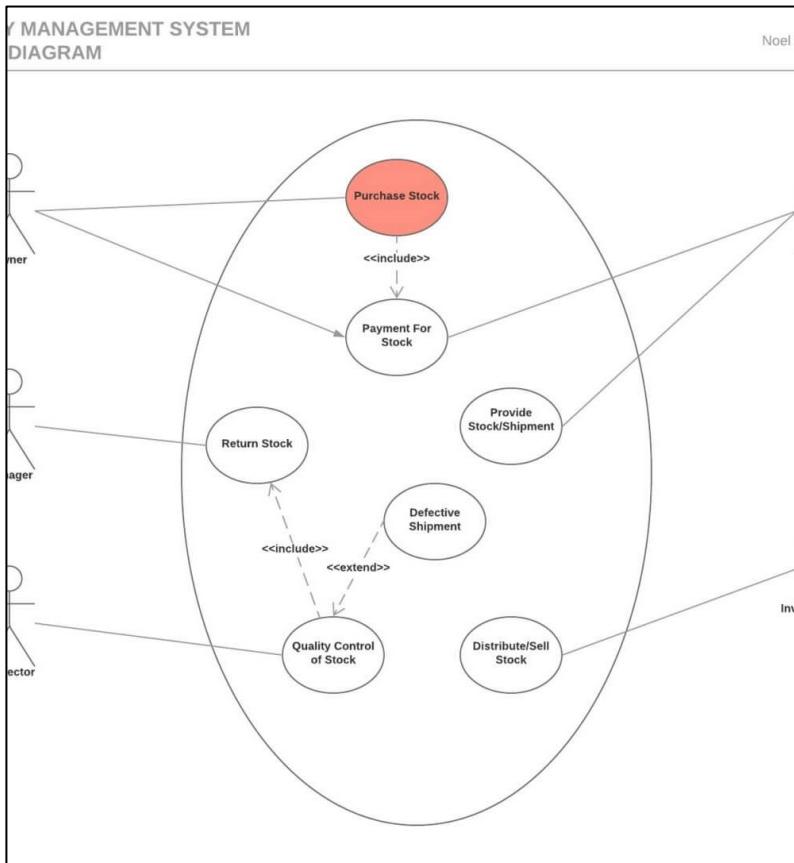
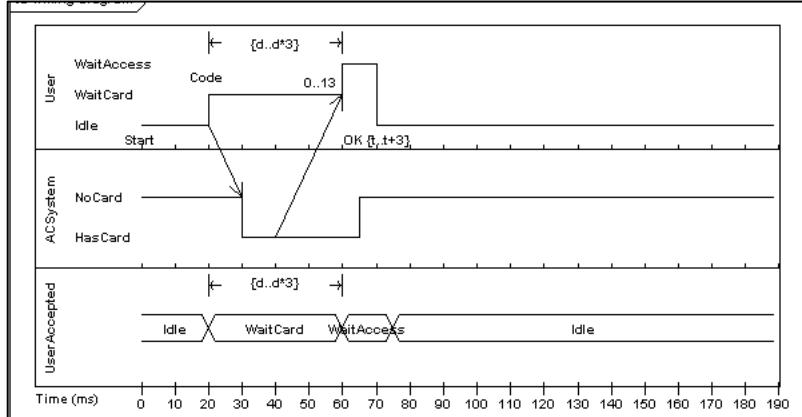
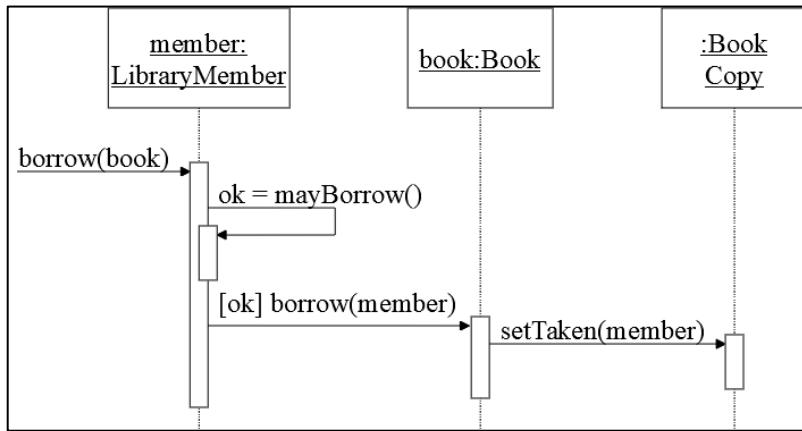


- Quais diagramas devo utilizar?
  - Não é necessário documentar o software por meio de todos os diagramas.
  - Dependendo da estrutura ou funcionalidade, existem diagramas mais adequados para realizar a sua respectiva modelagem.
- Devo modelar todo o sistema?
  - A estrutura do sistema costuma ser 100% modelada.
  - O comportamento do sistema costuma ter a modelagem dos principais casos de uso, pois dependendo da granularidade pode ser uma atividade custosa.

# Diagramas estruturais

- <https://www.lucidchart.com/pages/pt/diagrama-de-componentes-uml#>
- <https://www.lucidchart.com/pages/pt/o-que-e-diagrama-de-implementacao-uml>





# Ferramentas CASE: Computer-Aided Software Engineering





**StarUML**



**ENTERPRISE  
ARCHITECT**

**astah UML**

CASE: Computer-Aided Software Engineering

# Exemplo

- O objetivo da modelagem de dados em uma ferramenta CASE é:

A

especificar os requisitos funcionais e não funcionais de um projeto.

B

definir interfaces do usuário em um sistema.

C

criar diagramas de fluxo de controle.



representar a estrutura e as relações dos dados do sistema.



**Análise estruturada:**  
Diagrama de Fluxo de Dados (DFD)  
[https://pt.wikipedia.org/wiki/An%C3%A1lise\\_estruturada](https://pt.wikipedia.org/wiki/An%C3%A1lise_estruturada)



# UML - Questões

# Questão UML

- José é técnico em laboratório e foi pesquisar sobre o emprego da UML 2.0 em alguns modelos utilizados pelo analista de tecnologia da informação. Dos diagramas utilizados nos modelos pesquisados aquele que NÃO é um diagrama da UML 2.0 é:



Diagrama Hierárquico de Fluxo.

B    Diagrama de Classes.

C    Diagrama de Casos de Uso.

D    Diagrama de Sequência.

E    Diagrama de Atividade.

# Questão UML

A Linguagem de Modelagem Unificada (UML) é composta por vários diagramas, que têm por objetivo fornecer visões do sistema a ser modelado. “Existem dois diagramas que são associados diretamente à linguagem de programação, tendo como objetivo mostrar a organização do próprio código do projeto de software, escondendo detalhes de especificações através de seus artefatos de sistema e a dependência de seus relacionamentos.” Assinale-os.

A

De Classes e Casos de uso.



De Componentes e de Classes.

C

De Casos de Uso e de Sequência.

D

De Atividades e de Máquina de Estado.

# S.O.L.I.D: Os 5 princípios da POO.

Definidos por Robert Cecil Martin – um dos autores do manifesto ágil.

## **Single Responsibility Principle:**

- Uma classe deve representar uma única necessidade.

## **Open-Closed Principle:**

- Os objetos ou entidades devem estar abertos para extensão, mas fechados para modificação.

## **Liskov Substitution Principle:**

- Cada subclasse deve ser substituível pela classe sua classe base ou pai.

## **Interface Segregation Principle:**

- Um cliente nunca deve ser forçado a implementar uma interface que ele não usa.

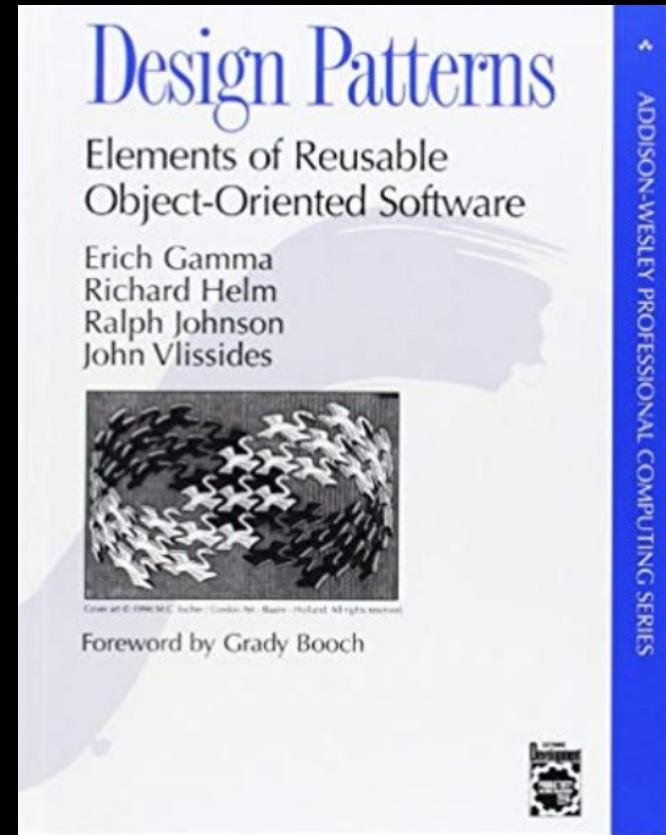
## **Dependency Inversion Principle:**

- As entidades devem depender de abstrações, não de implementações.

# GOF – Gang of four

## O que é um Padrão de Projeto?

- É a solução geral para um problema que ocorre com frequência dentro de um determinado contexto no projeto de software.
- Não é um projeto finalizado que pode ser diretamente transformado em código fonte ou de máquina, ele é uma descrição ou modelo (template) de como resolver um problema que pode ser usado em muitas situações diferentes.



# Os padrões de projeto (design patterns)

---

## Categorias de padrões:

- **Padrões criacionais:** abstraem e/ou adiam o processo criação dos objetos. Eles ajudam a tornar um sistema independente de como seus objetos são criados, compostos e representados.
- **Padrões estruturais:** Abordam a forma como classes e objetos são compostos para formar estruturas maiores. Os de **classes** utilizam a herança para compor interfaces ou implementações, e os de **objeto** descrevem maneiras de compor objetos para obter novas funcionalidades.
- **Padrões comportamentais:** incluem algoritmos e atribuições de responsabilidades entre os objetos. Eles não descrevem apenas padrões de objetos ou de classes, mas também os padrões de comunicação entre os objetos.

# GOF – Design Patterns

A documentação de cada padrão contém as informações:

- **Nome** do padrão;
- **Problema** a ser resolvido;
- **Solução** dada pelo padrão; e
- **Consequências**.

		Purpose		
		Creational	Structural	Behavioral
Scope	Class	Factory Method	Adapter	Interpreter Template Method
	Object	Abstract Factory Builder Prototype Singleton	Adapter Bridge Composite Decorator Facade Proxy	Chain of Responsibility Command Iterator Mediator Memento Flyweight Observer State Strategy Visitor

# Criacional: Singleton

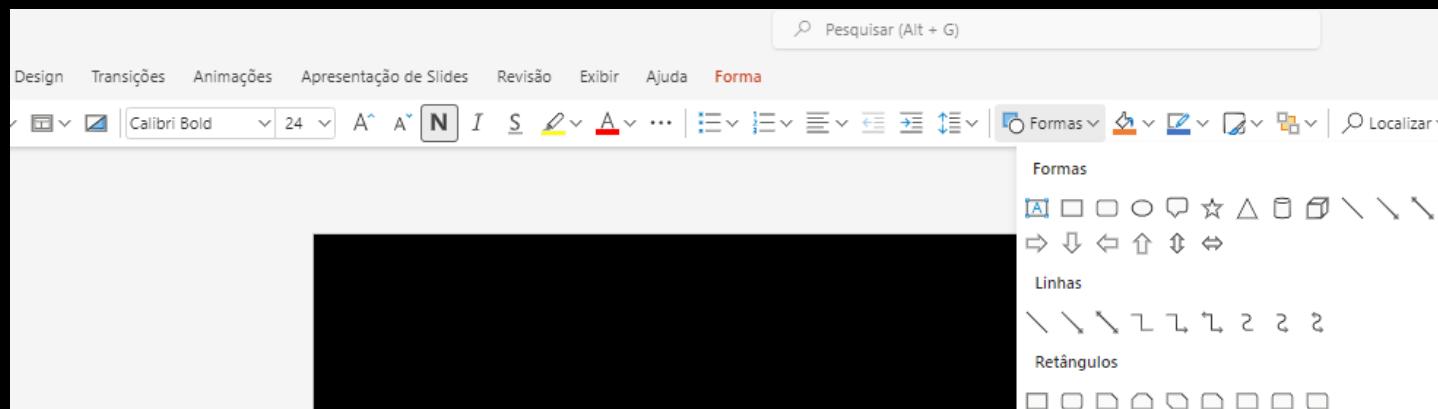
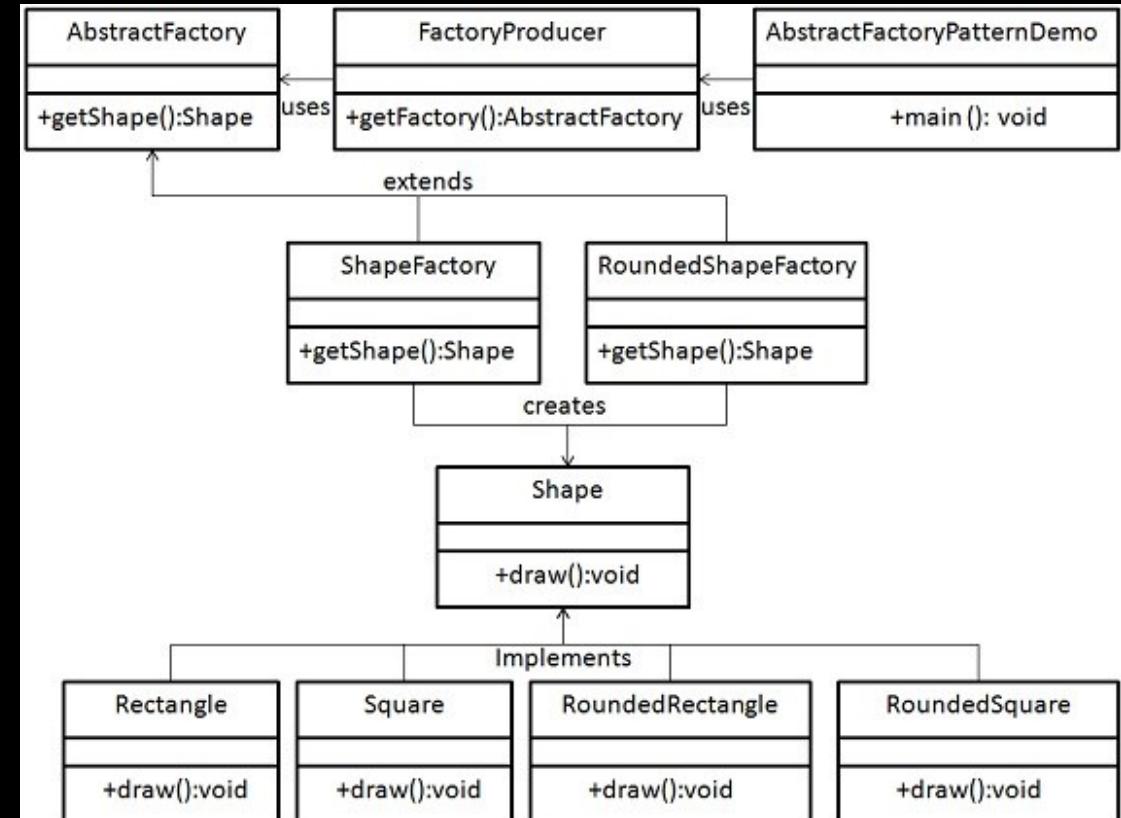
Objetivo:

*Singleton pattern restricts the instantiation of a class and ensures that only one instance of the class exists.*

<b>Singleton</b>
- <u>singleton : Singleton</u>
- <u>Singleton()</u>
+ <u>getInstance() : Singleton</u>

# Criacional: Abstract factory

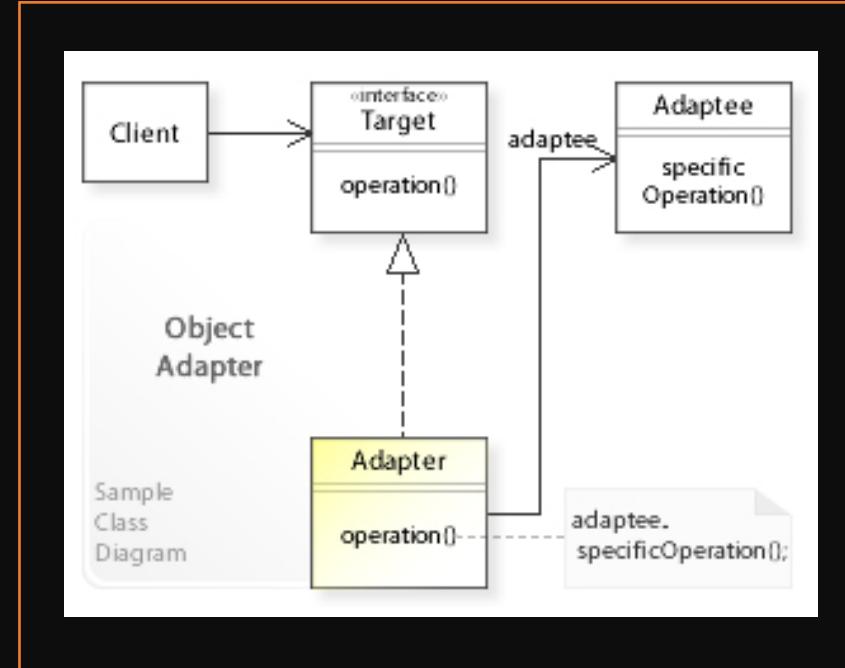
**Abstract Factory** is a creational design pattern that lets you produce families of related objects without specifying their concrete classes.



Neste exemplo, novos exemplos de *shapes* podem ser adicionadas, sem alterar demais funcionalidades genéricas (ex. Pintar fundo ou definir bordas de contorno).

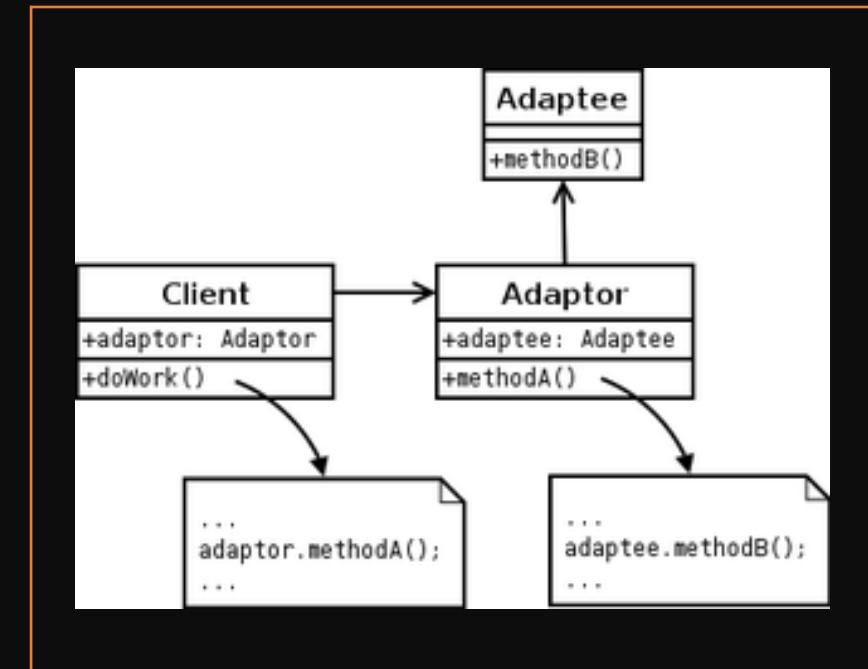
# Estrutural: Adapter

O **Adapter** é um padrão de projeto que permite objetos com interfaces incompatíveis colaborarem entre si.



*In the presented UML class diagram, the client class that requires a target interface cannot reuse the adaptee class directly because its interface doesn't conform to the target interface. Instead, the client works through an adapter class that implements the target interface in terms of adaptee.*

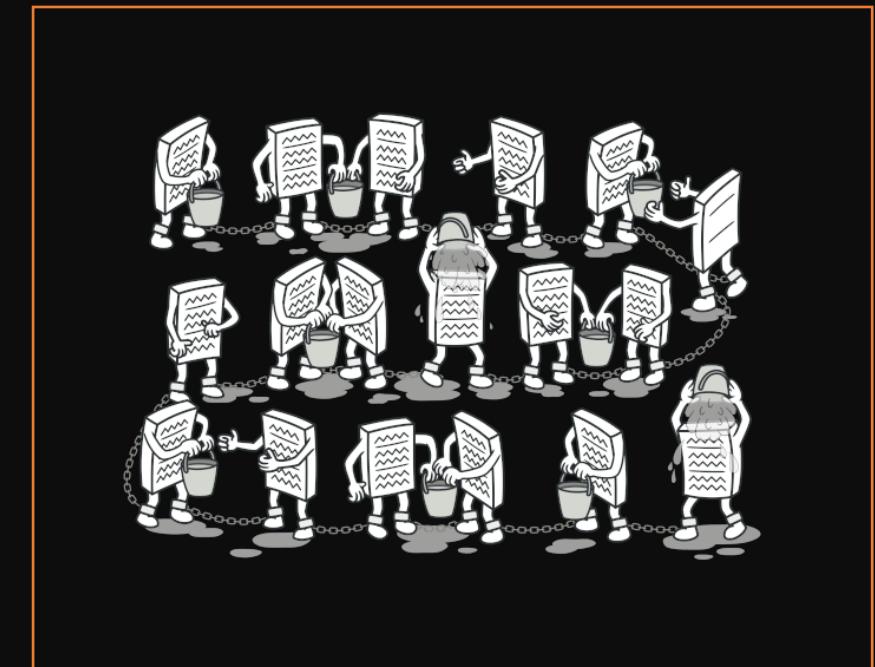
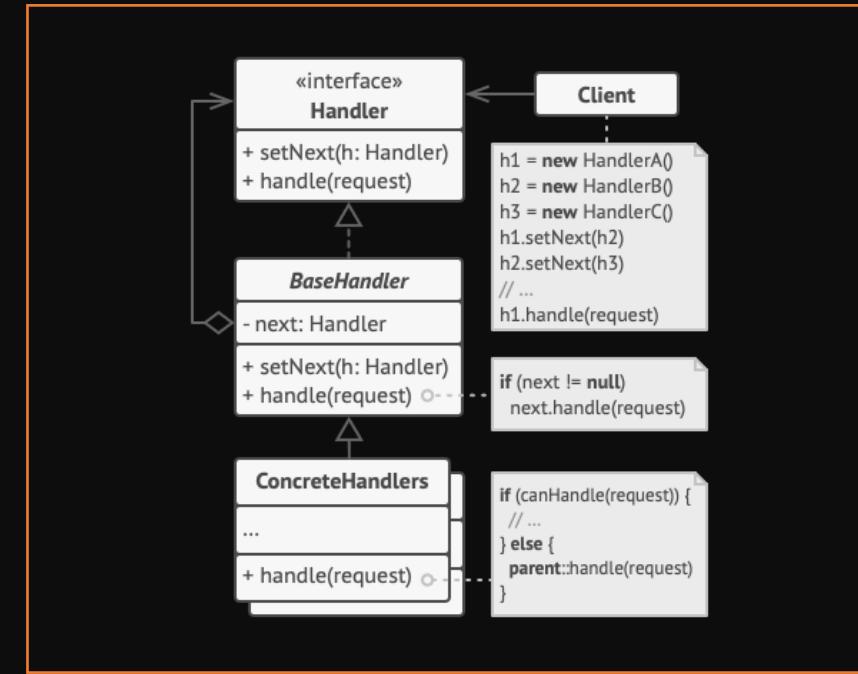
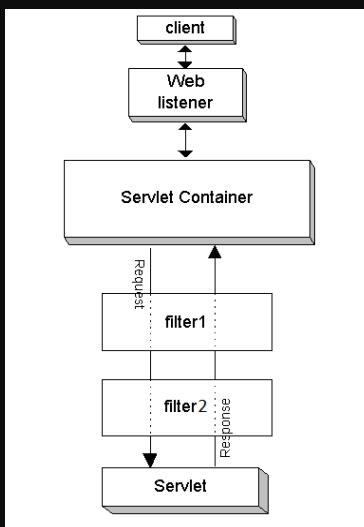
Exemplo: Há um método que converte conteúdo HTML em PDF. Porém tenho apenas a URL que redireciona para o conteúdo HTML que necessito gerar o PDF.

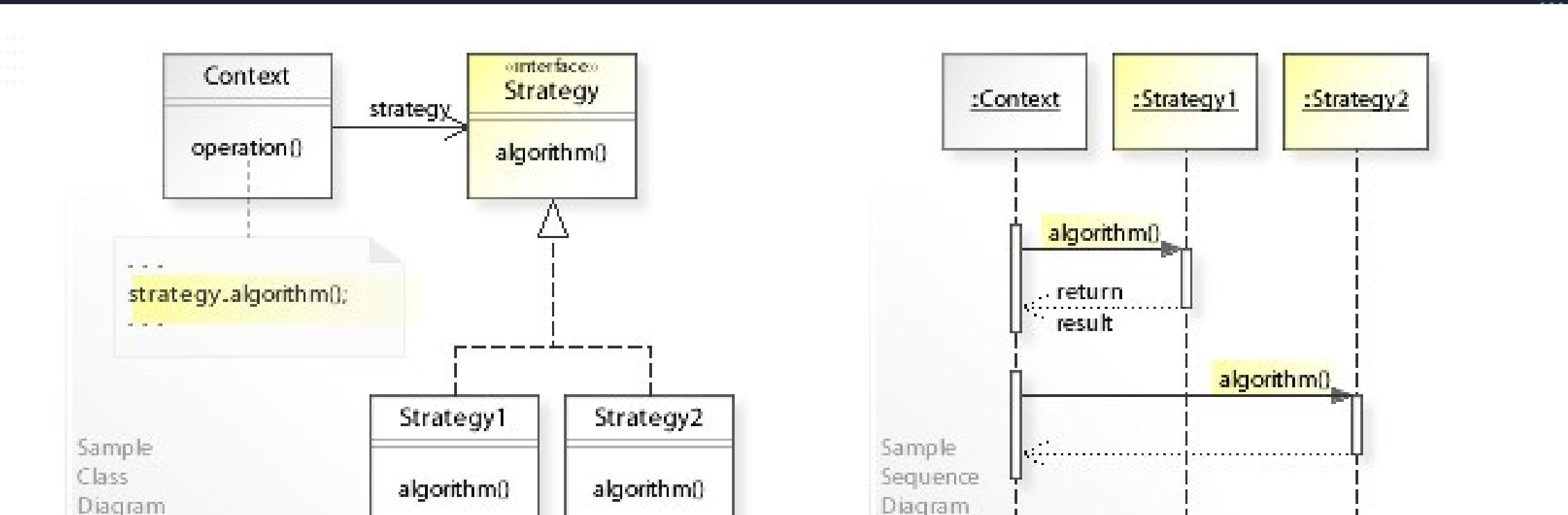


# Comportamental: chain of responsibility

**Chain of Responsibility** is a behavioral design pattern that lets you pass requests along a chain of handlers. Upon receiving a request, each handler decides either to process the request or to pass it to the next handler in the chain.

Exemplo: JEE filters (Filter chain)



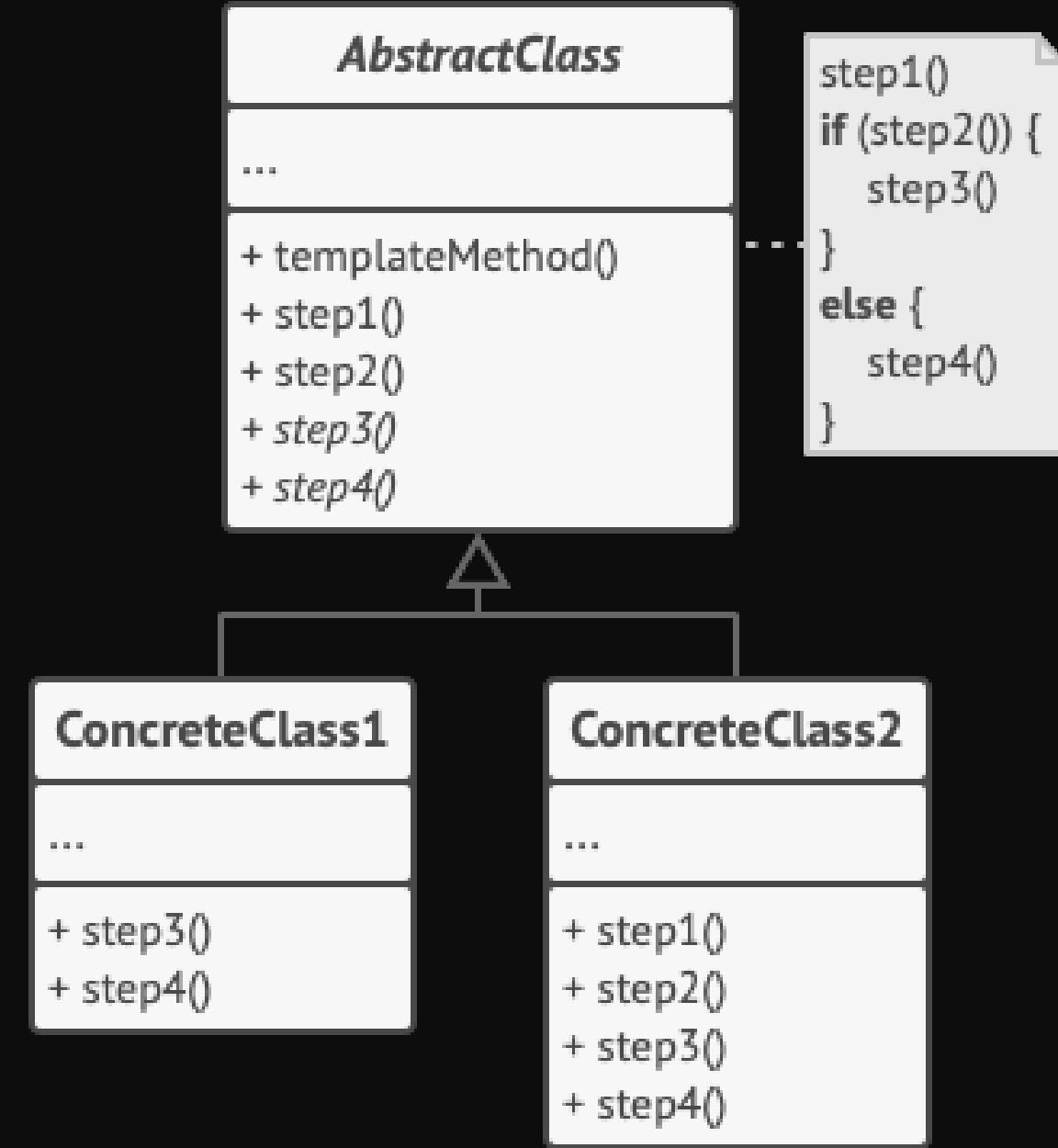


# Comportamental: Strategy

## Exemplo:

Se o dispositivo do usuário estiver online, atribuir a dependência que implementa o algoritmo que obtêm os dados por web service, e se estiver offline atribuir a dependência que implementa o algoritmo que realiza conjunto no banco de dados local.

**Propósito:** O Strategy é um padrão de projeto comportamental que permite que você defina uma família de algoritmos, coloque-os em classes separadas, e faça os objetos deles intercambiáveis.



## Comportamental: Template method

- O Template Method é um padrão de projeto comportamental que define o esqueleto de um algoritmo na superclasse mas deixa as subclasses sobrescreverem etapas específicas do algoritmo sem modificar sua estrutura.

# Padrões de projeto

## Exercícios



# Exemplo

Sobre padrões de projetos descritos por Gamma (2000), devemos programar para uma interface e não para uma implementação. Considerando os objetivos principais de alguns desses padrões, avalie as afirmações:

- I. Não declare variáveis como instâncias de classes concretas específicas. Em vez disso, prenda-se somente a uma interface definida por uma classe abstrata.
- II. Os padrões de criação permitem instanciar classes concretas (isto é, especificar uma particular implementação) em algum lugar do seu sistema.
- III. Ao abstrair o processo de criação de objetos, estes padrões lhe dão diferentes maneiras de associar uma interface com sua implementação de forma transparente no momento da instanciação.

Está CORRETO o que se afirma em:

É no processo de criação  
que se têm alguma  
**estrutura de decisão**,  
quanto a qual  
implementação concreta  
atribuir.

A

Nenhuma das afirmações é verdadeira.

B

I e II, apenas.

C

II e III, apenas.

D

I e III, apenas.



I, II e III.

# Exemplo

- Assinale a alternativa que apresenta um dos padrões de projeto GoF ("Gang of Four") classificado como estrutural.

A Abstract Factory.

B Command.

C Strategy.

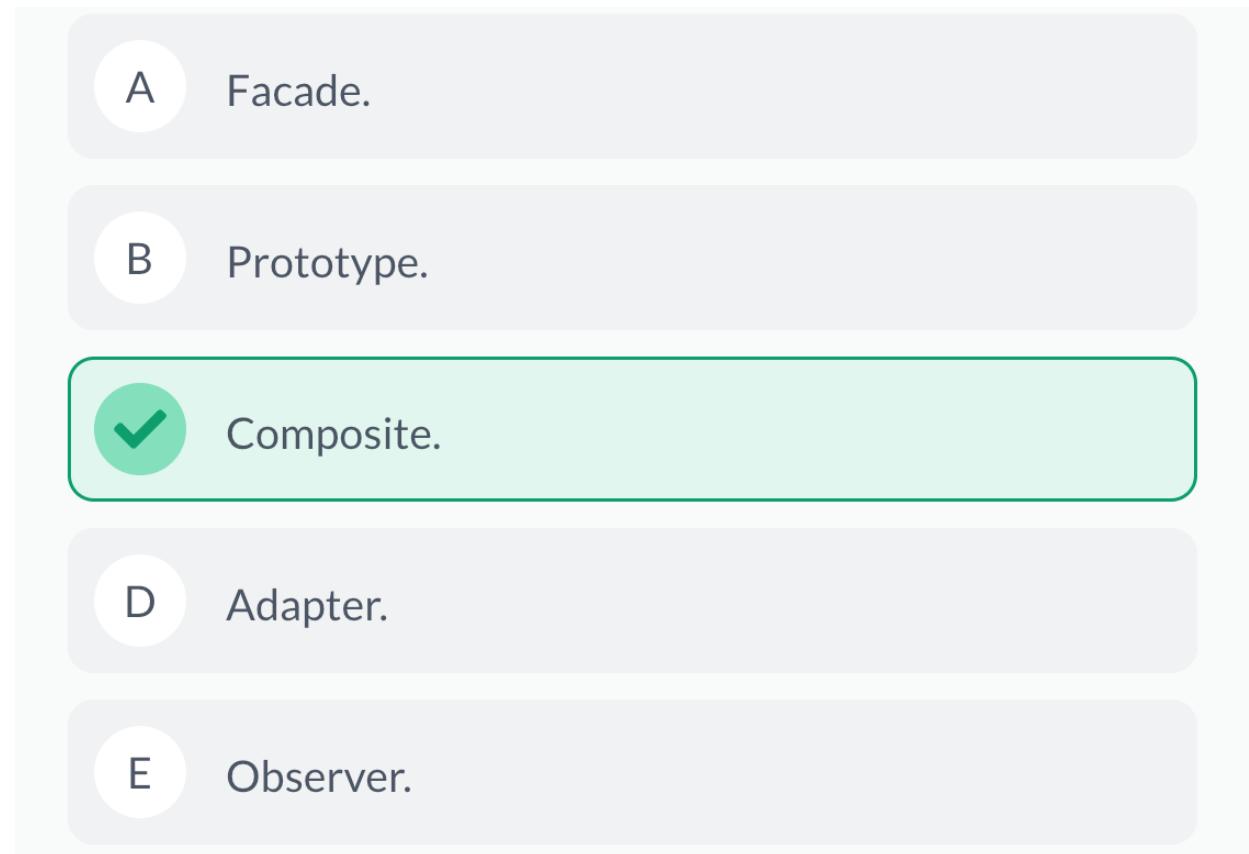


D Decorator.

E Observer.

# Exemplo

- Os padrões de projeto GoF ("Gang of Four") auxiliam os projetistas de software fornecendo soluções para problemas comuns, utilizando os conceitos da orientação a objetos. O padrão que permite a construção de estruturas complexas a partir de uma hierarquia de classes, compondo objetos em estruturas de árvore, é o:



# Exemplo

Qual padrão de projeto do projeto de software do Gang of Four (GoF, 1994) é adotado na implementação?

A Strategy.

 Template Method

C Adapter

D Composite

E Observer

PreparoBebida.java

```
1 public abstract class PreparoBebida {  
2       
3     public final void prepararBebida() {  
4         aquecerAgua();  
5         colocarIngrediente();  
6         misturarBebida();  
7         servirBebida();  
8     }  
9       
10    public abstract void colocarIngrediente();  
11      
12    public void aquecerAgua() {  
13        System.out.println("Aquecendo água...");  
14    }  
15    public void misturarBebida() {  
16        System.out.println("Misturando a bebida...");  
17    }  
18    public void servirBebida() {  
19        System.out.println("Servindo a bebida!");  
20    }  
21 }
```

PreparoChá.java

```
1 public class PreparoChá extends PreparoBebida {  
2     @Override  
3     public void colocarIngrediente() {  
4         System.out.println("Adicionando o chá na água...");  
5     }  
6 }
```

PreparoCafé.java

```
1 public class PreparoCafé extends PreparoBebida {  
2     @Override  
3     public void colocarIngrediente() {  
4         System.out.println("Colocando o pó de café na água...");  
5     }  
6     @Override  
7     public void misturarBebida() {  
8         System.out.println("Misturando...");  
9     }  
10 }
```

Bebida.java

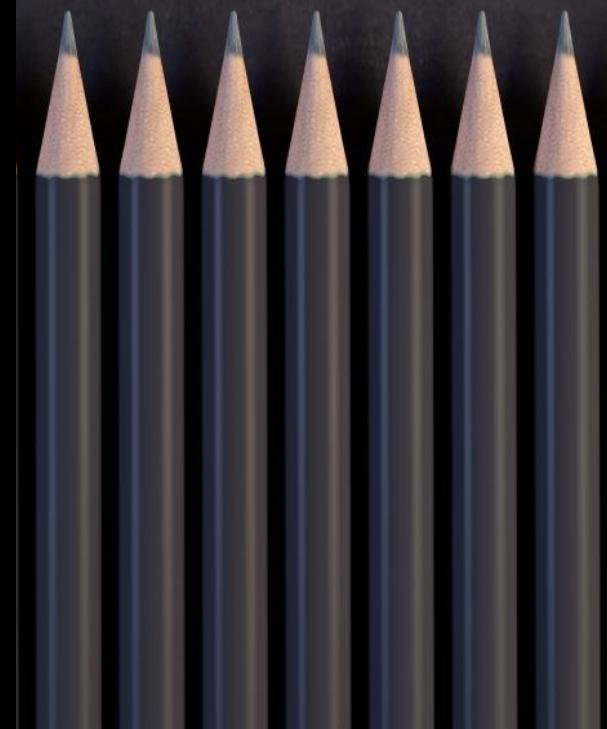
```
1 public class Bebida {  
2     public static void main(String[] args) {  
3         PreparoBebida preparoChá = new  
4             PreparoChá();  
5         preparoChá.prepararBebida();  
6         PreparoBebida preparoCafé = new  
7             PreparoCafé();  
8         preparoCafé.prepararBebida();  
9     }  
10 }
```

# Atividade

- Como parte da entrega da M1 (orientações gerais no slide 6), iniciar a produção da arquitetura candidata do trabalho da equipe:
- O documento de arquitetura candidata será composto, dentre outros, dos seguintes artefatos:
  - Diagrama de classes
  - Diagrama de sequência para o fluxo principal de cada caso de uso. (o trabalho é composto de 3 casos de uso).
  - Aplicar no mínimo 3 padrões de projeto GOF (sendo 1 de cada categoria).

# Aula 3

## RUP



# RUP

---

- **RUP (Rational Unified Process)**, Processo Unificado Rational foi criado pela Rational Software Corporation, mas em 2003 foi adquirida pela IBM.
- A metodologia RUP utiliza uma abordagem de **orientação a objetos** em sua concepção e é projetado e documentado utilizando o **UML** para ilustrar os processos.
- Tem como principais características ser **incremental** e **iterativo**.
  - **Iterativo**: indica que o processo é repetido mais de uma vez, caracterizando ciclos de execução.
  - **Incremental**: significa que aquele software é construído e entregue em pedaços, constituindo um conjunto de funcionalidades completas.
- Através de pequenos ciclos de projetos - que correspondem a uma iteração - o software é melhorado através da adição de mais detalhes, o que resulta em um incremento no software..



# Enquadramento do RUP

O RUP é uma metodologia com práticas ágeis, assim como Scrum e o *Extreme Programming (XP)*.

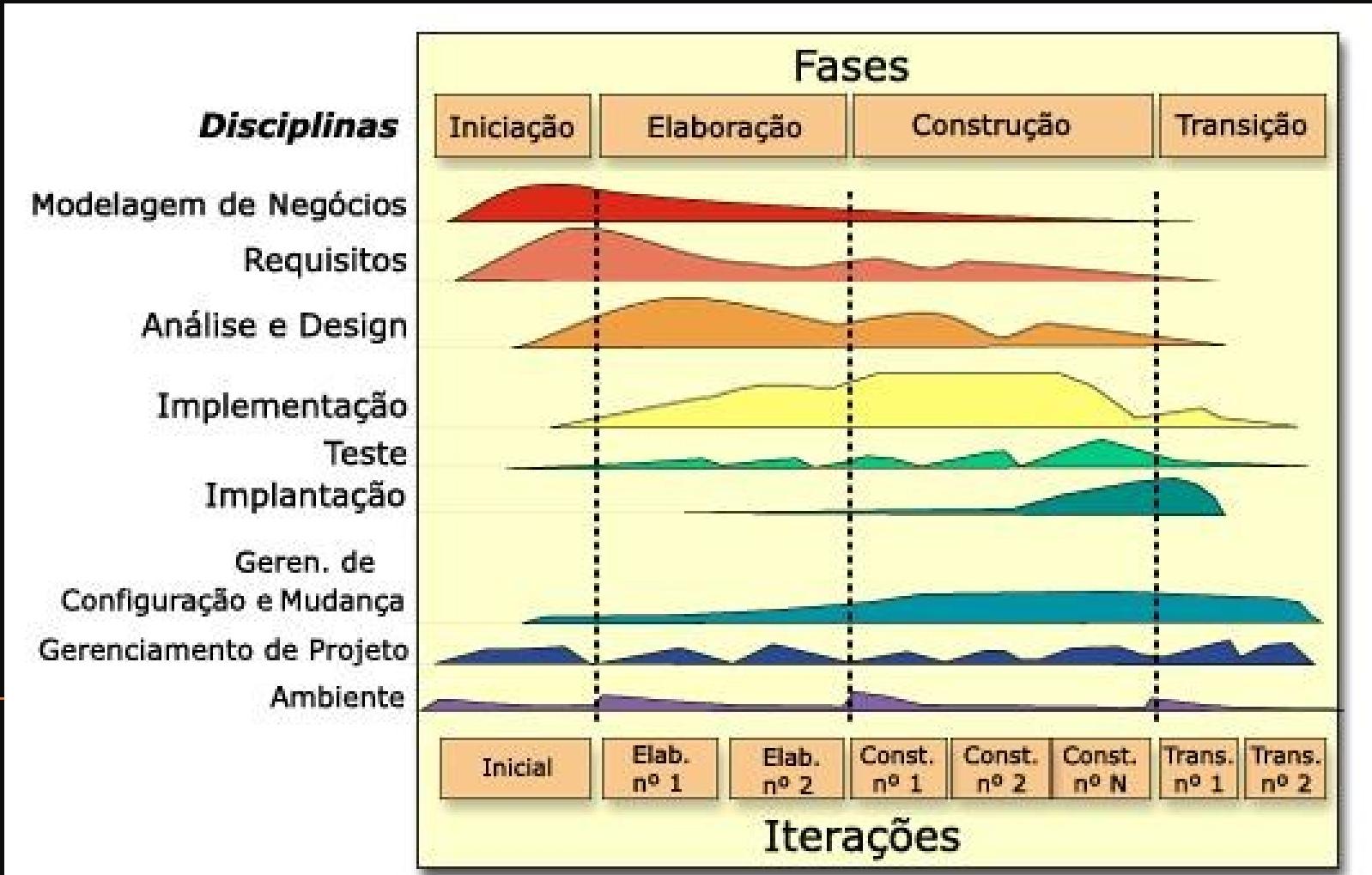
Os métodos ágeis adotam os valores e os princípios citados no manifesto ágil e exemplos desses processos são **Scrum**, **Extreme Programming (XP)**, **Lean Development**, **Feature Driven Development (FDD)**, **Kanban** e **Crystal**.

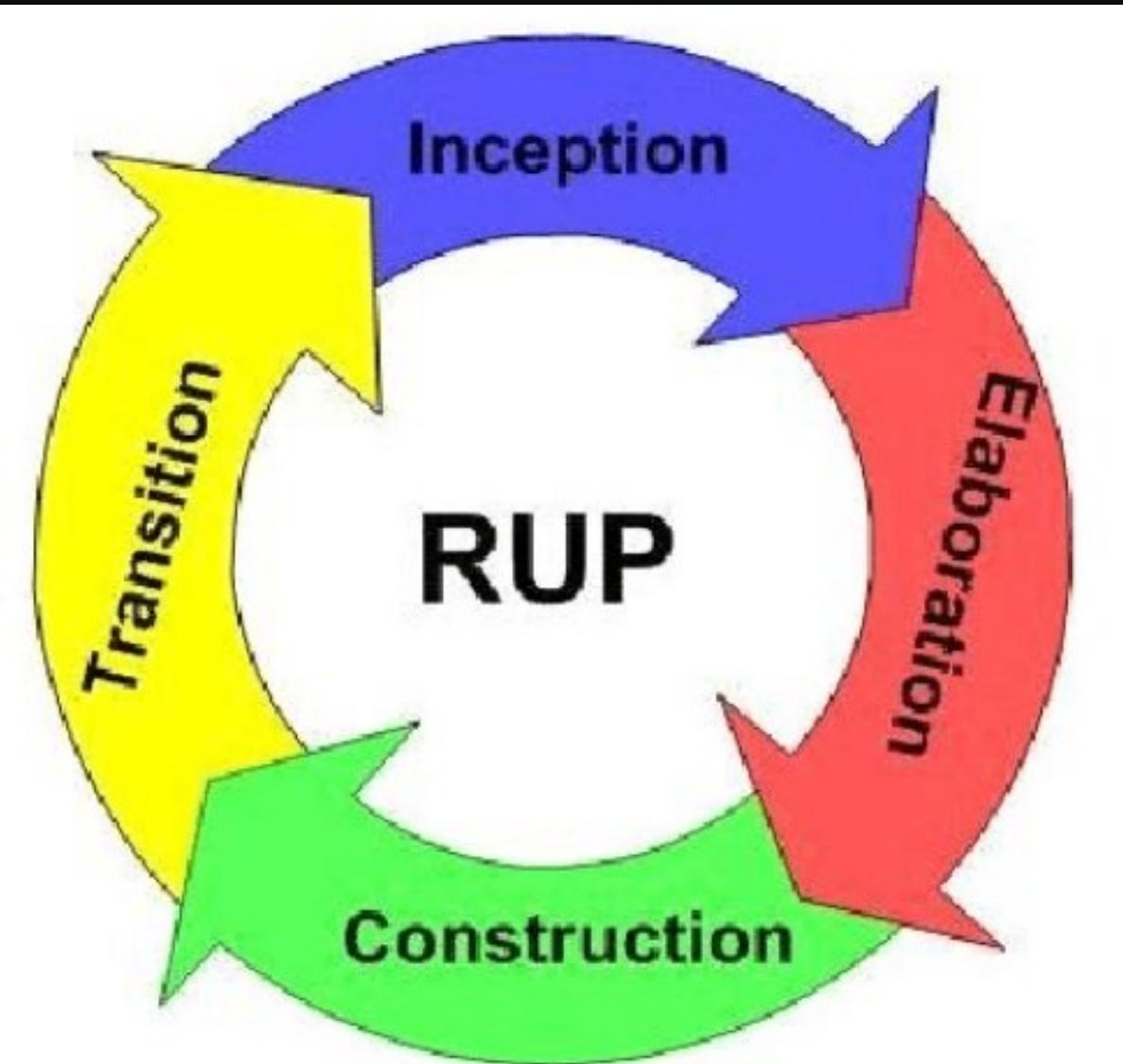
- Todos estas práticas seguem o modelo iterativo e incremental.
- As metodologias ágeis procuram utilizar iterações curtas, tempo estabelecido e despreza o uso excessivo de documentação.

O RUP é um framework de processo da engenharia de software que fornece práticas testadas na indústria de software de gerência de projetos.

- Preferencialmente, utilizado para projetos complexos com equipes grandes.
- Por ser um framework de processo, pode ser customizado conforme as necessidades organizacionais e do projeto, dessa forma, pode-se trabalhar um RUP mais “leve e ágil” ou “mais pesado”.
- O RUP permite que a equipe do projeto escolha as atividades e os **artefatos** para serem produzidos, reduzindo assim, o excesso de documentação para torná-lo mais ágil.

# RUP – Fases, disciplinas e iterações





## RUP – Fases do processo RUP

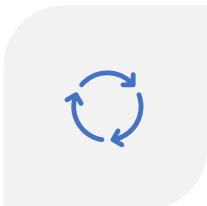
---

As iterações ocorrem dentro de cada fase, e também após completar o ciclo de fases.

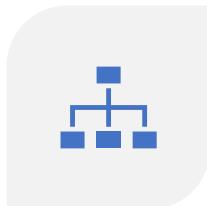
---

Source: Lalitha, Priya & C S, Lifna & Jagli, Dhanamma & Joy, Anooja. (2014). "Rational Unified Treatment for Web Application Vulnerability Assessment". 10.1109/CSCITA.2014.6839283.

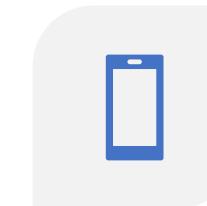
# Práticas reforçadas pelo processo RUP



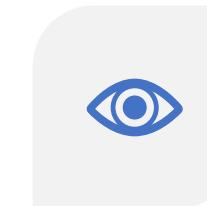
DESENVOLVER O  
SISTEMA  
ITERATIVAMENTE;



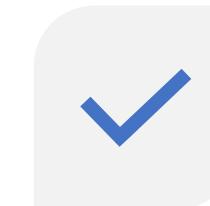
GERENCIAR  
REQUISITOS;



ARQUITETURAS  
BASEADAS EM  
COMPONENTES;



MODELAR O  
SOFTWARE  
VISUALMENTE;

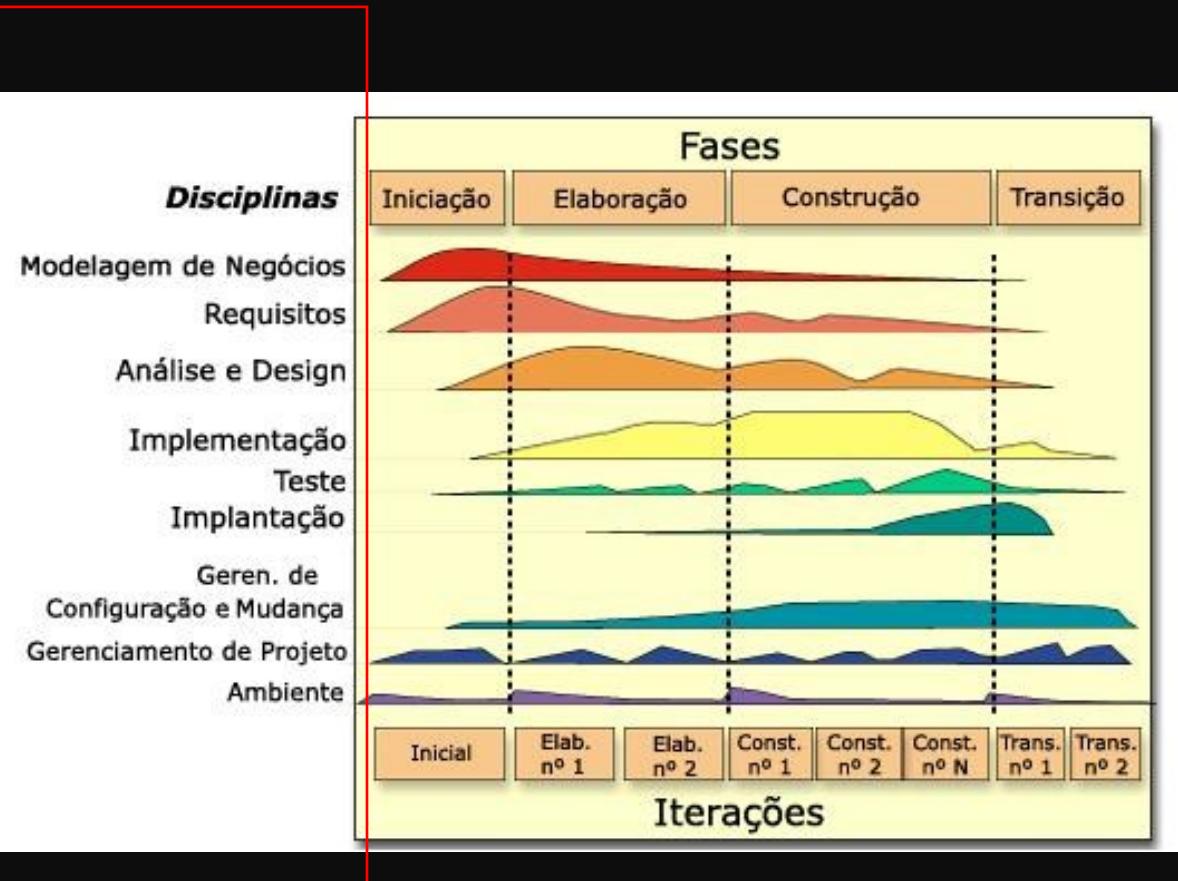


VERIFICAR A  
QUALIDADE DO  
SOFTWARE;



CONTROLAR AS  
MUDANÇAS DO  
SOFTWARE.

# Workflows (disciplinas) do RUP



As disciplinas são definidos da seguinte maneira:

- **Modelagem de negócios:** os processos de negócios são modelados com a utilização de casos de uso de negócio;
- **Requisitos:** os casos de usos são criados para modelar os requisitos do software;
- **Análise e projeto: Implementação:** os componentes do software são implementados;
- **Teste:** o teste é um processo iterativo e é efetuado em conjunto com a implementação do sistema;
- **Implantação:** cria-se uma versão do produto e instala-a no local de trabalho dos usuários;

Disciplinas de apoio:

- **Gerenciamento de configuração e mudanças:** esse workflow serve como apoio à gerência do projeto em relação às mudanças no sistema;
- **Gerenciamento de projetos:** esse workflow de apoio gerencia o processo de desenvolvimento do software;
- **Ambiente:** este workflow relaciona-se à disponibilização de ferramentas de software adequadas para a equipe de desenvolvimento.

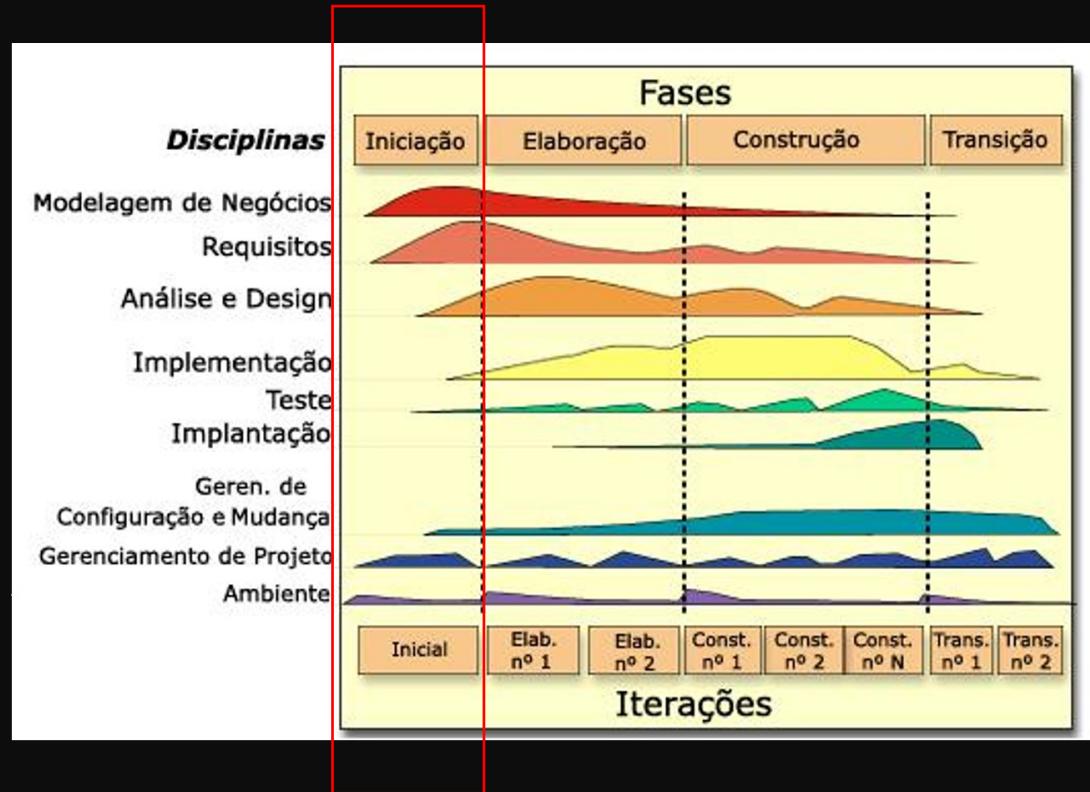
# Fase: Concepção ou iniciação

**Os objetivos principais dessa fase são:**

- Entender o que será construído (determinar objetivo, escopo e as partes interessadas).
- Identificar os casos de uso de maior relevância.
- Determinar uma solução possível (identificação de uma arquitetura candidata).
- Entender os custos, cronograma e riscos associados ao projeto. (estimativas)
- Determinar o processo a seguir e as ferramentas que serão utilizadas.

**Os principais artefatos associados a fase são:**

- Documento de visão (descrição em alto nível do software, incluindo benefícios, problemas a serem resolvidos, partes interessadas e alguns requisitos não funcionais chaves).
- Modelo de casos de uso (apenas os casos de usos críticos para o sistema).
- Arquitetura candidata do sistema (cobrindo os casos de usos críticos, estruturas de dados, e resultando em possíveis protótipos).
- Plano de Projeto, com todas as informações necessárias ao desenvolvimento do projeto (tempo, custo, escopo).



# Documento de visão

O documento de visão define:

- O propósito do projeto
- Uma instrução clara do problema
- A solução proposta
- O escopo de alto nível
- Uma estimativa de alto nível dos recursos
- Premissas e riscos
- Partes interessadas

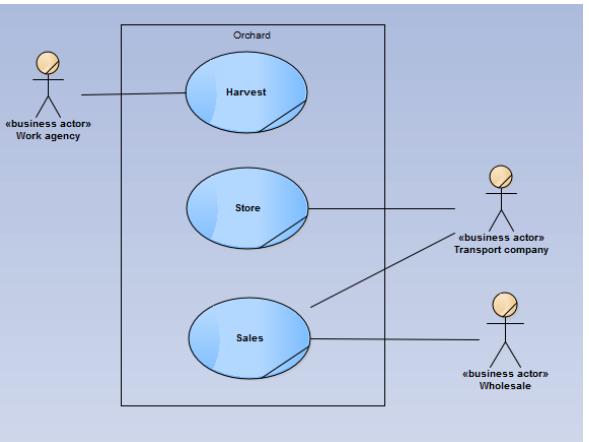
PROJECT CHARTER												
Project Title	Project and Portfolio Management Tool			Project Manager	Sameer Patel							
Project Start Date	May 21, 2017		Project End Date	August 31, 2017								
Business Need												
All Information Technology projects that require agreement on the Memorandum of Understanding between the Customer and the Service Provider are approved through email. This project was initiated to reduce the manual approvals and create a system to obtain and track the approvals to reduce any discrepancies and loss of data.												
Project Scope			Deliverables									
Create an in-house PPM to include all Global IT projects.			1. Generate consolidated project status report 2. Extract Global Headcount details for all projects									
Risks and Issues			Assumptions/Dependencies									
1. Data discrepancy due to large amount of projects 2. Involvement of multiple teams			1. All Global IT projects to be added to the tool 2. Managers to provide regular updates for the projects									
Financials												
Budget to complete this project is \$3000												
Milestones Schedule												
Milestone				Target Completion Date	Actual Date							
Upload all Global IT Projects to the tool				May 20, 2017								
Complete UAT testing for the tool				July 30, 2017								
Project Team				Approval/Review Committee								
Project Manager	Randy Hadden		Sponsor	Randy Hadden								
Project Manager	Sameer Patel		Business Division Head	Aniket Bhonsle								
Team Members	Vice President, Senior Manager, Analyst		Business Unit Head	Sunil Rajan								
			Finance Manager	Ketan Shah								

Na disciplina será adotado o Termo de Abertura do Projeto do PMBOK (Project Charter) como documento de visão.

Source:

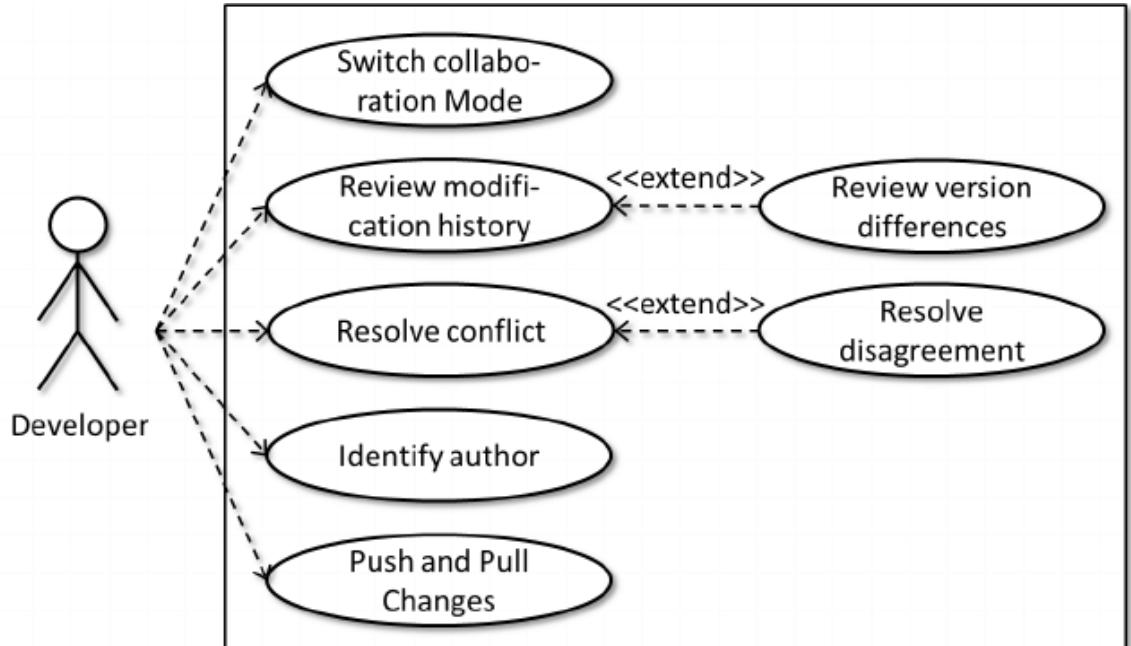
<https://www.ibm.com/docs/pt-br/engineering-lifecycle-management-suite/lifecycle-management/6.0.3?topic=requirements-vision-document>

Element	Detail
Name	The use case name. Typically the name is of the format <action> + <object>.
ID	An identifier that is unique to each Use Case.
Description	A brief description that states what the user wants to be able to do and what benefit he will derive.
Actors	The type of user who interacts with the system to accomplish a task. Actors are identified by role name.
Organization Benefits	The value the organization expects to receive from having the functionality described. Ideally this is a link directly to a Business Objective.
Frequency of Use	How often this Use Case is executed.
Triggers	Concrete actions made by the user within the system to start the Use Case.
Preconditions	Any states that the system must be in or conditions that must be met before the Use Case is started.
Postconditions	Any states that the system must be in or conditions that must be met after the Use Case is completed successfully. These will be met if the Main Course or Alternate Courses are followed. Some exceptions may result in failure to meet the Postconditions.
Main Course	The most common path of interactions between the user and the system. 1. Step 1 2. Step 2
Alternate Course	Alternate paths through the system. AC1:<condition for the alternate to be called> 1. Step 1 2. Step 2  AC2:<condition for the alternate to be called> 1. Step 1
Exception Courses	Exception handling by the system EX1:<condition for the exception to be called> 1. Step 1 2. Step 2  EX2:<condition for the exception to be called> 1. Step 1



# Modelo de caso de uso

- Um *caso de uso* é um artefato que define uma sequência de ações que gera um resultado de valor observável.
- Os casos de uso fornecem uma estrutura para expressar requisitos funcionais no contexto de processos de negócios e de sistema.
- Os casos de uso podem ser representados como um elemento gráfico em um diagrama e como uma especificação de caso de uso em um documento textual.



Source:

<https://www.ibm.com/docs/pt-br/engineering-lifecycle-management-suite/lifecycle-management/7.0.0?topic=requirements-defining-use-cases>

<https://www.ibm.com/docs/pt-br/engineering-lifecycle-management-suite/lifecycle-management/7.0.0?topic=cases-use-case-specification-outline>

<https://argondigital.com/blog/product-management/dont-forget-the-forgotten-use-cases-use-case-template-download/>

# Arquitetura do sistema

---

Produz o documento de arquitetura do sistema

- Modelar com diagramas UML
- A quantidade e tipos de diagramas varia de projeto para projeto

**No contexto da disciplina, o documento deve conter a seguinte modelagem:**

- Diagrama de deployment
- Diagramas de classes
- +1 diagrama estrutural a escolher.
- Diagrama de casos de uso
- Diagramas de sequência (1 por caso de uso detalhado – 3 ao total)
- +1 diagrama comportamental a escolher.

**Observação:**

- Implementar ao menos 3 padrões de projeto do GOF



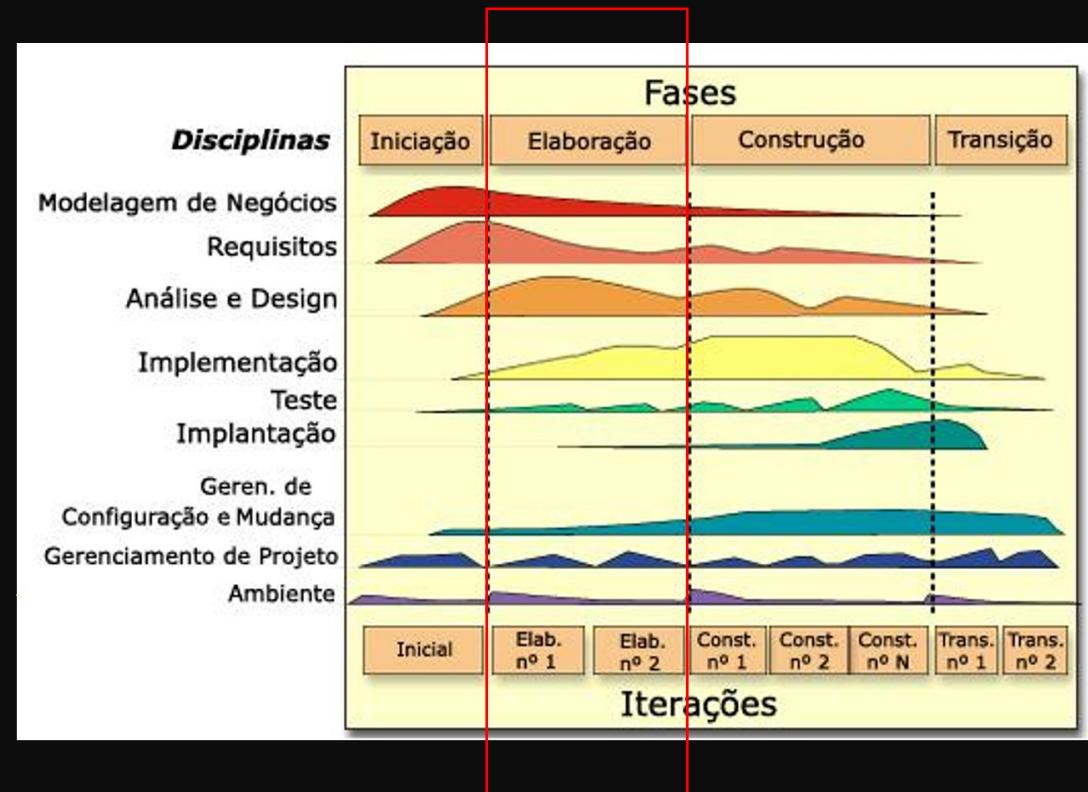
# Fase: Elaboração

**Os objetivos principais dessa fase são:**

- Obter um entendimento mais detalhado dos requisitos.
- Desenhar e validar a arquitetura do software.
- Implementação de protótipos funcionais com a arquitetura definida.
- Definir estimativas de cronograma e custos mais precisas com base na arquitetura definida, assim como a gestão de riscos do projeto.

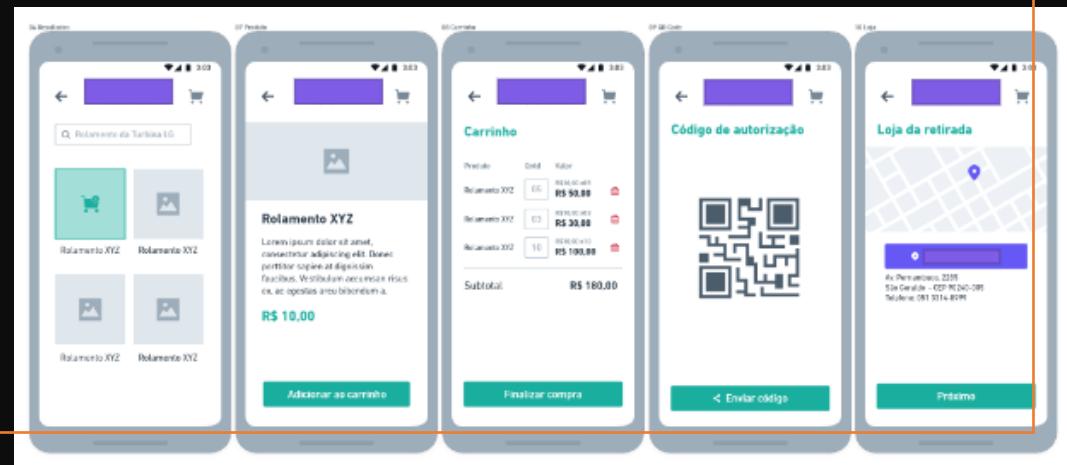
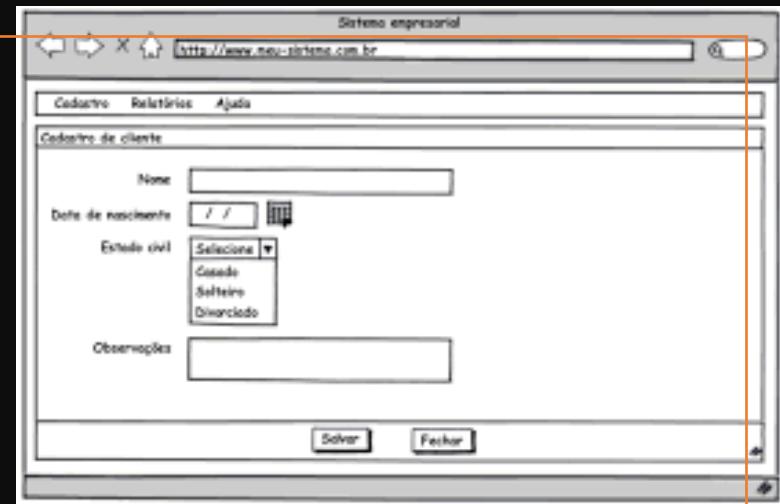
**Os principais artefatos associados a fase são:**

- Modelo de casos de uso detalhado (fluxos alternativos e de exceção detalhados).
- Protótipos (utilizados para validar requisitos e a arquitetura).
- Arquitetura executável.
- Plano de Projeto atualizado (com as novas estimativas de custos e prazos).

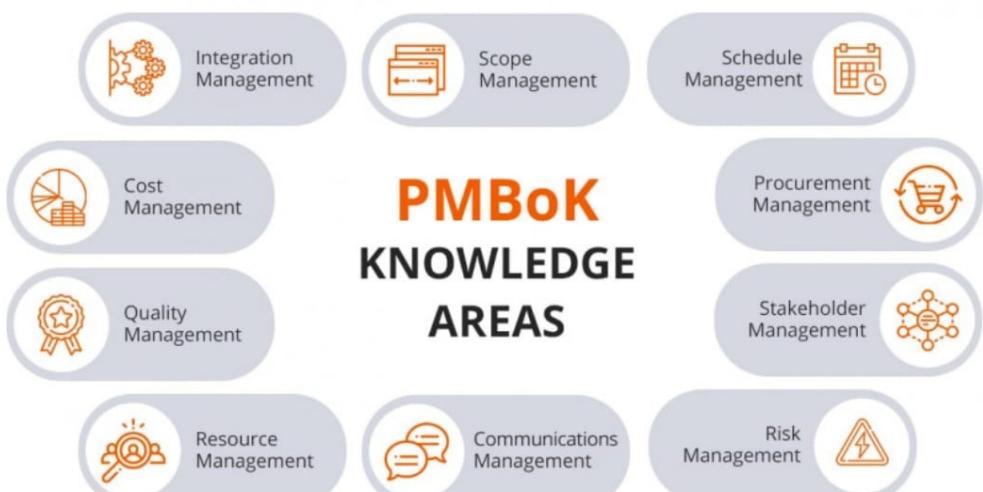


# Protótipos

- Protótipos de baixa fidelidade (concepção)
- Protótipos de alta fidelidade (elaboração)
- Recomenda-se o uso da tecnica prototipacao evolutiva



# Plano do projeto



Source:

<https://monday.com/blog/project-management/project-management-knowledge-areas/>  
<https://www.stakeholdermap.com/project-templates/pmbok-management-plan-templates.html>

Na disciplina será adotado o Plano de Projetos do Projeto do PMBOK.

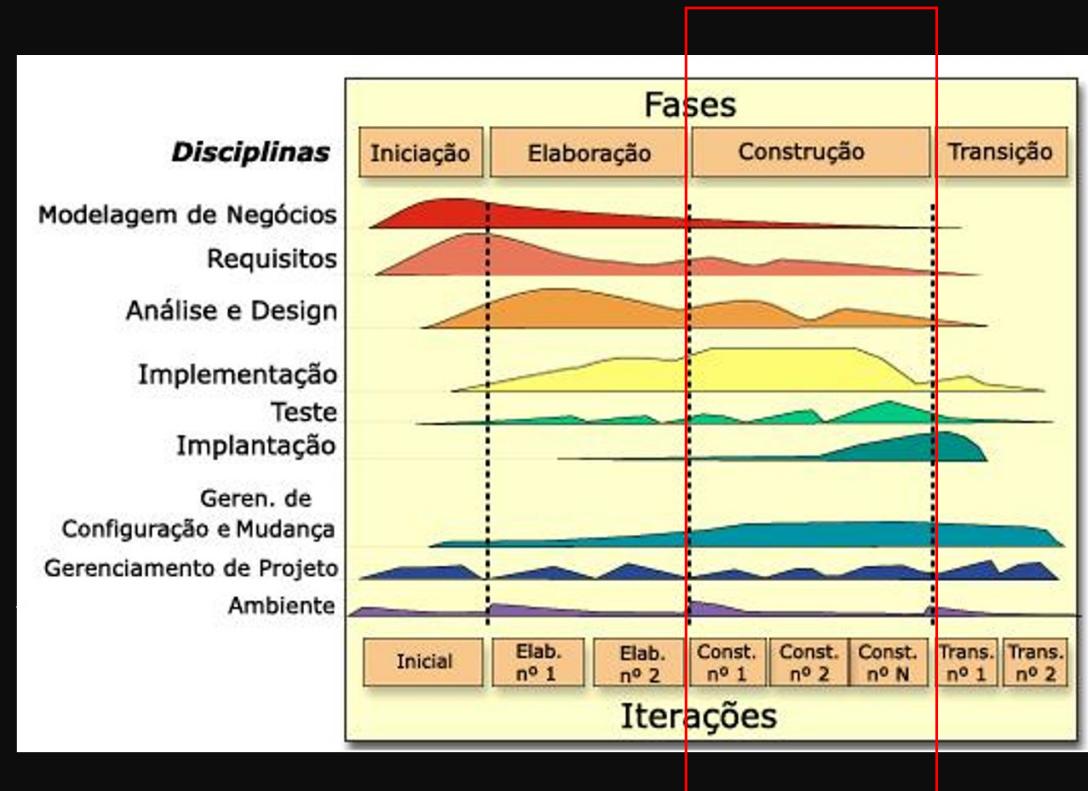
# Fase: Construção

Os objetivos principais dessa fase são:

- Minimizar os custos de desenvolvimento com algum grau de paralelismo; e
- Desenvolver iterativamente um produto que estará pronto para transição (*beta release*).

Os principais artefatos associados a fase são:

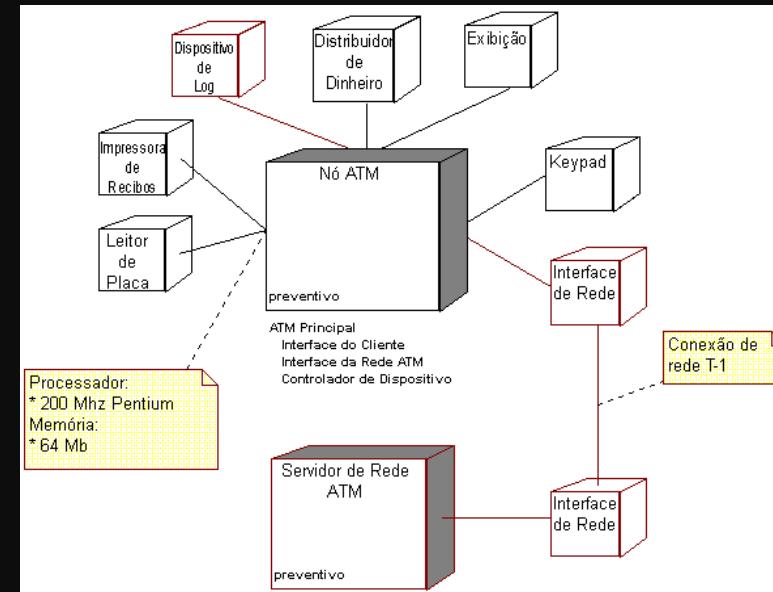
- o sistema (código)
- os planos de implantação; e
- os conjuntos de testes (casos de testes especificados)



# Plano de implantação

Todas as ações necessárias para colocar o software em ambiente operacional precisam estar documentadas. Além do diagrama de deployment, é necessário descrever itens, como:

- Preparação das máquinas (físicas ou virtuais, on premise, cloud).
- Instalação dos servidores (banco de dados, web, etc).
- Configuração do hardware e software (capacidades, versões, etc.)
- Criação dos usuários de sistema (banco de dados, servidores)
- Carga de dados inicial, parametrização do sistema em produção.
- Instalação do software e configurações de acesso (firewall, etc.)
- Criação de contas para os usuários, integração com „Ad“ de produção.



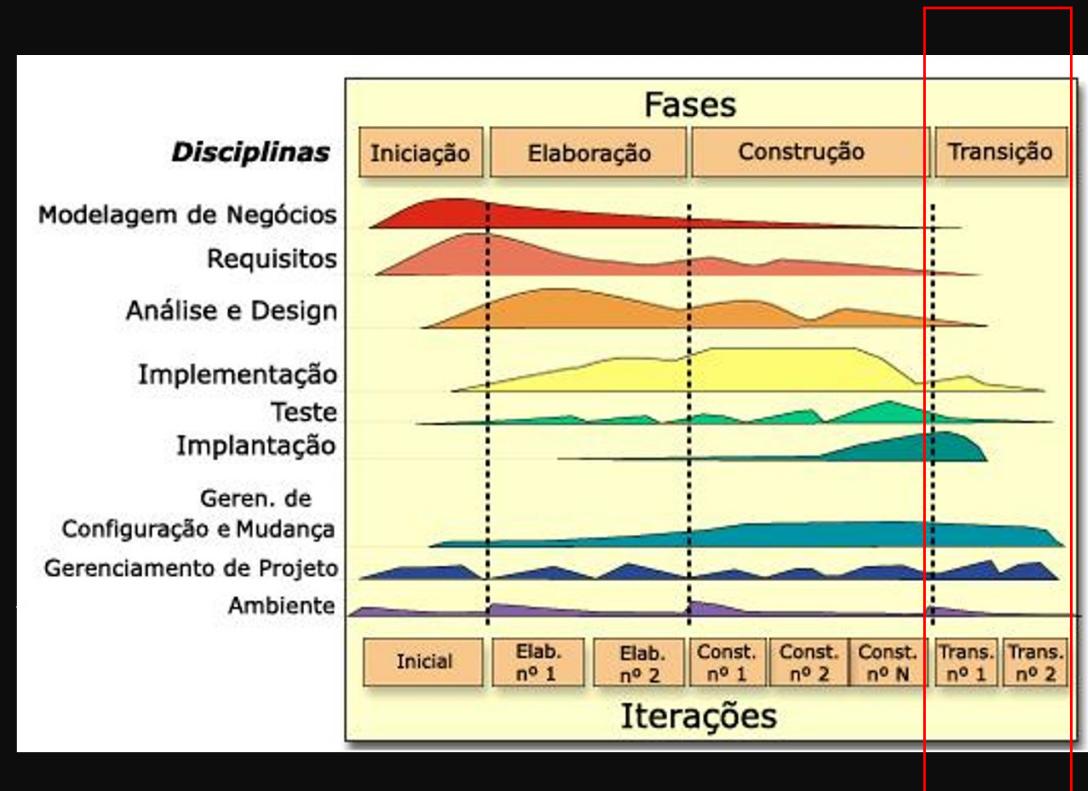
# Fase: Transição

Os objetivos principais dessa fase são:

- Realizar testes beta para validação do software.
- Treinar usuários.
- Preparar o hardware e as bases de dados.
- Preparar pacotes de instalação.
- Alcançar o OK das partes interessadas acerca dos critérios acordados inicialmente para o software (em outras palavras, validação do software).
- Anotar lições aprendidas.

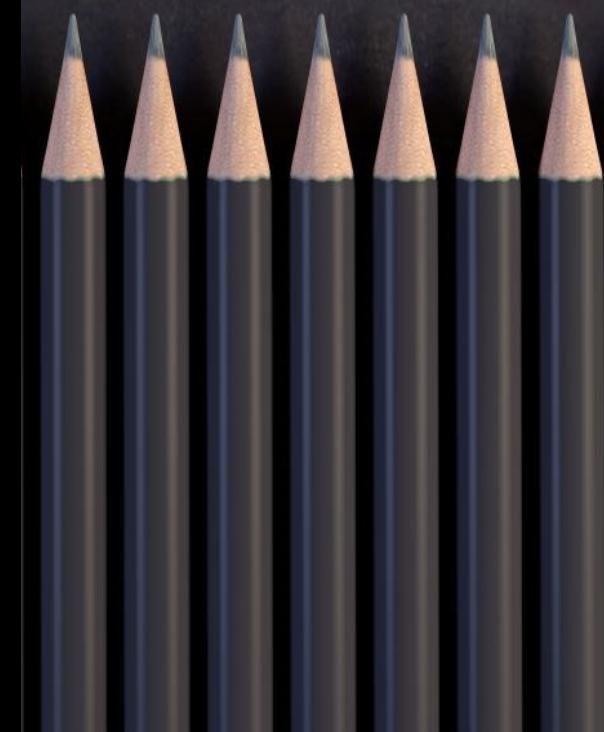
Os principais artefatos associados a fase são:

- as notas de release (que descrevem as versões dos softwares),
- os pacotes de instalação; e
- o material de treinamento.



# Aula 4

## Processos de iniciação do PMBOK



# Referências – Créditos



Este material faz uso dos slides de aula elaborados pela Profa. Dra. rer. nat. Christiane Gresse von Wangenheim, PMP

 creative  
COMMONS DEED

**Atribuição-Uso Não-Comercial-Compartilhamento pela Licença 2.5 Brasil**

**Você pode:**

- copiar, distribuir, exibir e executar a obra
- criar obras derivadas

**Sob as seguintes condições:**

**Atribuição** — Você deve dar crédito ao autor original, da forma especificada pelo autor ou licenciante.

**Uso Não-Comercial** — Você não pode utilizar esta obra com finalidades comerciais.

**Compartilhamento pela mesma Licença** — Se você alterar, transformar, ou criar outra obra com base nesta, você somente poderá distribuir a obra resultante sob uma licença idêntica a esta.

Para ver uma cópia desta licença, visite <http://creativecommons.org/licenses/by-nc-sa/2.5/br/> ou mande uma carta para Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

# Contextualização

## **PMBOK – Project Management Body of Knowledge**

- Corpo de conhecimentos em gerenciamento de projetos.
- PMBOK é um guia (um livro).

## Publicado pelo **PMI (Project Management Institute)**

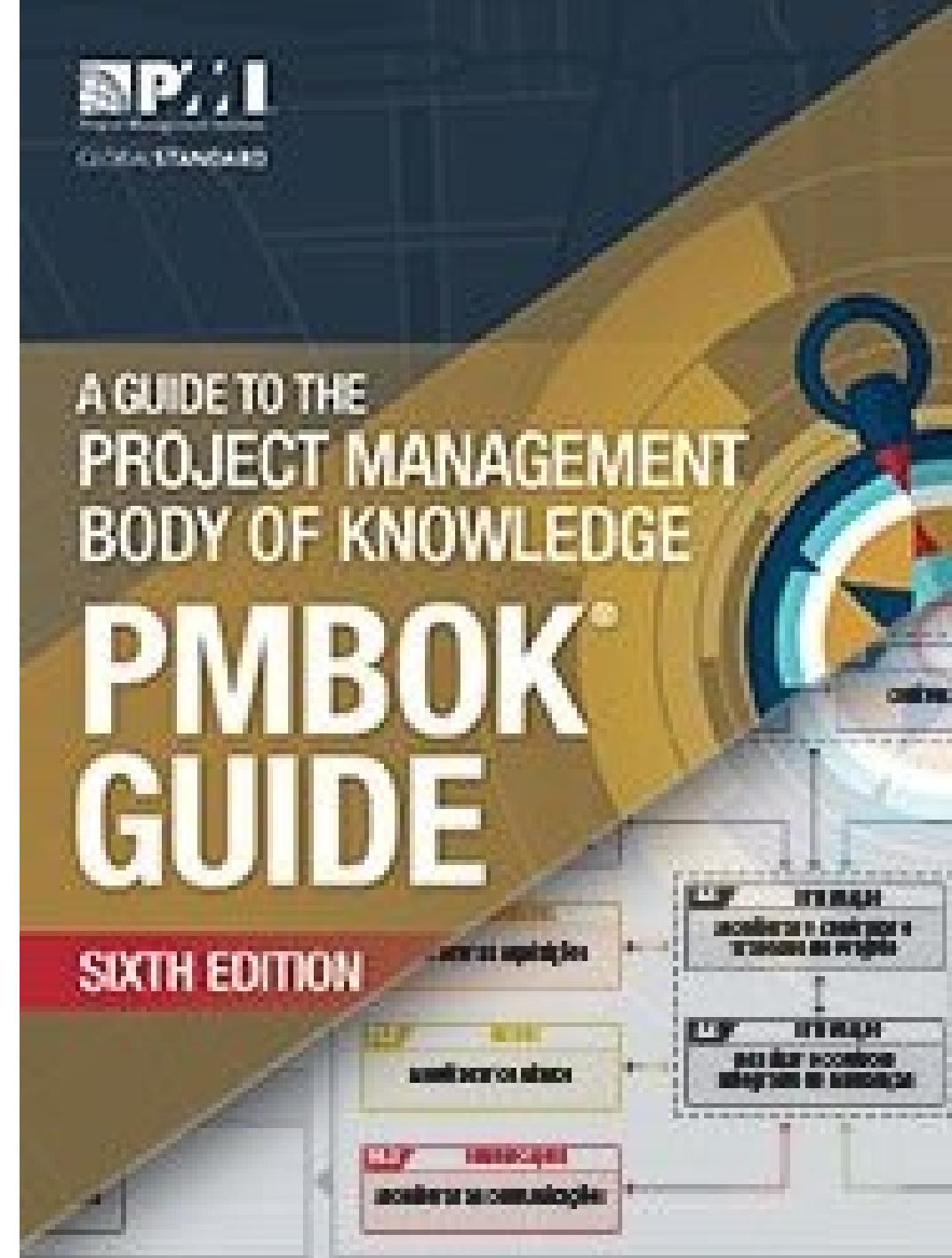
- É uma instituição internacional sem fins lucrativos.
- Associa profissionais de gestão de projetos.
- A maior associação do gênero no mundo (185 países).

## Certificação **PMP (Project Management Professional)**

- Reconhecida e exigida mundialmente.
- A certificação PMP atesta experiência e competência para conduzir e dirigir projetos.

# PMBOK

- PMBOK é a sigla para Project Management Body of Knowledge (Conjunto de Conhecimentos de Gerenciamento de Projetos). É um guia completo compilando as principais práticas, diretrizes, terminologias e métodos pertinentes ao setor de gestão.
- Em 1969 ocorreu a fundação do PMI (Project Management Institute) – uma instituição inicialmente idealizada como um fórum de estudos e debates para o desenvolvimento da área de gestão de projetos.
- Em 1984 o PMI lança o PMP (Project Management Professional), uma espécie de embrião do PMBOK. A partir de então, a idéia vai se desenvolvendo e ganhando corpo ao longo dos anos.
- Em 1996, o primeiro guia PMBOK é publicado.
- É importante observar que o final dos anos 90 foi um período marcado pela introdução da tecnologia no ambiente corporativo – foi, de fato, o começo de uma série de transformações que apontavam para as tendências que hoje observamos.



Source: <https://blog.zapsign.com.br/pmbok/>

# Contextualização

---

- **Gerenciamento de Projetos:** é a aplicação de **habilidades, conhecimentos, ferramentas e técnicas** com o objetivo de satisfazer os requisitos do projeto.
- **Projeto:** um esforço **temporário** (finito), com início e fim bem definidos, empreendido para alcançar um objetivo exclusivo (produzindo um resultado único).
- PMBOK é aplicável para projeto de qualquer área.

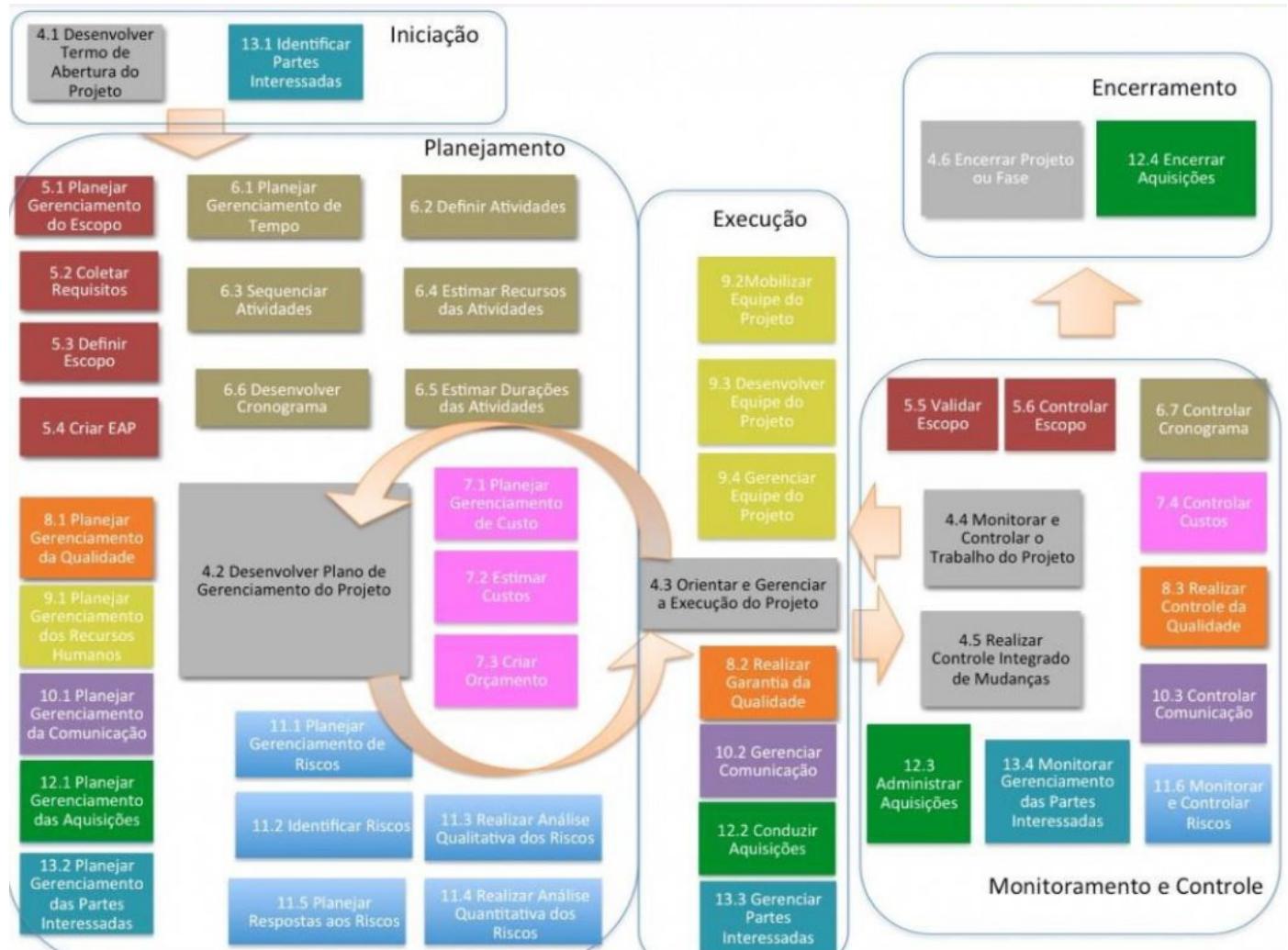


# Estrutura do PMBOK

	Iniciação	Planejamento	Execução	Monitoramento & Controle	Encerramento
Integração	4.1 Desenvolver o termo de abertura do projeto	4.2 Desenvolver o plano de gerenciamento do projeto	4.3 Orientar e gerenciar o trabalho do projeto	4.4 Monitorar & controlar o trabalho do projeto 4.5 Realizar o controle integrado de mudança	4.6 Encerrar o projeto ou a fase
Escopo		5.1 Planejar o gerenciamento de escopo 5.2 Coletar os requisitos 5.3 Definir o Escopo 5.4 Criar a EAP		5.5 Verificar o escopo 5.6 Controlar o escopo	
Tempo		6.1 Planejar o gerenciamento de tempo 6.2 Definir as atividades 6.3 Sequenciar as atividades 6.4 Estimar os recursos das atividades 6.5 Estimar a duração das atividades 6.6 Desenvolver o cronograma		6.7 Controlar o cronograma	
Custos		7.1 Planejar o gerenciamento de custos 7.2 Estimar os custos 7.3 Determinar o orçamento		7.4 Controlar os custos	
Qualidade		8.1 Planejar o gerenciamento de qualidade	8.2 Realizar a garantia da qualidade	8.3 Realizar o controle da qualidade	
RH		9.1 Planejar o gerenciamento de RH	9.2 Mobilizar a equipe do projeto 9.3 Desenvolver a equipe do projeto 9.4 Gerenciar a equipe do projeto		
Comunicações		10.1 Planejar o gerenciamento de comunicações	10.2 Gerenciar comunicações	10.3 Controlar as comunicações	
Riscos		11.1 Planejar gerenciamento de riscos 11.2 Identificar os riscos 11.3 Realizar a análise qualitativa dos riscos 11.4 Realizar a análise quantitativa dos riscos 11.5 Planejar as respostas aos riscos		11.6 Controlar os riscos	
Aquisições		12.1 Planejar o gerenciamento de aquisições	12.2 Realizar as aquisições	12.3 Controlar as aquisições	12.4 Encerrar as aquisições
Stakeholder	11.1 Identificar as partes interessadas	11.2 Planejar o gerenciamento das partes envolvidas	11.3 Gerenciar o envolvimento das partes interessadas	11.4 Controlar o envolvimento das partes interessadas	

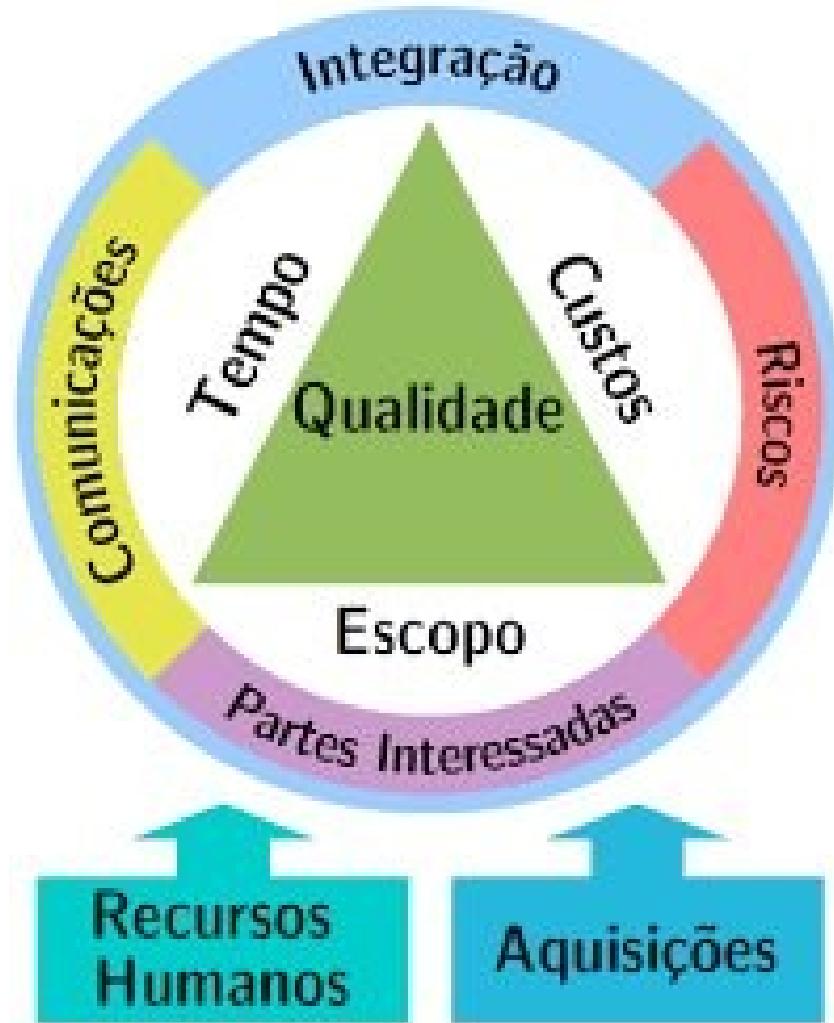
Fonte (PMI, 2013)

# Estrutura do PMBOK

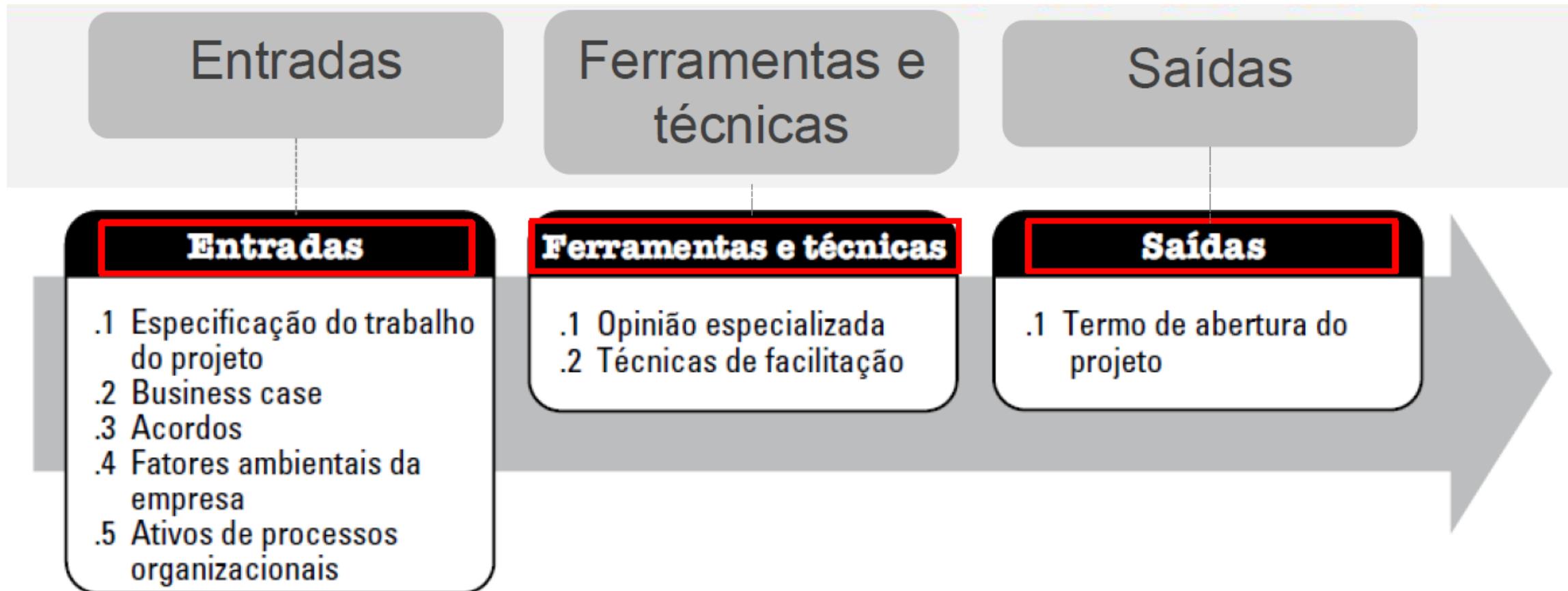


Fonte: <http://blog.mundopm.com.br/2013/03/14/47-processos-do-pmbok-5/>

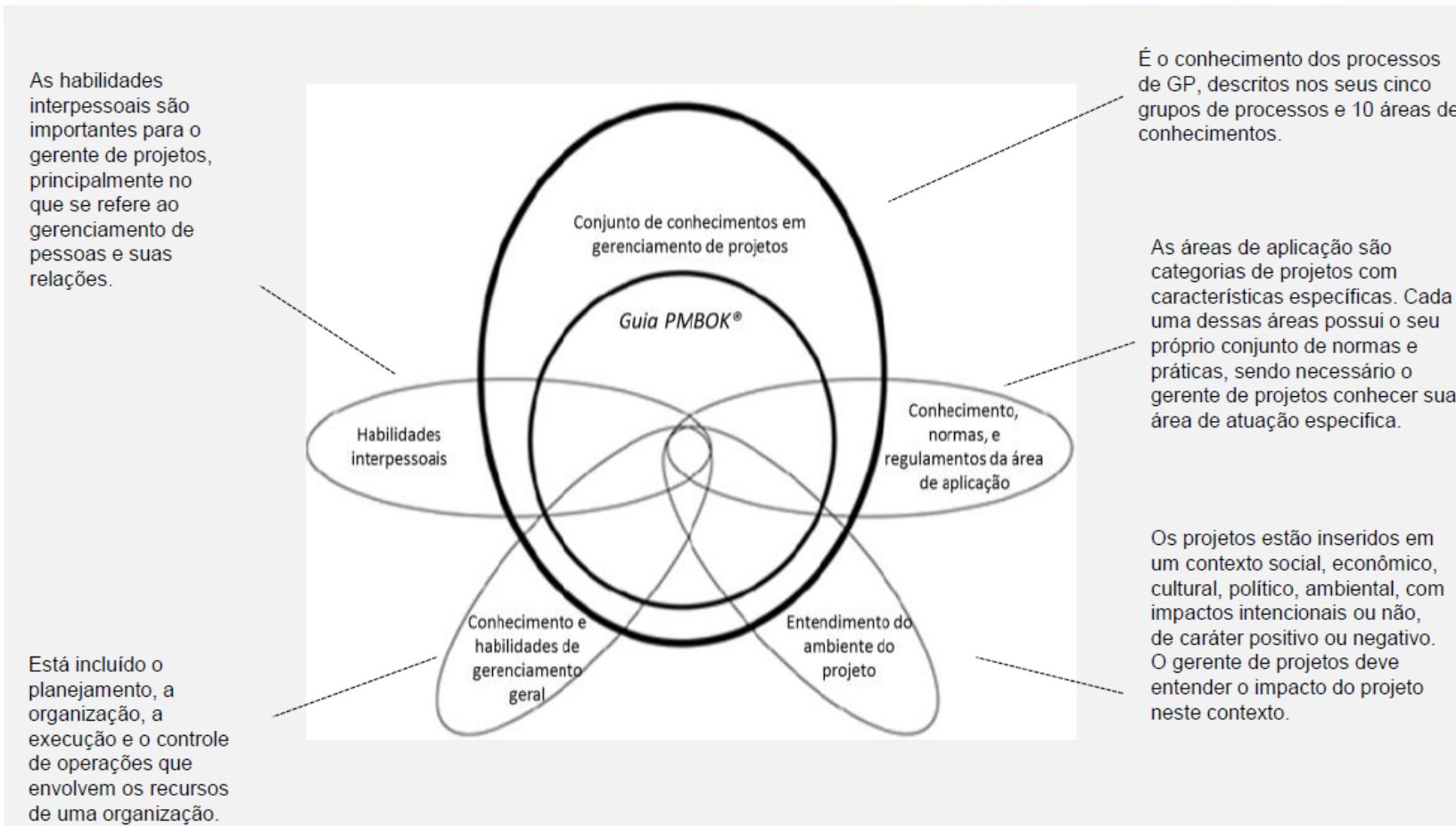
# Tripla restrição



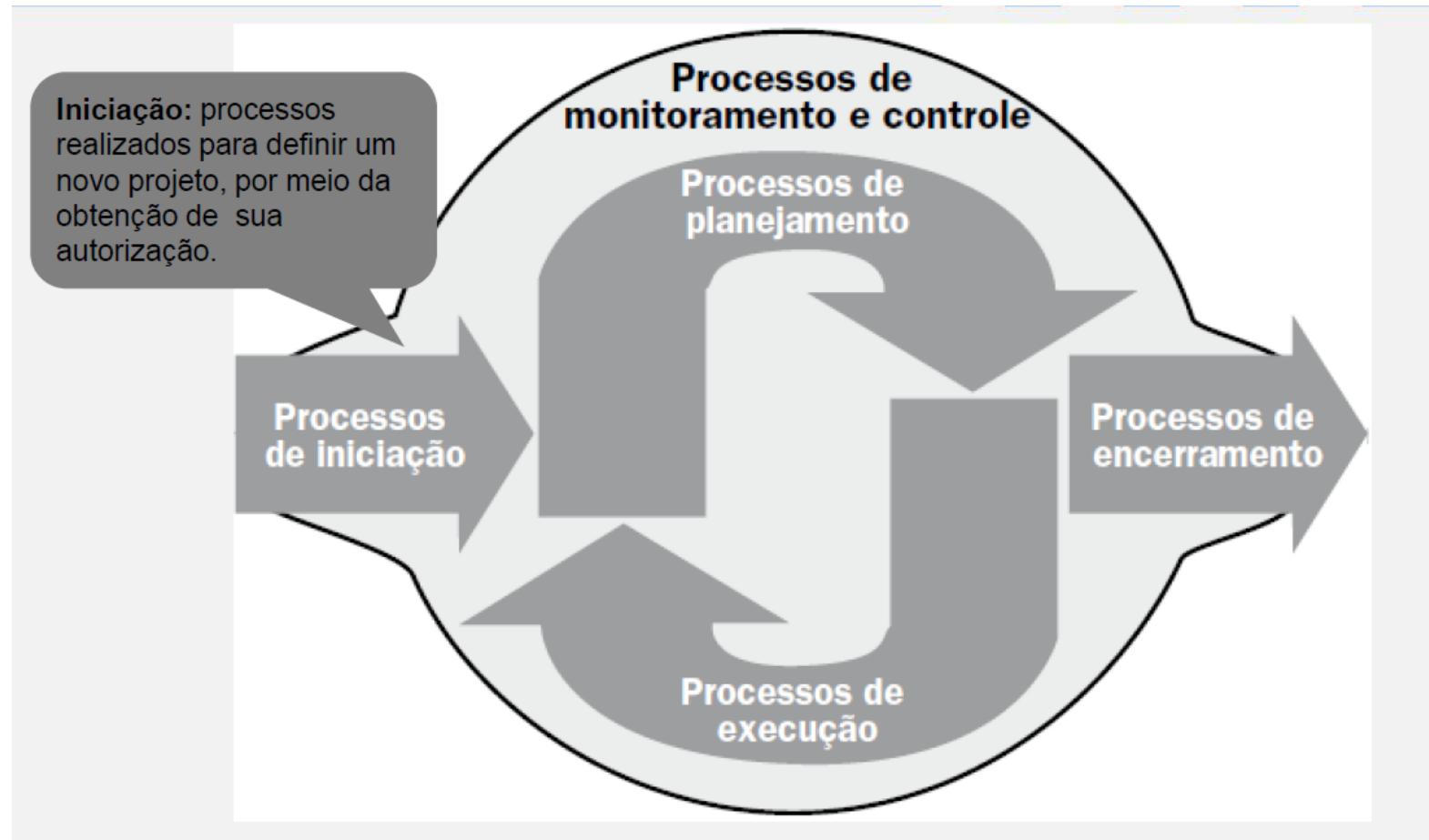
# Estrutura de um processo do PMBOK



# O papel do Gerente de Projetos



# Grupos de Processos



# Processos da Iniciação

	Iniciação	Planejamento	Execução	Monitoramento & Controle	Encerramento
Integração	4.1 Desenvolver o termo de abertura do projeto	4.2 Desenvolver o plano de gerenciamento do projeto	4.3 Orientar e gerenciar o trabalho do projeto	4.4 Monitorar & controlar o trabalho do projeto 4.5 Realizar o controle integrado de mudança	4.6 Encerrar o projeto ou a fase
Escopo		5.1 Planejar o gerenciamento de escopo 5.2 Coletar os requisitos 5.3 Definir o Escopo 5.4 Criar a EAP		5.5 Verificar o escopo 5.6 Controlar o escopo	
Tempo		6.1 Planejar o gerenciamento de tempo 6.2 Definir as atividades 6.3 Sequenciar as atividades 6.4 Estimar os recursos das atividades 6.5 Estimar a duração das atividades 6.6 Desenvolver o cronograma		6.7 Controlar o cronograma	
Custos		7.1 Planejar o gerenciamento de custos 7.2 Estimar os custos 7.3 Determinar o orçamento		7.4 Controlar os custos	
Qualidade		8.1 Planejar o gerenciamento de qualidade	8.2 Realizar a garantia da qualidade	8.3 Realizar o controle da qualidade	
RH		9.1 Planejar o gerenciamento de RH	9.2 Mobilizar a equipe do projeto 9.3 Desenvolver a equipe do projeto 9.4 Gerenciar a equipe do projeto		
Comunicações		10.1 Planejar o gerenciamento de comunicações	10.2 Gerenciar comunicações	10.3 Controlar as comunicações	
Riscos		11.1 Planejar gerenciamento de riscos 11.2 Identificar os riscos 11.3 Realizar a análise qualitativa dos riscos 11.4 Realizar a análise quantitativa dos riscos 11.5 Planejar as respostas aos riscos		11.6 Controlar os riscos	
Aquisições		12.1 Planejar o gerenciamento de aquisições	12.2 Realizar as aquisições	12.3 Controlar as aquisições	12.4 Encerrar as aquisições
Stakeholder	11.1 Identificar as partes interessadas	11.2 Planejar o gerenciamento das partes envolvidas	11.3 Gerenciar o envolvimento das partes interessadas	11.4 Controlar o envolvimento das partes interessadas	

# Iniciação

Processo: Desenvolver o Termo de Abertura do Projeto.

## Entradas

- .1 Especificação do trabalho do projeto
- .2 Business case
- .3 Acordos
- .4 Fatores ambientais da empresa
- .5 Ativos de processos organizacionais

## Ferramentas e técnicas

- .1 Opinião especializada
- .2 Técnicas de facilitação

## Saídas

- .1 Termo de abertura do projeto

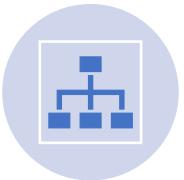
# Termo de Abertura



O DOCUMENTO QUE  
INICIA O PROJETO.



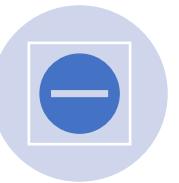
RECONHECE  
FORMALMENTE O  
INÍCIO DE UM NOVO  
PROJETO



BASE PARA O  
TRABALHO DO  
GERENTE DE PROJETO.



DESIGNA AUTORIDADE  
PARA O GERENTE DE  
PROJETOS APPLICAR OS  
RECURSOS  
ORGANIZACIONAIS NAS  
ATIVIDADES DO  
PROJETO.



POSSUI INFORMAÇÕES  
PRELIMINARES  
QUANTO AO ESCOPO, E  
RESTRIÇÕES DE  
PRAZOS, CUSTOS.



# Termo de Abertura

- ❑ Selecionar um novo projeto.
- ❑ Definir um novo projeto (ou uma nova fase de um projeto existente).
  - ❑ Definir escopo inicial.
  - ❑ Comprometer recursos financeiros iniciais.
  - ❑ Identificar partes interessadas.
  - ❑ Designar gerente de projeto.
- ❑ Resultado: **Termo de abertura do projeto**
  - ❑ Obter autorização do projeto (aprovação do termo de abertura).

# TAP: Entradas



Um business case é uma das possíveis entradas para elaboração de um termo de abertura.

## O propósito do projeto ou justificativa da questão

- Responder a pergunta “Por que?”
- Usado para justificar a necessidade do projeto.
- Deve claramente amarrar o projeto à estratégia da organização.
- Deve convencer responsáveis pelas decisões a suportar o projeto e inspirar membros da equipe.

TAP: Entradas –  
Business case

# TAP: Entradas

## Fatores ambientais da empresa

- Condições de mercado
- Carteira de clientes
- Cultura organizacional
- Estrutura organizacional

Fatores internos ou externos à empresa, que podem influenciar positiva ou negativamente o projeto.

O que depende do ambiente: é volátil, tem maior variabilidade no tempo.

## Ativos de processos organizacionais

- Políticas de estimativas de custo
- Informações históricas
- Conhecimento da equipe
- Lições aprendidas

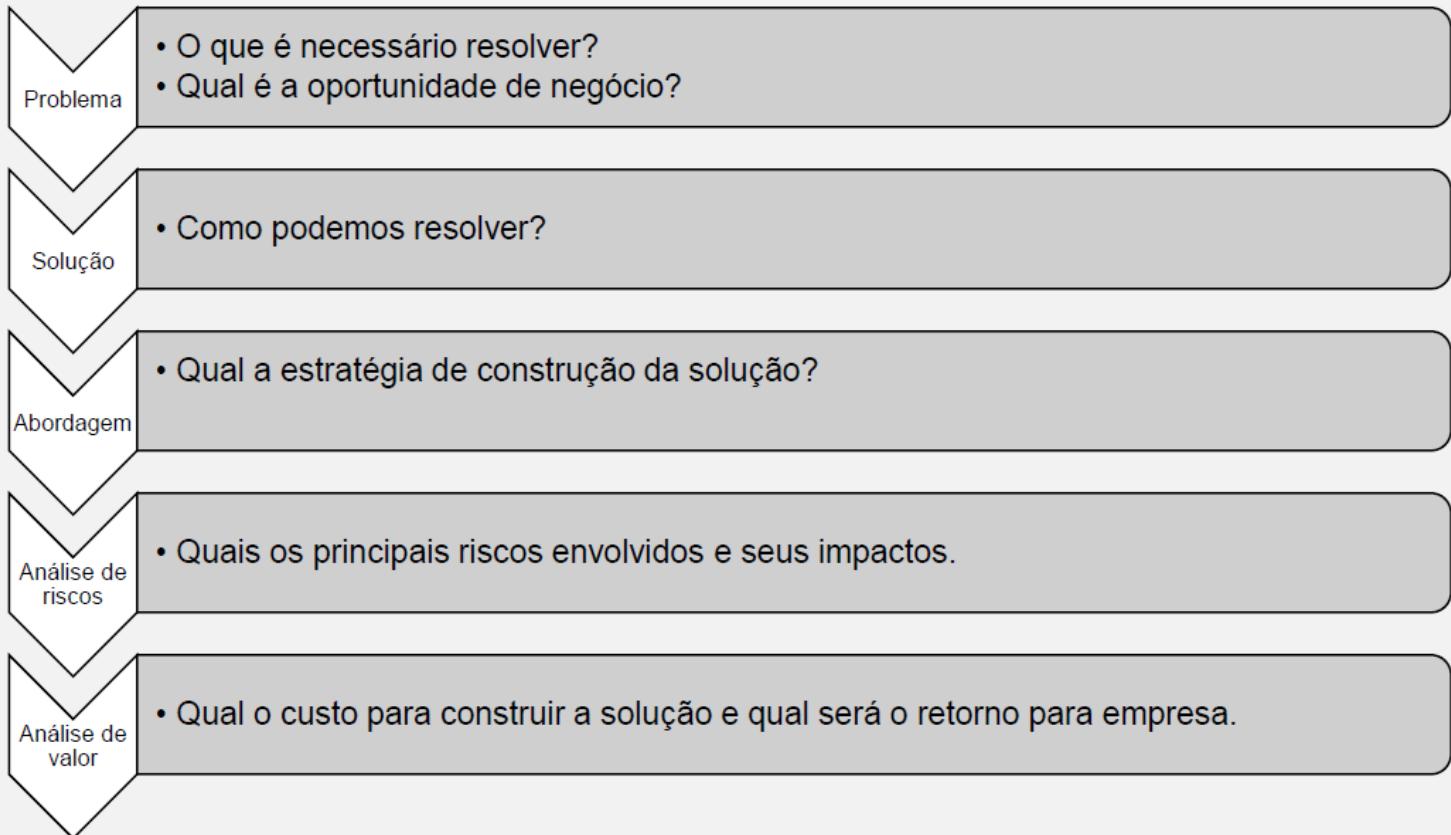
Planos e diretrizes a serem seguidos para execução do projeto.

Conhecimento da organização obtido em experiências anteriores.

Algo que a organização já possui e que mudanças ambientais tem pouco impacto.

# TAP: Entradas - Business case

## Business case



# Estrutura típica de um TAP

- ❑ **Título do Projeto**
- ❑ **Propósito/Justificativa do Projeto:** razão do negócio para iniciar o projeto, identificando a necessidade do negócio e o alinhamento estratégico.
- ❑ **Objetivo(s) do projeto**
- ❑ **Resultado(s) esperado(s)**
- ❑ **Premissas e restrições**
- ❑ **Resumo do orçamento** (estimativa do custo total)
- ❑ **Resumo do cronograma** (data inicio/ fim; duração total) **e marcos**
- ❑ **Partes interessadas:** incluindo a designação do gerente de projeto
- ❑ **Assinaturas de aprovação**

# Proposta – Justificativa do projeto

- Definido com base no caso de negocio.
- Exemplos típicos:
  - Aumentar a faixa de mercado na região por X% Y Z em meses.
  - Alcançar um volume de vendas de unidades de X ou Y no valor total de R\$ Z em x meses.
  - Atingir X% de lucro ou retorno sobre o investimento em Y meses Y.
  - Alcançar uma medida de satisfação do cliente de pelo menos Y em X meses depois do lançamento.
  - Desenvolver competências tecnológicas mensuráveis do núcleo da organização.

# Objetivo(s) do projeto:

Boa prática:  
SMART

- ❑ Identificar claramente os objetivos a serem atingidos pelo projeto:
  - ❑ Específico (*specific*)
  - ❑ Mensurável (*measurable*)
  - ❑ Orientado a ação (*action-oriented*)
  - ❑ Realista (*realistic*)
  - ❑ Limitado pelo tempo (*time-limited*)

# Objetivo(s) do projeto

- Desenvolver um sistema de software para Megalith Records.  
:( Incompleto
- Desenvolver um sistema de software para gerenciar os contatos, álbuns produzidos e shows organizados pela Megalith Records.  
☺ Aceitável
- Desenvolver e instalar um sistema de software para gerenciar os contatos, álbuns produzidos (incluindo lista de faixas) e shows organizados (com venda de ingresso) pela Megalith Records .  
☺ Bom

# Resultados esperados

**Entregável (*deliverable*):** qualquer produto, resultado ou capacidade para realizar um serviço único e verificável e que deve ser produzido para concluir um processo, uma fase ou um projeto.

- ❑ Muitas vezes utilizado mais especificamente com referência a uma **entrega externa**, que a uma entrega sujeita a aprovação do patrocinador ou do cliente do projeto.
- ❑ Exemplos:
  - ❑ Documento de definição de requisitos
  - ❑ Web site da empresa ABC
  - ❑ Instalação e configuração do software de internet e hardware

# Premissas

**Premissas** são aspectos que são assumidos como verdadeiros e que caso não ocorram causarão um problema ao projeto.

Exemplos:

- O fornecedor de requisitos no cliente deverá estar disponível por 40 horas no primeiro mês do projeto.
- A nova versão do banco de dados a ser utilizado neste projeto será lançado no dia 1 de Junho.

# Restrições

**Restrições** são fatos concretos que limitam as opções.

- ❑ Determinam os **critérios de sucesso**: definições mensuráveis do que deve ser feito para o projeto seja aceitável para os *stakeholders*.

Exemplos:

- ❑ O projeto deverá estar concluído até o inicio da feira de informática em junho.
- ❑ O limite de custos é de R\$ 100.000,00.
- ❑ O processo de software deve seguir o processo padrão em conformidade ao nível 2 de maturidade do CMMI.
- ❑ Somente software de código aberto e livre poderá ser utilizado neste projeto.

# Resumo do orçamento

- Estimativas de custo total do projeto
- Exatidão de uma estimativa aumenta conforme o projeto se desenvolve.
  - Fase de iniciação: Estimativa grosseira na faixa de -50 a +100%
  - Etapa posterior: redução a uma faixa de -10 a +15%

# Resumo do cronograma

- ❑ Identificação de marcos e prazos para entregas.
  - ❑ **Marco (Milestone)**: evento significativo do projeto
    - ❑ Completada uma fase do projeto
    - ❑ Entrega de um dos *entregáveis* ao cliente
    - ❑ Representa premissas ou restrições
  - ❑ Normalmente representados como eventos sem duração, recursos ou custo.
  - ❑ Identificação da data inicio e data término e/ou duração total.

# Aprovação

- ❑ Projetos são tipicamente autorizados como resultado de:
  - ❑ Demanda do mercado
  - ❑ Necessidade de negocio
  - ❑ Requisição de cliente
  - ❑ Avanço tecnológico
  - ❑ Requisito legal
- ❑ Quem aprova?
  - ❑ Tipicamente: Patrocinador, PMO, etc.
  - ❑ Outras partes chaves interessadas no projeto

## Designação do Gerente de Projetos



- ❑ Um gerente de projeto é identificado, selecionado e designado.
- ❑ Autorização do projeto é feita por alguém externo ao projeto:
  - ❑ Patrocinador
  - ❑ PMO
  - ❑ Comitê direutivo de portfólio ...

# Por que precisamos de um TAP?

---

- ❑ Obter autorização para prosseguir com o projeto e obter a aprovação de recursos suficientes para avançar para a próxima fase do projeto.
- ❑ Designar e autorizar o Gerente de Projetos.
- ❑ Comunicar às partes interessadas os objetivos do projeto para que estes possam ter um entendimento comum.
- ❑ Rapidamente eliminar projetos ruins ou sem condições de realização.

# Iniciação

- Processo: Identificar as Partes Interessadas

## Entradas

- .1 Termo de abertura do projeto
- .2 Documentos de aquisição
- .3 Fatores ambientais da empresa
- .4 Ativos de processos organizacionais

## Ferramentas e técnicas

- .1 Análise de partes interessadas
- .2 Opinião especializada
- .3 Reuniões

## Saídas

- .1 Registro das partes interessadas



## Identificar as partes interessadas

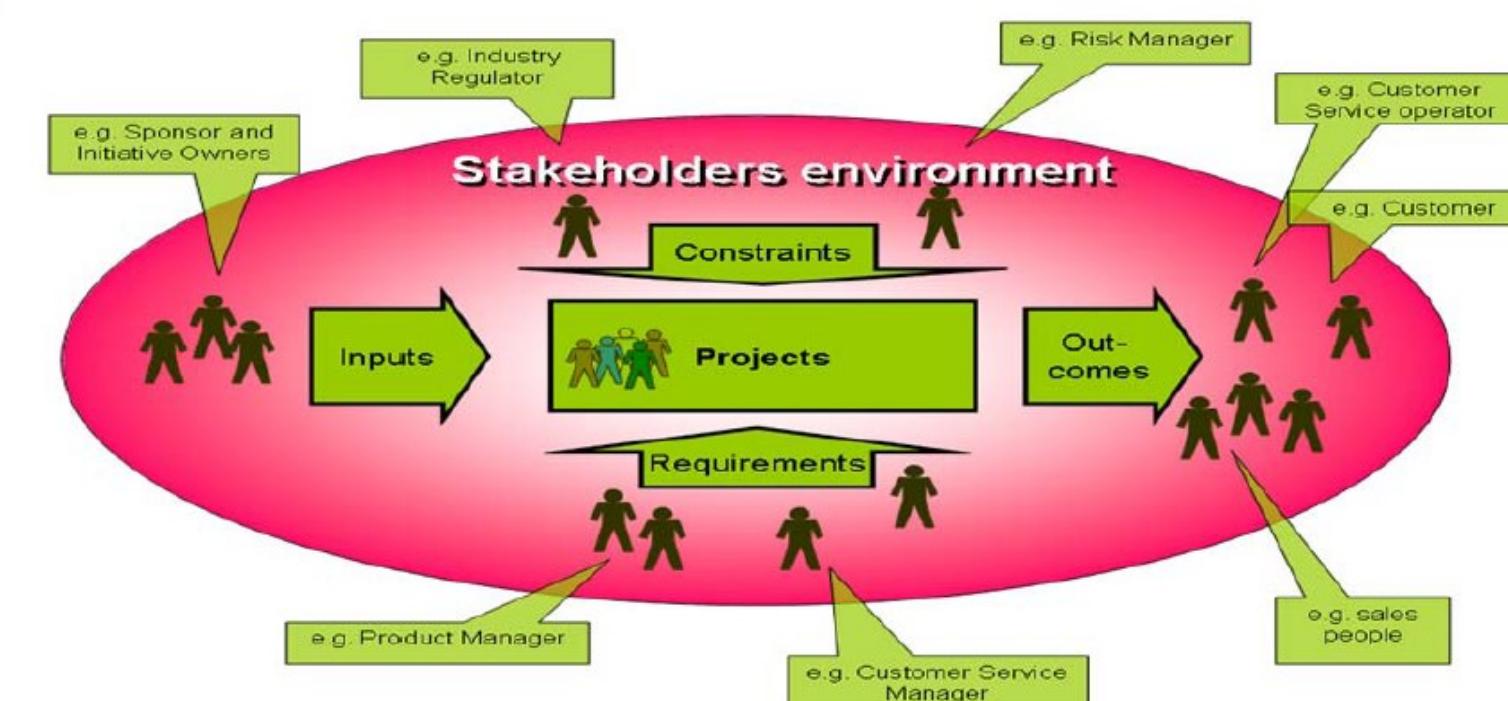


É fundamental para o sucesso do projeto identificar as partes interessadas desde o início e analisar seus níveis de interesse, envolvimento, importância e influência.

- Identificar todas as pessoas ou organizações que podem ser afetadas (positivamente ou negativamente) pelo projeto.
- Documentar as informações relevantes relacionadas aos seus interesses, envolvimento e impacto no sucesso do projeto.

# Partes interessadas

**Partes interessadas (stakeholders)** são pessoas ou organizações ativamente envolvidas no projeto ou cujos interesses podem ser positivo ou negativamente afetados pela execução ou término do projeto.



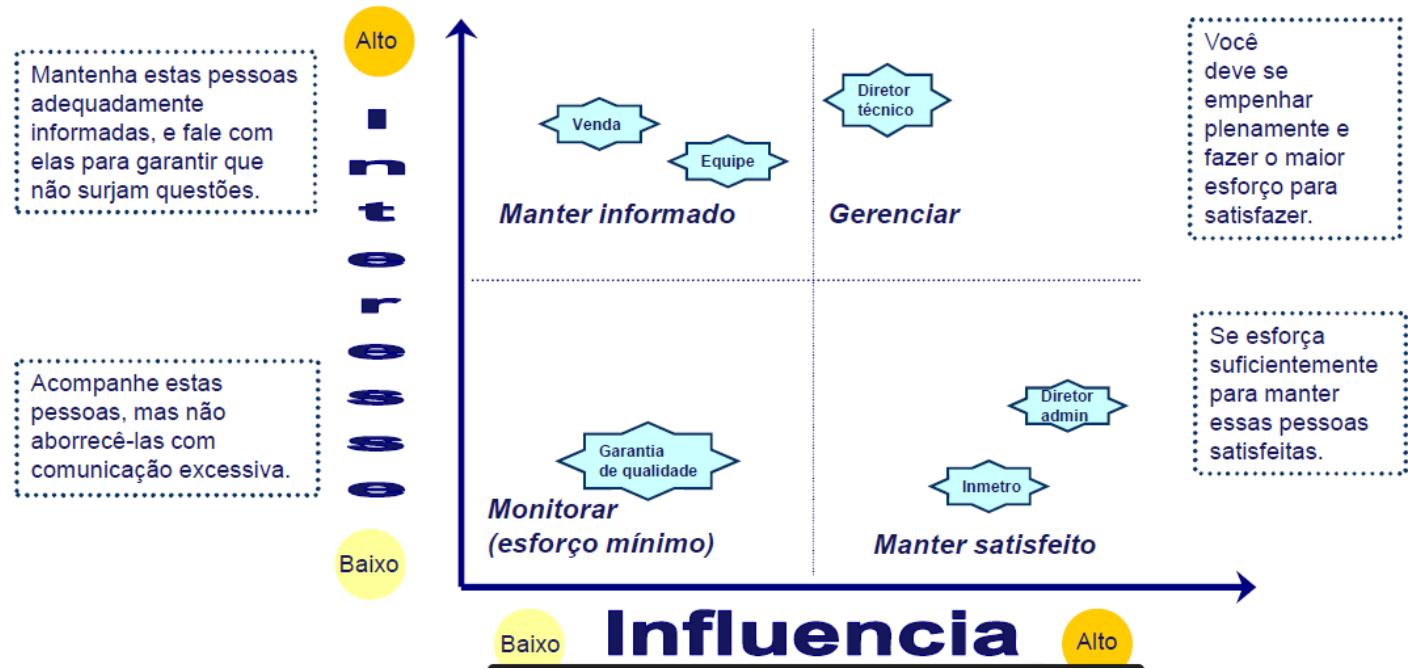
## Identificar partes interessadas

- ☐ Identificar todos os *stakeholder* utilizando um **diagrama de contexto** de análise de *stakeholders*.



# Análise de interesse e influência

- ❑ Coloque todas as partes interessadas no diagrama de acordo com o nível de interesse e influência que eles têm no projeto, e decida sobre um curso de ação.



# Template – Partes interessadas

Parte interessada	interno/externo	Interesse	Influência	Estratégia	

# Exemplo - questão

Acerca de gerenciamento de projetos (PMBOK 7), gerenciamento de serviços (ITIL v4) e governança de TI (COBIT 2019), julgue o item seguinte.

Conforme o PMBOK, termo de abertura é o documento que formaliza o projeto, estabelecendo suas premissas e seus objetivos

**Verdadeiro**

Falso

# Atividade: Elaborar o termo de abertura do projeto

- Título do Projeto**
- Propósito/Justificativa do Projeto:** razão do negócio para iniciar o projeto, identificando a necessidade do negócio e o alinhamento estratégico.
- Objetivo(s) do projeto**
- Resultado(s) esperado(s)**
- Premissas e restrições**
- Resumo do orçamento** (estimativa do custo total)
- Resumo do cronograma** (data inicio/ fim; duração total) **e marcos**
- Partes interessadas:** incluindo a designação do gerente de projeto
- Assinaturas de aprovação**
- Incluir análise de interesse e influência das partes interessadas**



Aula 5  
Hands on



# Aula prática: Produzir os artefatos da M1

Utilize este encontro para produzir os artefatos que precisam ser entregues na M1:

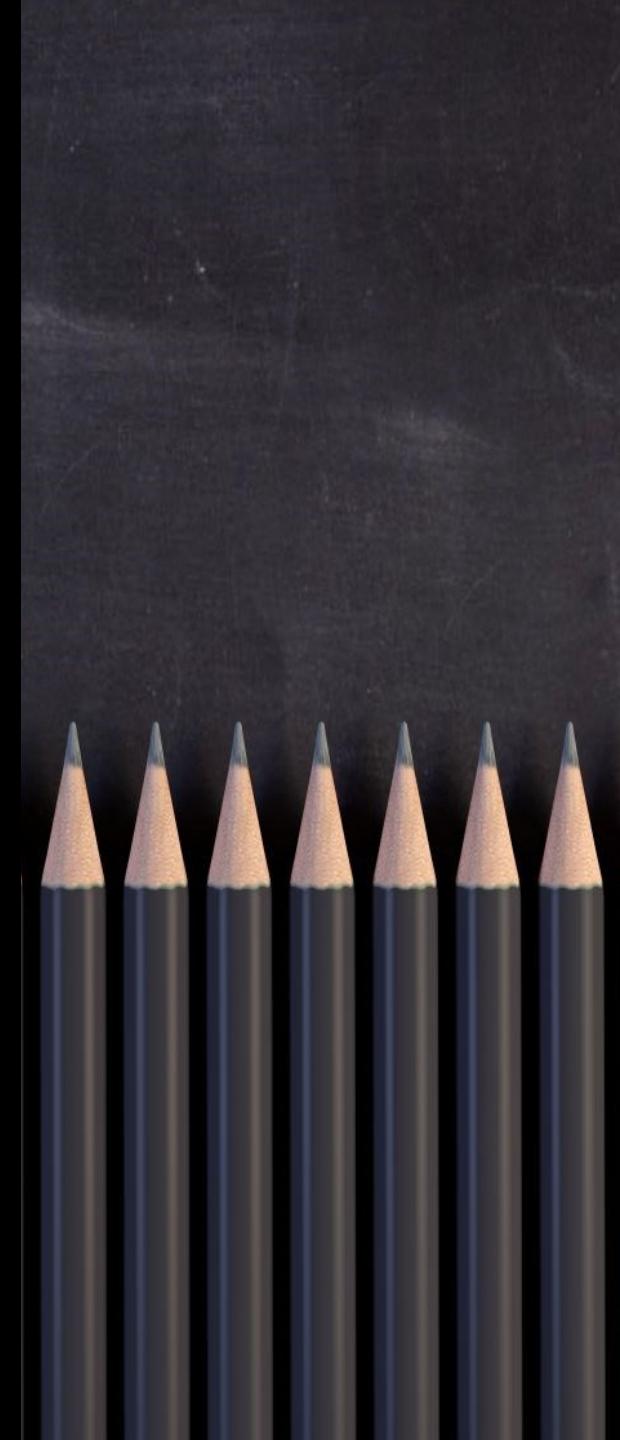
- Termo de abertura do projeto
- Modelo de casos de uso
  - Diagrama de casos de uso da UML
  - Detalhar o fluxo principal de cada caso de uso
- Protótipos de tela para os casos de uso
- Documento de Arquitetura candidata do sistema
  - Diagrama de classes
  - Diagrama de sequência para cada caso de uso
  - Aplicar 3 padrões de projeto GOF ( 1 de cada categoria)

# Aula 6

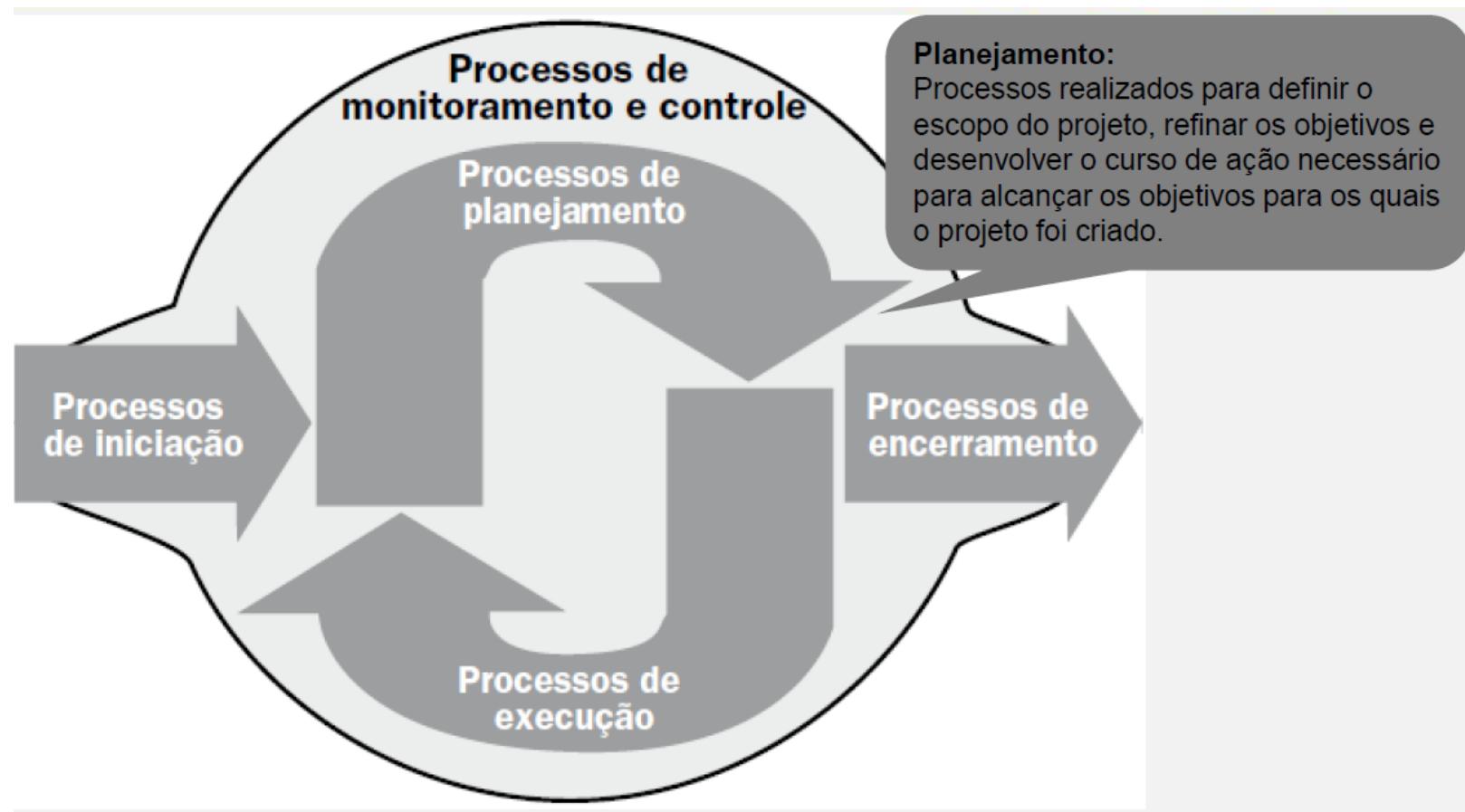
Trabalho M1 – Apresentação e entrega



# Aula 7 – Planejar escopo do projeto



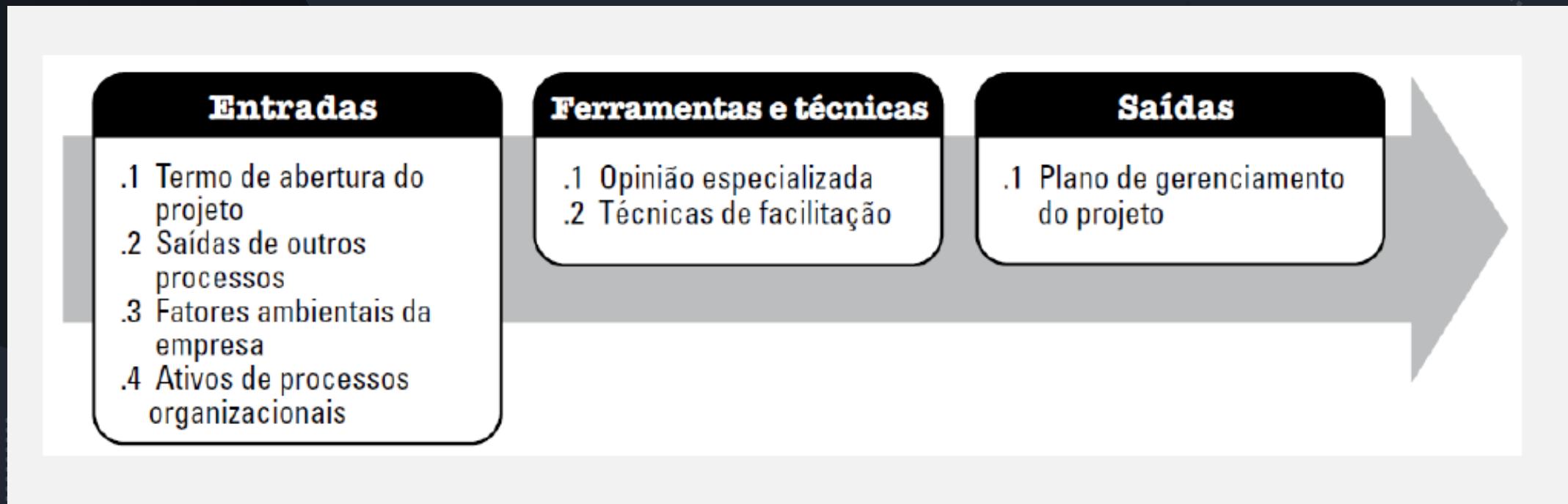
# Grupo de Processos Planejamento



# Processos de planejamento

	Iniciação	Planejamento	Execução	Monitoramento & Controle	Encerramento
Integração	4.1 Desenvolver o termo de abertura do projeto	4.2 Desenvolver o plano de gerenciamento do projeto	4.3 Orientar e gerenciar o trabalho do projeto	4.4 Monitorar & controlar o trabalho do projeto 4.5 Realizar o controle integrado de mudança	4.6 Encerrar o projeto ou a fase
Escopo		5.1 Planejar o gerenciamento de escopo 5.2 Coletar os requisitos 5.3 Definir o Escopo 5.4 Criar a EAP		5.5 Verificar o escopo 5.6 Controlar o escopo	
Tempo		6.1 Planejar o gerenciamento de tempo 6.2 Definir as atividades 6.3 Sequenciar as atividades 6.4 Estimar os recursos das atividades 6.5 Estimar a duração das atividades 6.6 Desenvolver o cronograma		6.7 Controlar o cronograma	
Custos		7.1 Planejar o gerenciamento de custos 7.2 Estimar os custos 7.3 Determinar o orçamento		7.4 Controlar os custos	
Qualidade		8.1 Planejar o gerenciamento de qualidade	8.2 Realizar a garantia da qualidade	8.3 Realizar o controle da qualidade	
RH		9.1 Planejar o gerenciamento de RH	9.2 Mobilizar a equipe do projeto 9.3 Desenvolver a equipe do projeto 9.4 Gerenciar a equipe do projeto		
Comunicações		10.1 Planejar o gerenciamento de comunicações	10.2 Gerenciar comunicações	10.3 Controlar as comunicações	
Riscos		11.1 Planejar gerenciamento de riscos 11.2 Identificar os riscos 11.3 Realizar a análise qualitativa dos riscos 11.4 Realizar a análise quantitativa dos riscos 11.5 Planejar as respostas aos riscos		11.6 Controlar os riscos	
Aquisições		12.1 Planejar o gerenciamento de aquisições	12.2 Realizar as aquisições	12.3 Controlar as aquisições	12.4 Encerrar as aquisições
Stakeholder	11.1 Identificar as partes interessadas	11.2 Planejar o gerenciamento das partes envolvidas	11.3 Gerenciar o envolvimento das partes interessadas	11.4 Controlar o envolvimento das partes interessadas	

# Desenvolver o plano do projeto



# Desenvolver o plano do projeto

- Realizado no **início e ao longo do ciclo de vida** do projeto!
  - Adequação a **mudanças significativas** ⇒ re-planejamento

## □ **Planejamento por ondas sucessivas**

- Planejamento alto nível do projeto todo
- Planejamento detalhado da próxima fase



# Desenvolver o plano do projeto

- ❑ Documentar as ações necessárias para definir, preparar, integrar e coordenar todos os planos auxiliares.
  - ❑ “*Planejar o planejamento.*”
- ❑ **Plano (de gerenciamento) do projeto:** fonte principal de informações sobre como o mesmo será planejado, executado, monitorado, controlado e encerrado.
  - ❑ Conteúdo do plano de projeto varia dependendo da área de aplicação e complexidade do projeto.
  - ❑ O plano é desenvolvido, revisado e controlado por meio de uma série de processos integrados até o encerramento do projeto.

# Escopo

---



O que será feito?

- ❑ Refere-se a todo resultado a ser gerado no projeto.
- ❑ Documentação formal usada para descrever o que deve e o que não deve ser incluído no projeto.

# Planejar o Gerenciamento do Escopo

## Entradas

- .1 Plano de gerenciamento do projeto
- .2 Termo de abertura do projeto
- .3 Fatores ambientais da empresa
- .4 Ativos de processos organizacionais

## Ferramentas e técnicas

- .1 Opinião especializada
- .2 Reuniões

## Saídas

- .1 Plano de gerenciamento do escopo
- .2 Plano de gerenciamento dos requisitos

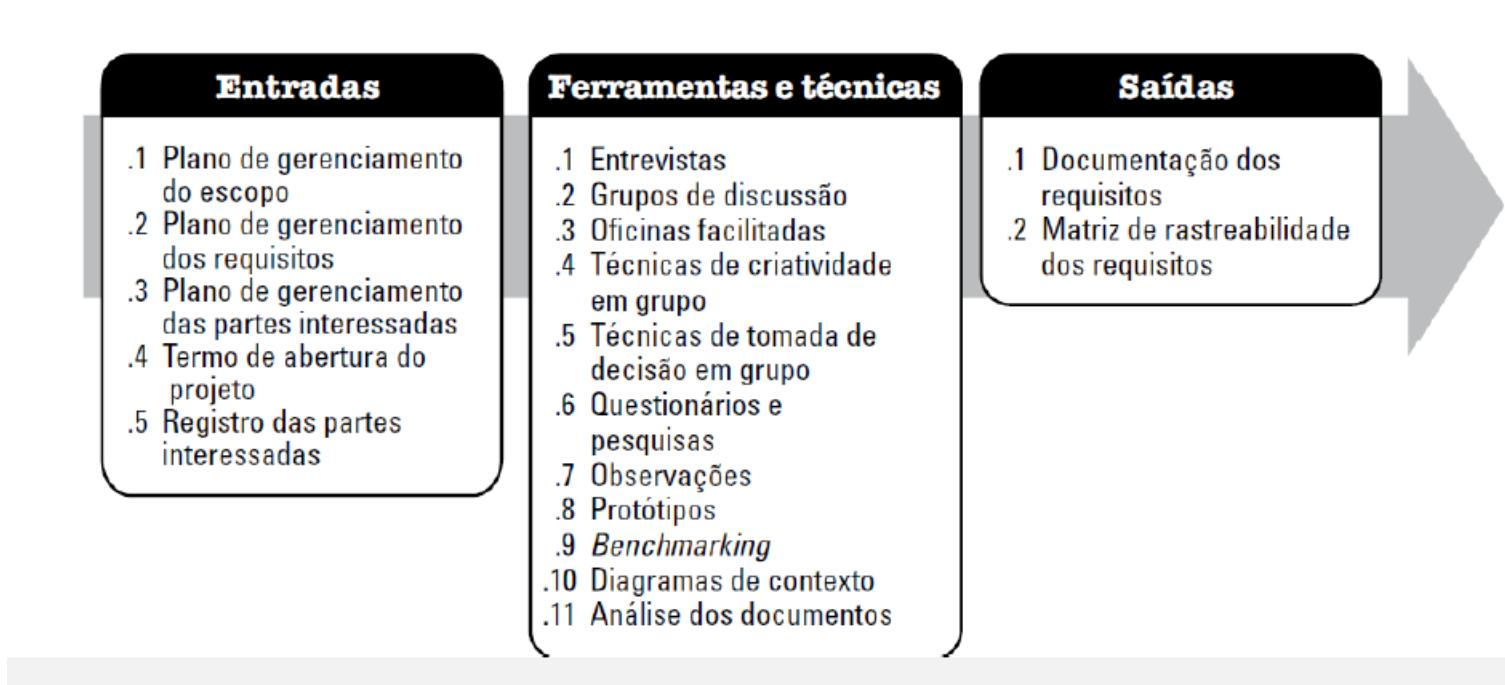


# Coletar Requisitos

Coletar os requisitos  
Definir o Escopo  
Criar a EAP

Primeiro passo: Entender  
as necessidades e  
expectativas das partes  
interessadas.

# Coletar Requisitos



# Coletar Requisitos

- ❑ Definir e documentar as funções/funcionalidades do projeto e do produto necessárias para atender às necessidades e expectativas das partes interessadas.
  - ❑ Requisitos do projeto: requisitos de negócio, de gerenciamento do projeto, de entrega, etc.
  - ❑ Requisitos do produto: requisitos funcionais e/ou não funcionais do produto.

# Coletar Requisitos

Documentação de requisitos:

Stakeholder	Requisitos	Prioridade
<Identificar o nome ou a organização dos interessados>	<Identificar os requisitos>	<Priorizar os requisitos>
Diretor	Módulo do sistema web para a composição de jogos	Alta
Diretor	Módulo do sistema web para a venda de jogos	Médio
Vendas	Campanha de propaganda do serviço	Baixo
...	...	...

# Coletar Requisitos

Documentar no mínimo 6 requisitos para o projeto considerando os temas definidos.

Stakeholder	Requisitos	Prioridade
<i>&lt;Identificar o nome ou a organização dos interessados&gt;</i>	<i>&lt;Identificar os requisitos&gt;</i>	<i>&lt;Priorizar os requisitos&gt;</i>
...	...	...

# Coletar Requisitos

Gerenciar conflitos de requisitos entre stakeholders.

- Objetivos concorrentes  
incluem demandas conflitantes ou o tempo / custos do projeto.
- Conflitos comuns:
  - Cronograma
  - Prioridades
  - Recursos
  - Crenças técnicas
  - Políticas e procedimentos
  - Custos
  - Personalidades



# Gestão de mudanças

Requisitos inevitavelmente mudam:

- Negócios ou ambientes operacionais mudam
- Uma melhor compreensão do sistema é desenvolvida
- Funcionalidades faltante
- Sistema precisa de ser adaptado a necessidades específicas
- Correção de defeitos



# Matriz de rastreabilidade

- ❑ Liga os requisitos às suas origens e os rastreia durante todo o ciclo de vida do projeto.
- ❑ Fornece uma estrutura de gerenciamento das mudanças do escopo do produto.

Requirement Source	Product Requirements	HLD Section #	LLD Section #	Code Unit	UTS Case #	STS Case #	User Manual
Business Rule #1	R00120 Credit Card Types	4.1 Parse Mag Strip	4.1.1 Read Card Type	Read_Card_Type.c Read_Card_Type.h	UT 4.1.032 UT 4.1.033 UT 4.1.038 UT 4.1.043	ST 120.020 ST 120.021 ST 120.022	Section 12
			4.1.2 Verify Card Type	Ver_Card_Type.c Ver_Card_Type.h Ver_Card_Types.dat	UT 4.2.012 UT 4.2.013 UT 4.2.016 UT 4.2.031 UT 4.2.045	ST 120.035 ST 120.036 ST 120.037 ST 120.037	Section 12
Use Case #132 step 6	R00230 Read Gas Flow	7.2.2 Gas Flow Meter Interface	7.2.2 Read Gas Flow Indicator	Read_Gas_Flow.c	UT 7.2.043 UT 7.2.044	ST 230.002 ST 230.003	Section 21.1.2
	R00231 Calculate Gas Price	7.3 Calculate Gas price	7.3 Calculate Gas price	Cal_Gas_Price.c	UT 7.3.005 UT 7.3.006 UT 7.3.007	ST 231.001 ST 231.002 ST 231.003	Section 21.1.3

# Definir o escopo

Coletar os requisitos  
Definir o Escopo  
Criar a EAP

Desenvolvimento de uma descrição detalhada do projeto e do produto.

# Definir o escopo

## Entradas

- .1 Plano de gerenciamento do escopo
- .2 Termo de abertura do projeto
- .3 Documentação dos requisitos
- .4 Ativos de processos organizacionais

## Ferramentas e técnicas

- .1 Opinião especializada
- .2 Análise de produto
- .3 Geração de alternativas
- .4 Oficinas facilitadas

## Saídas

- .1 Declaração do escopo do projeto
- .2 Atualizações nos documentos do projeto



# Definir o escopo

---

- ❑ **Declaração do escopo do projeto:** descreve detalhadamente as entregas do projeto e o trabalho necessário para criar as entregas.
- ❑ Propósito: documentar o objetivo do projeto, entregas, e requisitos para que possam ser usados como *baseline* de futuras decisões do projeto.
- ❑ Elaborar progressivamente a descrição do projeto no termo da abertura do projeto.
- ❑ Inclui tipicamente:
  - ❑ Descrição do escopo do produto
  - ❑ Critérios de aceitação
  - ❑ Entregas do projeto
  - ❑ Limites do projeto
  - ❑ Restrições e premissas

# Exemplo simplificado: Declaração do escopo

**Objetivo do projeto:** Desenvolver um web site onde autores de jogos possam montar o jogo de forma profissional e vender online.

O sistema suportará a composição de um jogo, permitindo o *upload* do design gráfico do tabuleiro de cartas e a seleção de peças (dados, etc.) . O sistema automaticamente calculará os custos de produção do jogo e o autor poderá definir um preço (destinando 50% do lucro para o autor e 50% para a empresa Jogos Educativos Ltda.), uma vez composto, o jogo será disponibilizado na loja online que fará parte do site, permitindo a compra dos jogos (incluindo um catalogo de jogos e a realização de um pedido). O sistema também suportará gerencia do reembolso do lucro aos autores. O acesso a composição de jogos e a efetuação da compra será restrito a usuário cadastrado, o catalogo dos jogos será aberto a todos.

## Resultados esperados:

Módulo do sistema web para a composição de jogos (incluindo *upload* de design de tabuleiro, *upload* de design de cartas, seleção de dados, etc.) , cálculo automático do custo de produção, definição do preço pelo autor.

Módulo de sistema web para venda dos jogos (incluindo catalogo de jogos (browsing e busca), pedidos)

Módulo para gerênciia dos usuários (incluindo controle de acesso, controle financeira)

Instalação do sistema web no servidor da empresa

Campanha de propaganda do serviço (incluindo a definição da estratégia da campanha, design de logo e folders, realização das ações de propaganda)

## Outros requisitos:

Segurança: O site da intranet deve fornecer vários níveis de segurança...

O site deve ser acessíveis usando um navegador de Internet padrão.

O site devem estar disponíveis 24 horas por dia, 7 dias por semana, com uma hora por semana para manutenção do sistema de manutenção periódica e outros, conforme o caso.

# Criar a estrutura analítica do projeto - EAP

Coletar os requisitos

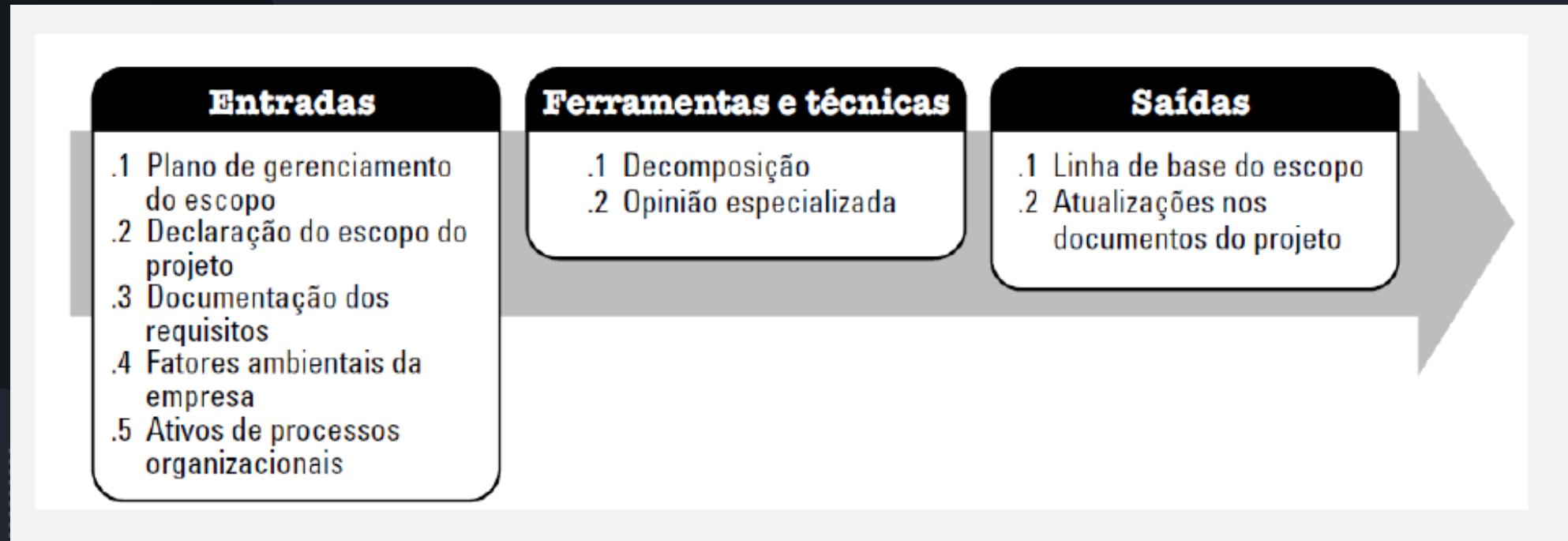
Definir o Escopo

Criar a EAP

Subdivisão das entregas e do trabalho do projeto em componentes menores e de gerenciamento mais fácil.

# Criar a estrutura analítica do projeto

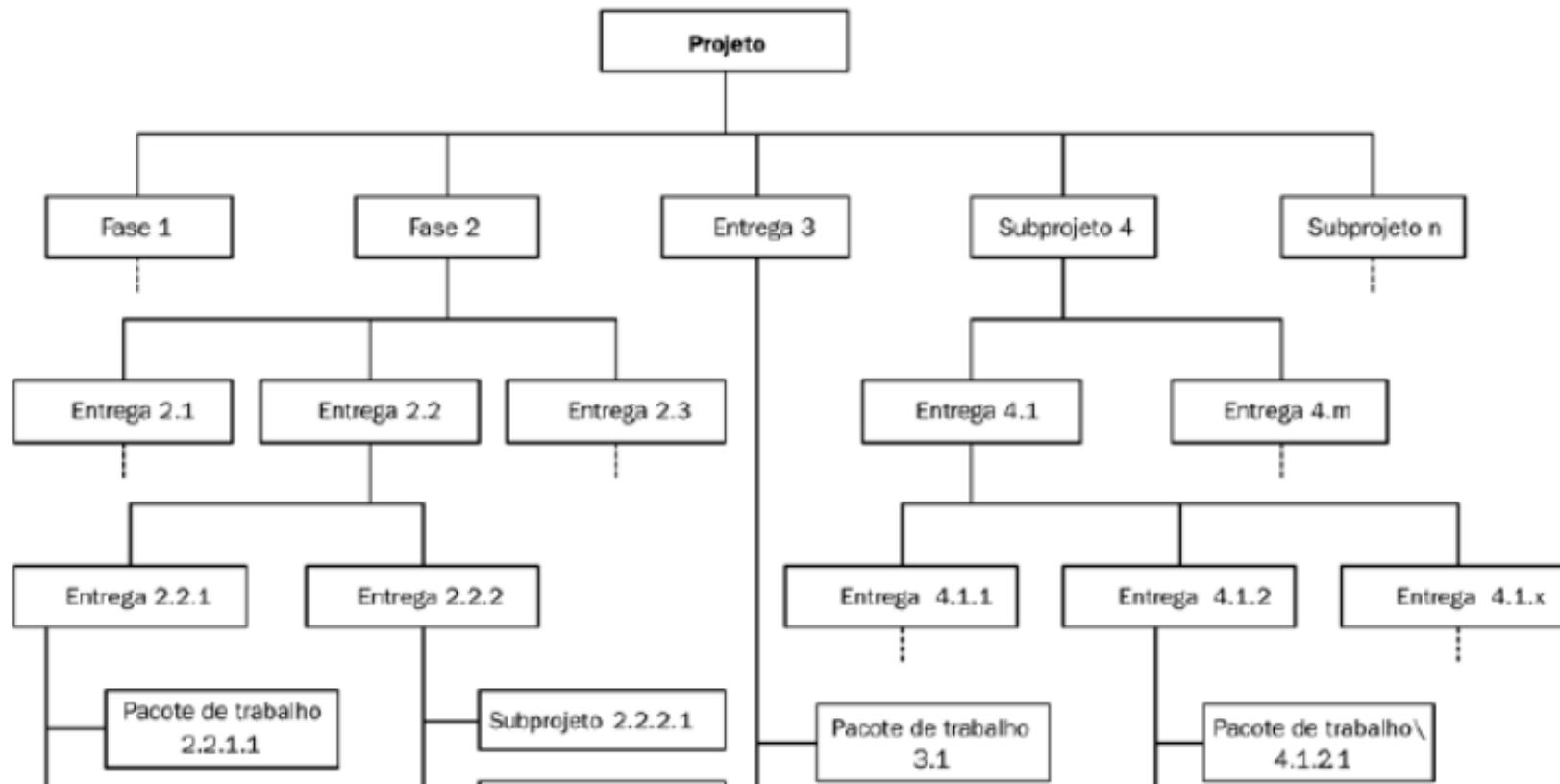
## - EAP



# Criar a estrutura analítica do projeto - EAP

Estrutura analítica do projeto (EAP) é uma decomposição hierárquica, **orientando às entregas** do trabalho a ser executado para atingir os objetivos do projeto.

# Exemplo de EAP



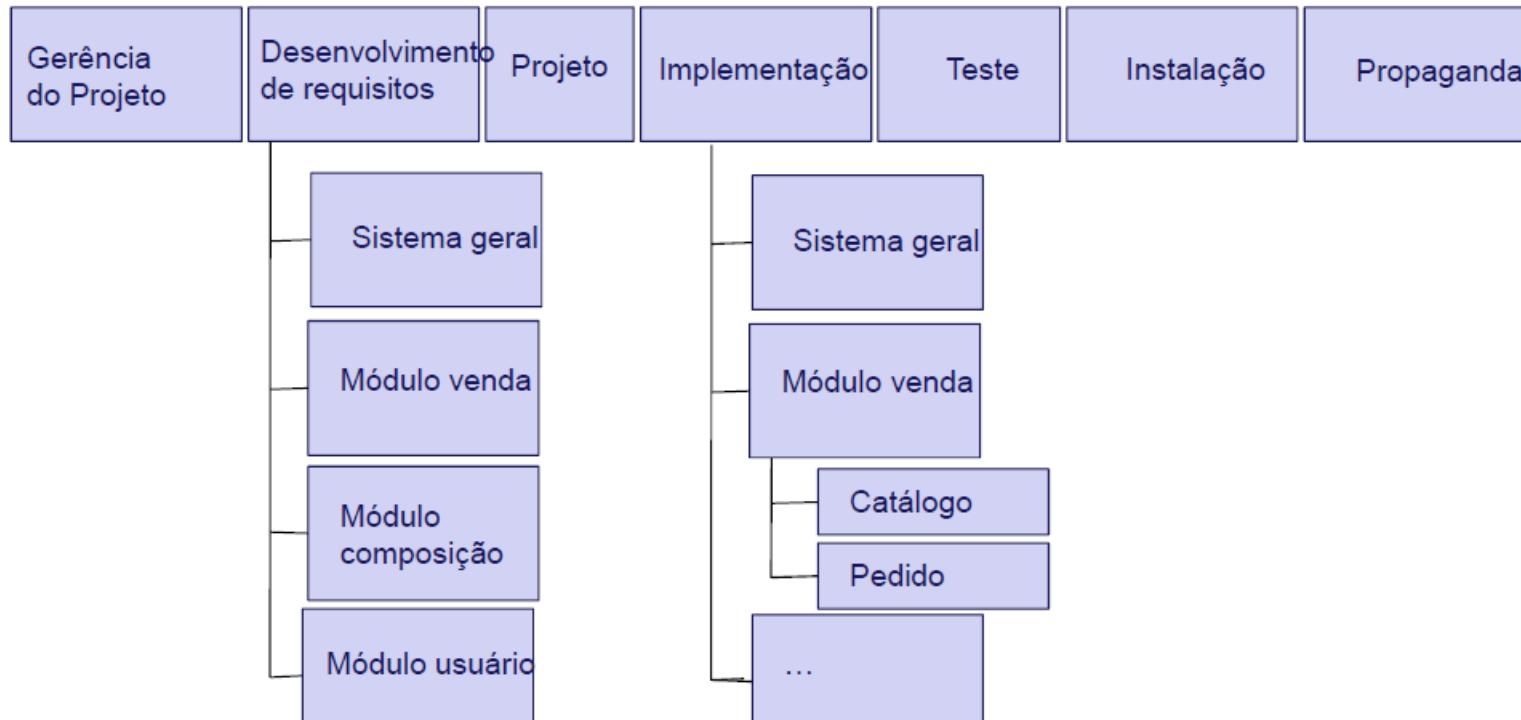
# EAP

- Cada nível descendente da EAP representa uma definição gradualmente mais detalhada da declaração do escopo do projeto.
- **Pacotes de trabalho:** o trabalho planejado é contido dentro dos componentes de nível mais baixo da EAP.
  - Um pacote de trabalho pode ser agendado, ter seu custo estimado, monitorado e controlado.
  - Tipicamente, pacotes de trabalho no EAP têm um esforço estimado entre 8 a 80 pessoa-horas.

# EAP

Pode ser criada de várias maneiras:

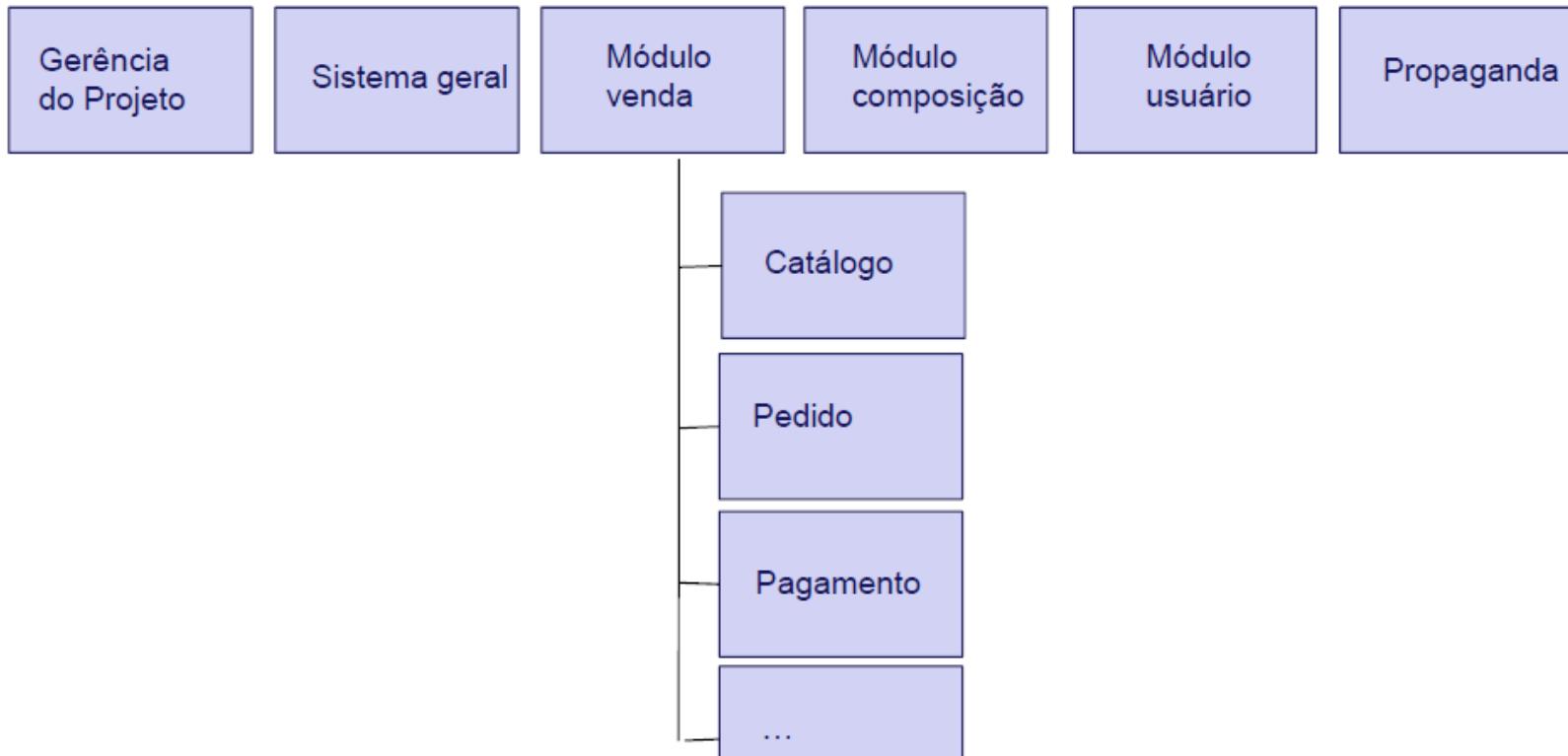
- ❑ 1. Nível: fases do ciclo de vida do projeto



# EAP

Pode ser criada de várias maneiras:

- ❑ 2. Nível: principais resultados



# Regra dos 100%

- ❑ A EAP representa **TODO** produto e trabalho do projeto.
  - ❑ inclusive o trabalho de gerência do mesmo.
- ❑ Todo o trabalho nos níveis mais baixos tem que estar dentro do trabalho nos níveis mais altos para que nada seja omitido.
- ❑ Trabalho que não está na EAP está fora do escopo do projeto.

# Identificar os itens da EAP

- Cada elemento de cada nível da EAP deve ser atribuído a um identificador único.
  - 1. EAP para Projeto de Implementação de Software
    - 1.1 Gerenciamento de Projetos
    - 1.2 Requisitos do Produto
      - 1.2.1 Requisitos do Software
        - 1.2.1.1 Rascunho de Requisitos de Software
        - 1.2.1.2 Requisitos de Software Final
        - 1.2.1.3 Aprovação de Requisitos de Software
      - 1.2.2 Documentação do Usuário
        - 1.2.2.1 Rascunho da Documentação do Usuário
        - 1.2.2.2 Documentação do Usuário Final
        - 1.2.2.3 Aprovação da Documentação do Usuário
      - 1.2.3 Material do Programa de Treinamento
        - 1.2.3.1 Requisitos de Treinamento Inicial
        - 1.2.3.2 Material de Treinamento Inicial
        - 1.2.3.3 Entrega Experimental do Curso
      - 1.2.4 Hardware
        - 1.2.4.1 Rascunho dos Requisitos de Hardware
        - 1.2.4.2 Requisitos de Hardware Final
        - 1.2.4.3 Aprovação dos Requisitos de Hardware
      - 1.2.5 Implementação e Suporte Futuros
    - 1.3 Detalhes do Projeto de Software
      - 1.3.1 Projeto de Software Inicial
- Tipicamente usando números
- Código de contas: Normalmente associado a custo de contas da administração financeira.

# Exemplo simplificado: Projeto de conclusão de curso

1. Gerência do TCC
2. Relatório Projeto I
  - 2.1 Introdução
  - 2.2 Fundamentação teórica
  - 2.3 Estado da arte e prática
  - 2.4 Proposta da solução
3. Relatório Projeto II
  - 3.1 Solução ...
  - 3.2 Avaliação
  - 3.3 Discussão e conclusão
4. Defesa

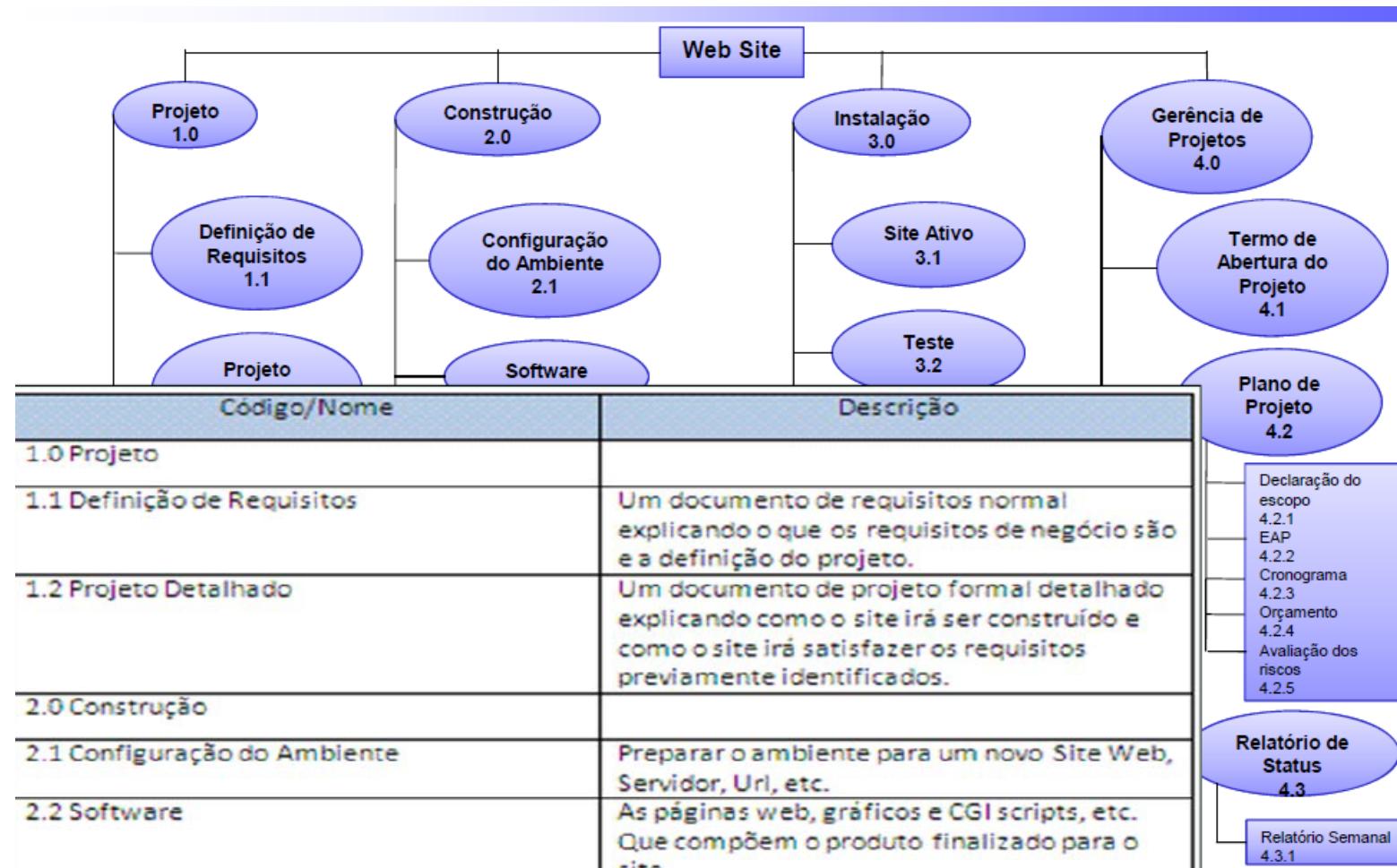
# Detalhamento da EAP

- ❑ Muitos elementos EAP são vagos e precisam ser explicados em mais detalhes para pessoas saberem o que fazer, e poder estimar quanto tempo o trabalho levará e quanto irá custar.
- ❑ **Dicionário da EAP:** documento que descreve cada componente da EAP.
- ❑ Inclui:
  - ❑ Escopo/declaração do trabalho
  - ❑ Entrega(s)
  - ❑ Lista de atividades associadas
  - ❑ Lista de marcos
  - ❑ Responsável
  - ❑ Datas de início e de conclusão
  - ❑ Recursos necessários
  - ❑ Estimativa de custo

# Detalhamento da EAP

DICIONÁRIO EAP			
Id de controle de conta	Pacote de trabalho	Data de atualização	Responsabilidade da Organização/Individual
Descrição do pacote de trabalho			
Critério de aceitação (como saber se o trabalho é aceitável)			
Entregáveis			
Suposições			
Recursos alocados			
Duração			
Marcos			
Custo			
Data combinada			
Interdependências Antes deste pacote de trabalho _____			
Depois deste pacote de trabalho _____			
Aprovado por: Gerente de Projetos _____		Data: _____	

# Exemplo de dicionário da EAP



# Exemplo

- A Estrutura Analítica de Projeto (EAP) constitui-se em uma importante ferramenta para o desenvolvimento de projetos.
- Assinale a alternativa que representa uma característica dessa estrutura.



Seu nível mais baixo representa um pacote de trabalho, com um identificador único.

B

Uma conta de controle representa uma parte de um pacote de trabalho.

C

Cada pacote de trabalho pode ser associado a várias contas de controle.

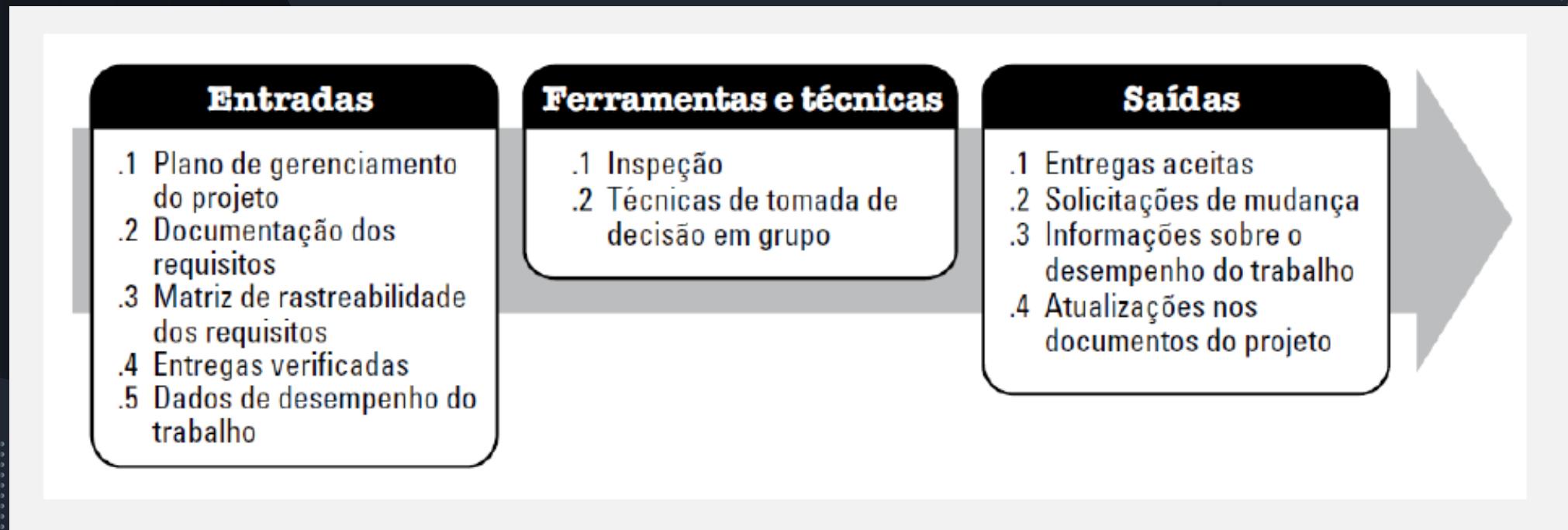
D

Uma conta de controle pode incluir um único pacote de planejamento.

E

O dicionário da EAP contém apenas o nome do projeto e sua descrição.

# Validar o escopo



# Baseline do escopo

- ❑ ***Baseline*** do escopo: uma versão específica aprovada incluindo:
    - ❑ Declaração do escopo do projeto
    - ❑ EAP
    - ❑ Dicionário da EAP
  - ❑ ***Baselines***: estabelecem uma referência pela qual o progresso real do projeto é medido.
  - ❑ Tipos de ***Baseline*** incluem: escopo, cronograma, custo.
  - ❑ Uma vez aprovada, mudanças só podem ser feitas através de solicitações formais de mudanças.

# Atividade: Elaborar EAP para o projeto.

## **Incluir:**

- EAP gráfica
- Dicionário da EAP
- Detalhamento para cada pacote de trabalho



# Aula 8

Planejamento de tempo do projeto  
(cronograma)

# Tempo | Cronograma

Planejar o gerenciamento de tempo

Definir as atividades

Sequenciar as atividades

Estimar os recursos das atividades

Estimar a duração das atividades

Desenvolver o cronograma

# Definir as atividades

## Entradas

- .1 Plano de gerenciamento do cronograma
- .2 Linha de base do escopo
- .3 Fatores ambientais da empresa
- .4 Ativos de processos organizacionais

## Ferramentas e técnicas

- .1 Decomposição
- .2 Planejamento em ondas sucessivas
- .3 Opinião especializada

## Saídas

- .1 Lista de atividades
- .2 Atributos das atividades
- .3 Lista de marcos



# Definir as atividades

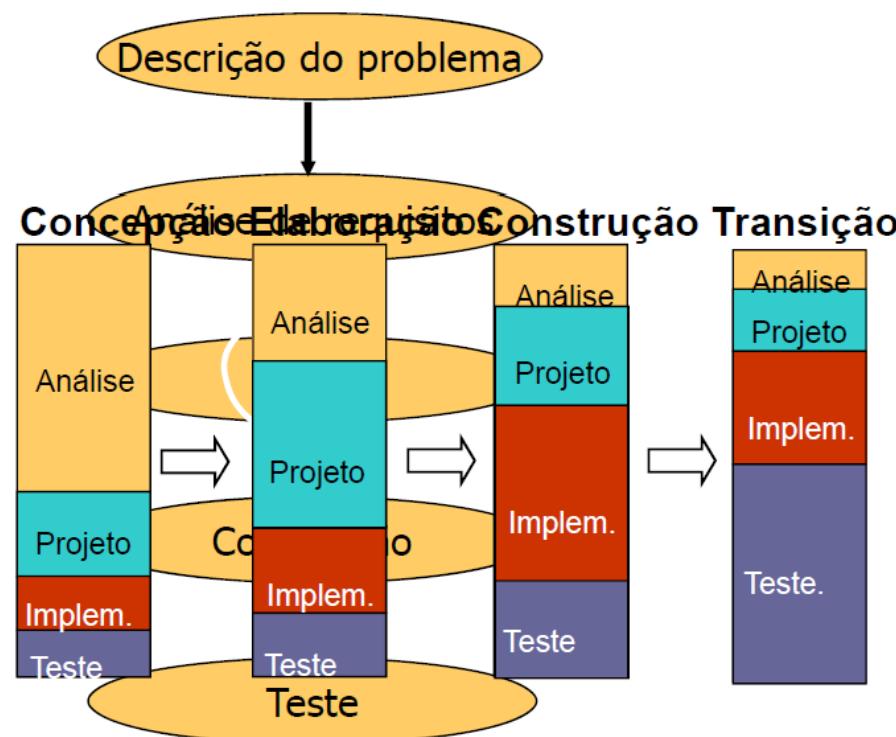
- ❑ **Ciclo de vida do projeto:** determina as fases ( e a sua sequência) de acordo com o escopo de requisitos, as estimativas para os recursos do projeto e a natureza do projeto.

Modelo cascata

Modelo de prototipação

Modelo iterativo/incremental

Modelo espiral ...

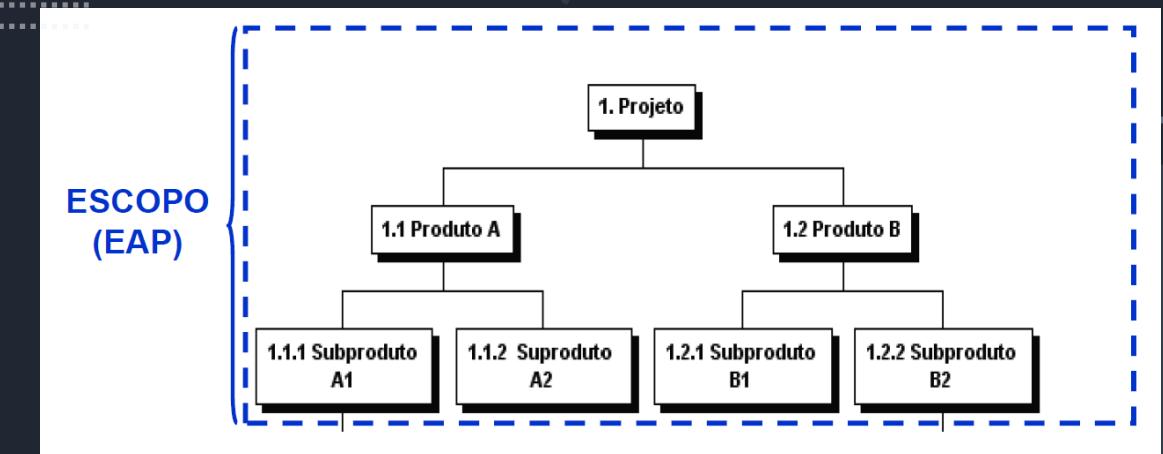


# Definir as atividades

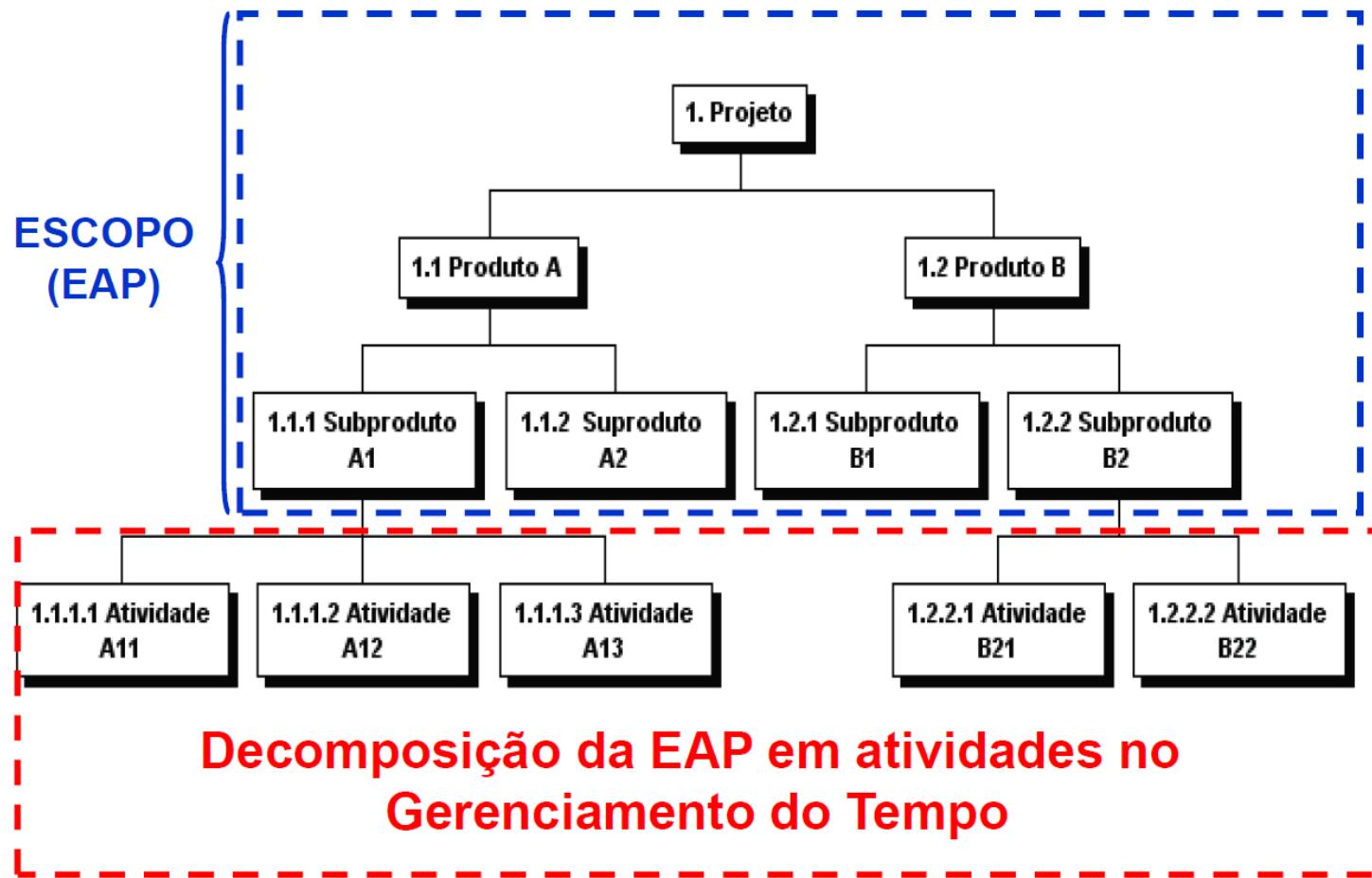
- ❑ Identificação das ações específicas a serem realizadas para produzir as entregas do projeto.
- ❑ Tipicamente os pacotes de trabalho são compostos em componentes menores ⇒ **atividades**
- ❑ Atividades representam o trabalho necessário para completar o pacote de trabalho.
- ❑ As atividades e principalmente a sua sequência depende do modelo de ciclo de vida.

# Definir as atividades: Entrada

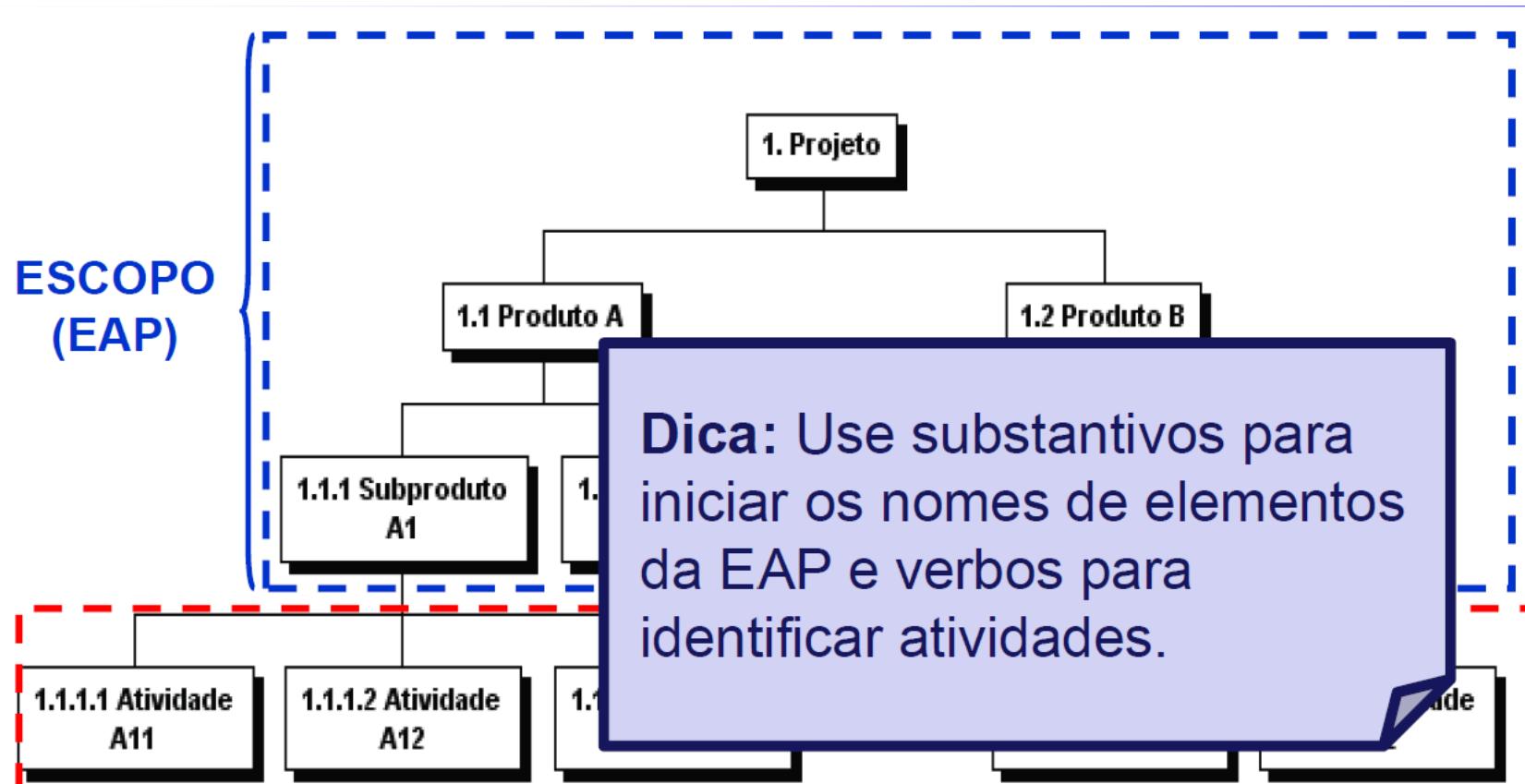
- Para definir as atividades é necessário ter um escopo conhecido.
- As atividades são as ações ("o como") necessárias para entregar um resultado (escopo/ "o quê")



# Definir as atividades



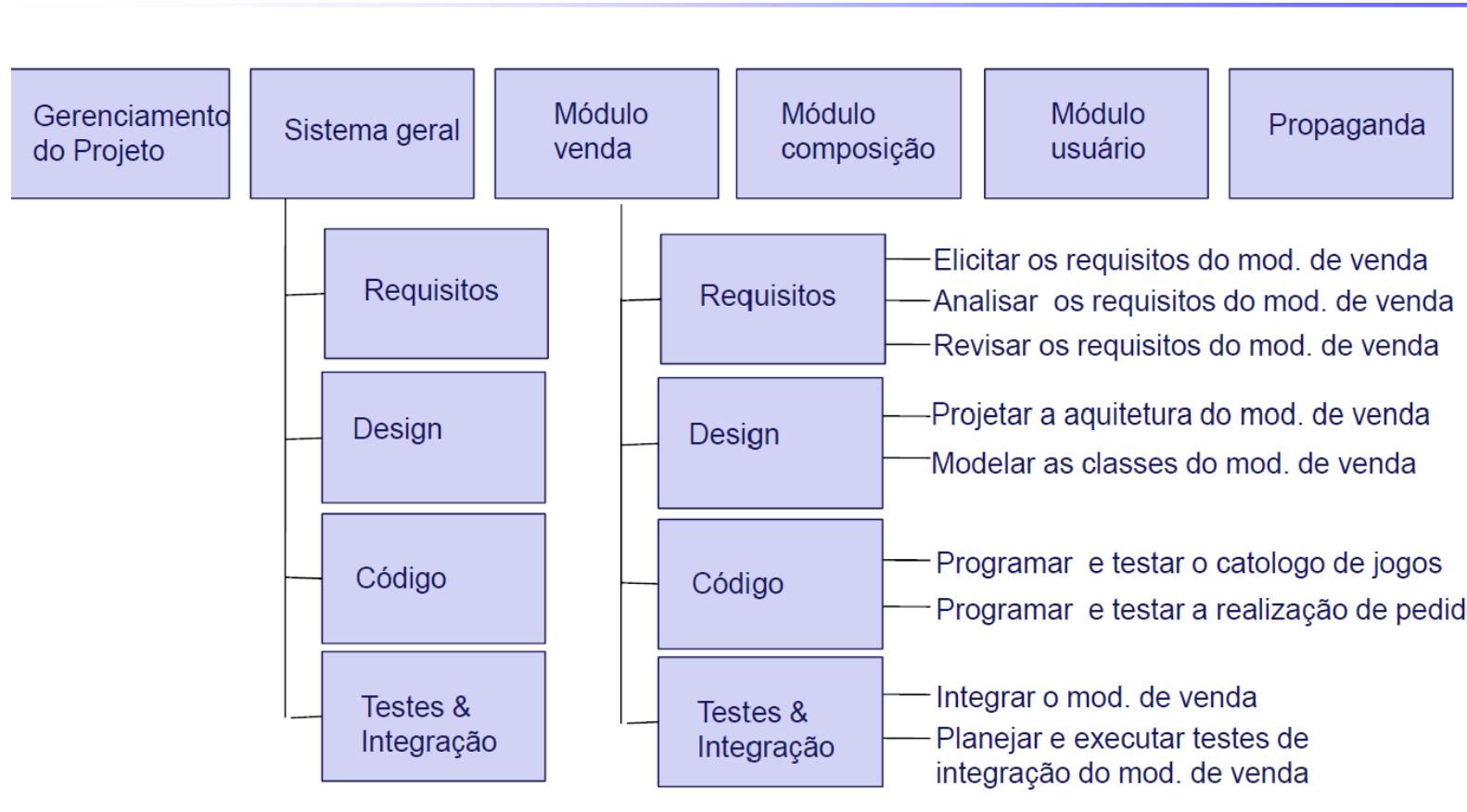
# Definir as atividades



# Definir as atividades - Resultados

- **Lista das atividades:** inclui todas as atividades necessárias no projeto.
  - Identificador e descrição de cada atividade em detalhe suficiente para os membros da equipe entendam que o trabalho precisa ser feito.
- **Lista dos marcos:** identifica todos os marcos do projeto.

# Definir as atividades - Exemplo



# Sequenciar as atividades

Planejar o gerenciamento de tempo

Definir as atividades

Sequenciar as atividades

Estimar os recursos das atividades

Estimar a duração das atividades

Desenvolver o cronograma

## Entradas

- .1 Plano de gerenciamento do cronograma
- .2 Lista de atividades
- .3 Atributos das atividades
- .4 Lista dos marcos
- .5 Especificação do escopo do projeto
- .6 Fatores ambientais da empresa
- .7 Ativos de processos organizacionais

## Ferramentas e técnicas

- .1 Método do diagrama de precedência (MDP)
- .2 Determinação de dependência
- .3 Antecipações e esperas

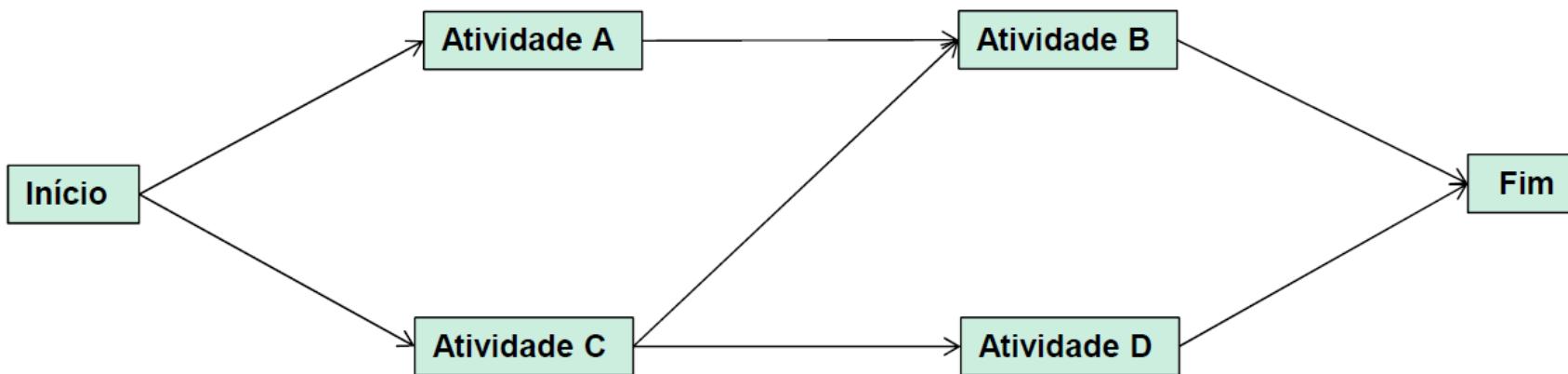
## Saídas

- .1 Diagramas de rede do cronograma do projeto
- .2 Atualizações nos documentos do projeto

Sequenciar as atividades

# Sequenciar as atividades

- Identificação e documentação dos relacionamentos entre atividades do projeto identificando predecessores e sucessores.
  - **Método do diagrama de precedência (MDP):** construção da rede do cronograma do projeto mostrando as atividades nos nós.
  - **Diagramas de rede:** demonstração esquemática dos relacionamentos lógicos entre as atividades do cronograma do projeto.

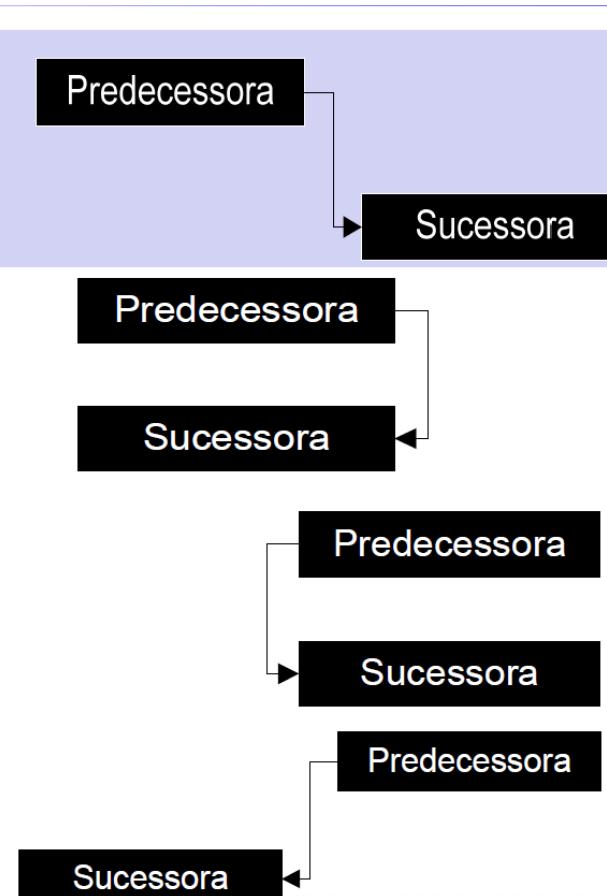


# Exemplo simplificado

ID	<b>Lista de atividades</b>	<b>Atividade(s) precedente(s)</b>
3.1	Elicitar os requisitos do mod. de venda	---
3.2	Analizar os requisitos do mod. de venda	3.1
3.3	Revisar os requisitos do mod. de venda	3.2
3.4	Projetar arquitetura do mod. de venda	3.3
3.5	Modelar as classes do mod. de venda	3.4
3.6	Programar e testar o catalogo de jogos	3.5
3.7	Programar e testar a realização de pedidos	3.5
3.8	Integrar o mod. de venda	3.6, 3.7
3.9	Planejar e executar testes de integração do mod. de venda	3.8

# Tipos de dependência

- Término para início (TI): início da atividade sucessora depende do término da atividade predecessora. (o mais comum)
- Término para término (TT): término da atividade sucessora depende do término da atividade predecessora.
- Início para início (II): início da atividade sucessora depende do início da atividade predecessora.
- Início para término (IT): término da atividade sucessora depende do início da atividade predecessora.



Estimar os  
recursos das  
atividades

Planejar o gerenciamento de tempo  
Definir as atividades  
Sequenciar as atividades  
**Estimar os recursos das atividades**  
Estimar a duração das atividades  
Desenvolver o cronograma

# Estimar os recursos das atividades



## Estimar os recursos das atividades

Estimar **tipos e quantidades** de pessoas, material, equipamentos ou outros recursos que serão necessários para realizar cada atividade.



# Lista de recursos

Item	Quantidade
<b>Pessoal</b>	
Coordenador	1 (10 horas/mês por 12 meses)
Pesquisador sênior	2 (40 horas/mês por 12 meses)

## Equipamentos e Material permanente

Notebook	2
Projetor	1
Projetor multímeia	1

Que recursos tipicamente precisamos para projetos de software?

# Estimar recursos das atividades

- ❑ Como fazer?

- ❑ Opinião especializada
- ❑ Análise de alternativas
- ❑ Dados publicados para auxílio a estimativas
- ❑ Estimativa bottom-up

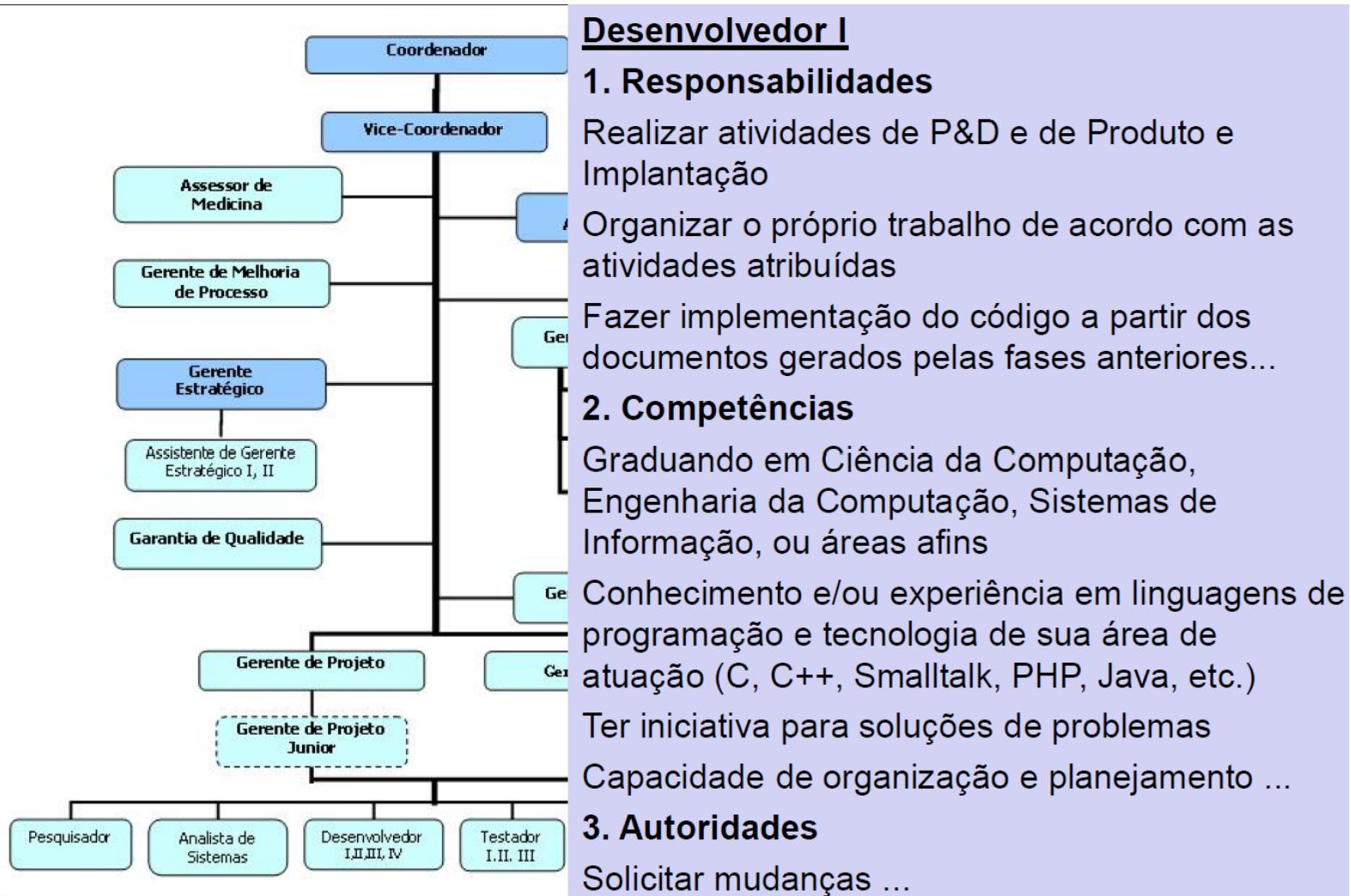
- ❑ Entradas:

- ❑ Organograma
- ❑ Estrutura analítica de recursos

# Organograma

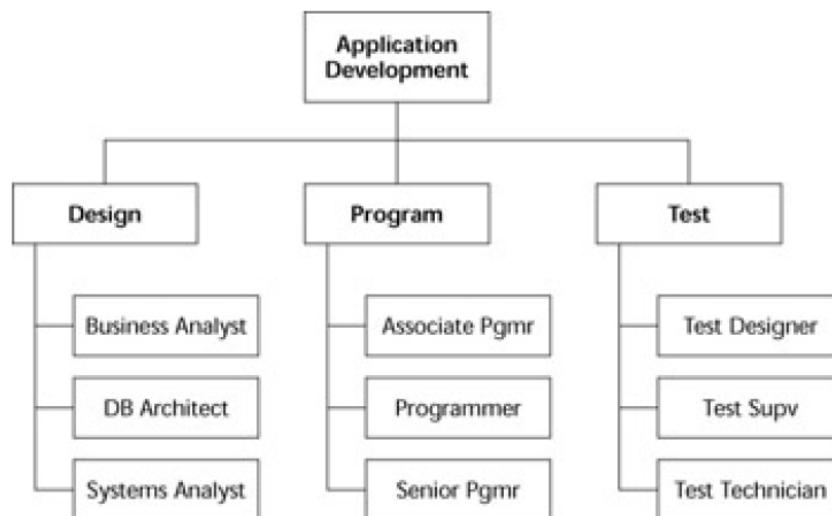
**Organograma:** representa a estrutura formal de uma organização mostrando tipicamente hierarquia de papéis, responsabilidades, competências e autoridades.

# Organograma



# Organograma

- **Estrutura analítica dos recursos:** estrutura hierárquica dos recursos identificados organizada por categoria e tipo de recursos.
  - Categorias: mão de obra, material, equipamento, ...
  - Tipos: nível de habilidade, etc.



# Exemplo: estimativa de recursos

ID	Lista de atividades	Atividade(s) precedente(s)	Recursos humanos
3.1	Elicitar os requisitos do mod. de venda	---	Analista sênior, Diretor
3.2	Analisar os requisitos do mod. de venda	3.1	Analista sênior e Analista junior
3.3	Revisar os requisitos do mod. de venda	3.2	Analista sênior, Diretor
3.4	Projetar arquitetura do mod. de venda	3.3	Projetista
3.5	Modelar as classes do mod. de venda	3.4	Projetista, Programador sênior
3.6	Programar e testar o catalogo de jogos	3.5	2 Programadores junior
3.7	Programar e testar a realização de pedidos	3.5	Programador sênior, Programador junior
3.8	Integrar o mod. de venda	3.6, 3.7	Programador sênior
3.9	Planejar e executar testes de integração do mod. de venda	3.8	Testador

# Atividade: Definição das atividades.

## **Incluir:**

- Definição das atividades por pacote de trabalho da EAP
- Sequenciamento das atividades
- Estimativa de recursos por atividade

# Estimar a duração das atividades

Planejar o gerenciamento de tempo

Definir as atividades

Sequenciar as atividades

Estimar os recursos das atividades

Estimar a duração das atividades

Desenvolver o cronograma

### **Entradas**

- .1 Plano de gerenciamento do cronograma
- .2 Lista de atividades
- .3 Atributos das atividades
- .4 Requisitos de recursos das atividades
- .5 Calendários do recurso
- .6 Especificação do escopo do projeto
- .7 Registro dos riscos
- .8 Estrutura analítica dos recursos
- .9 Fatores ambientais da empresa
- .10 Ativos de processos organizacionais

### **Ferramentas e técnicas**

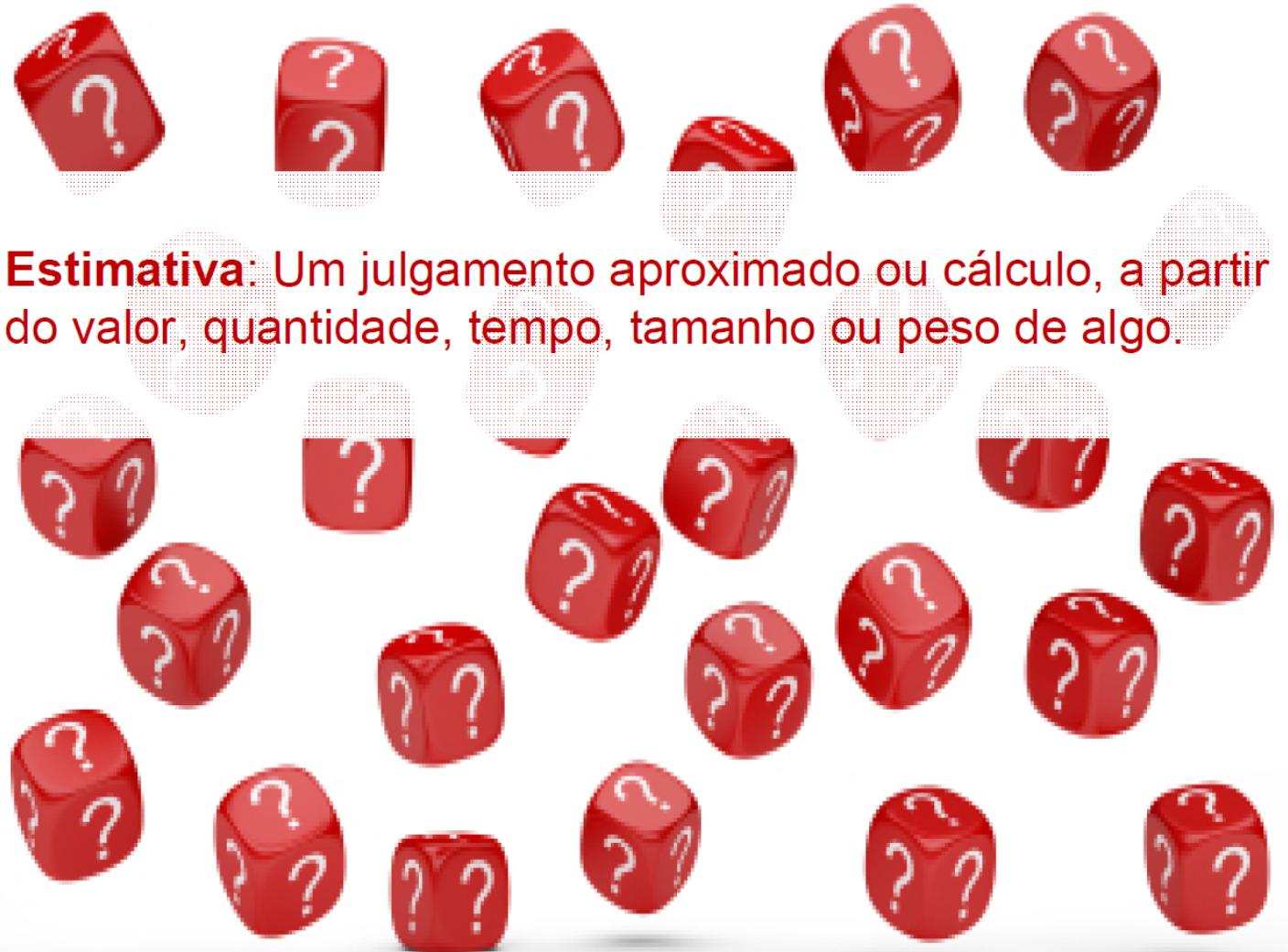
- .1 Opinião especializada
- .2 Estimativa análoga
- .3 Estimativa paramétrica
- .4 Estimativa de três pontos
- .5 Técnicas de tomada de decisões em grupo
- .6 Análise de reservas

### **Saídas**

- .1 Estimativas de duração das atividades
- .2 Atualizações nos documentos do projeto

# Estimar a duração das atividades

# Estimar a duração das atividades



**Estimativa:** Um julgamento aproximado ou cálculo, a partir do valor, quantidade, tempo, tamanho ou peso de algo.

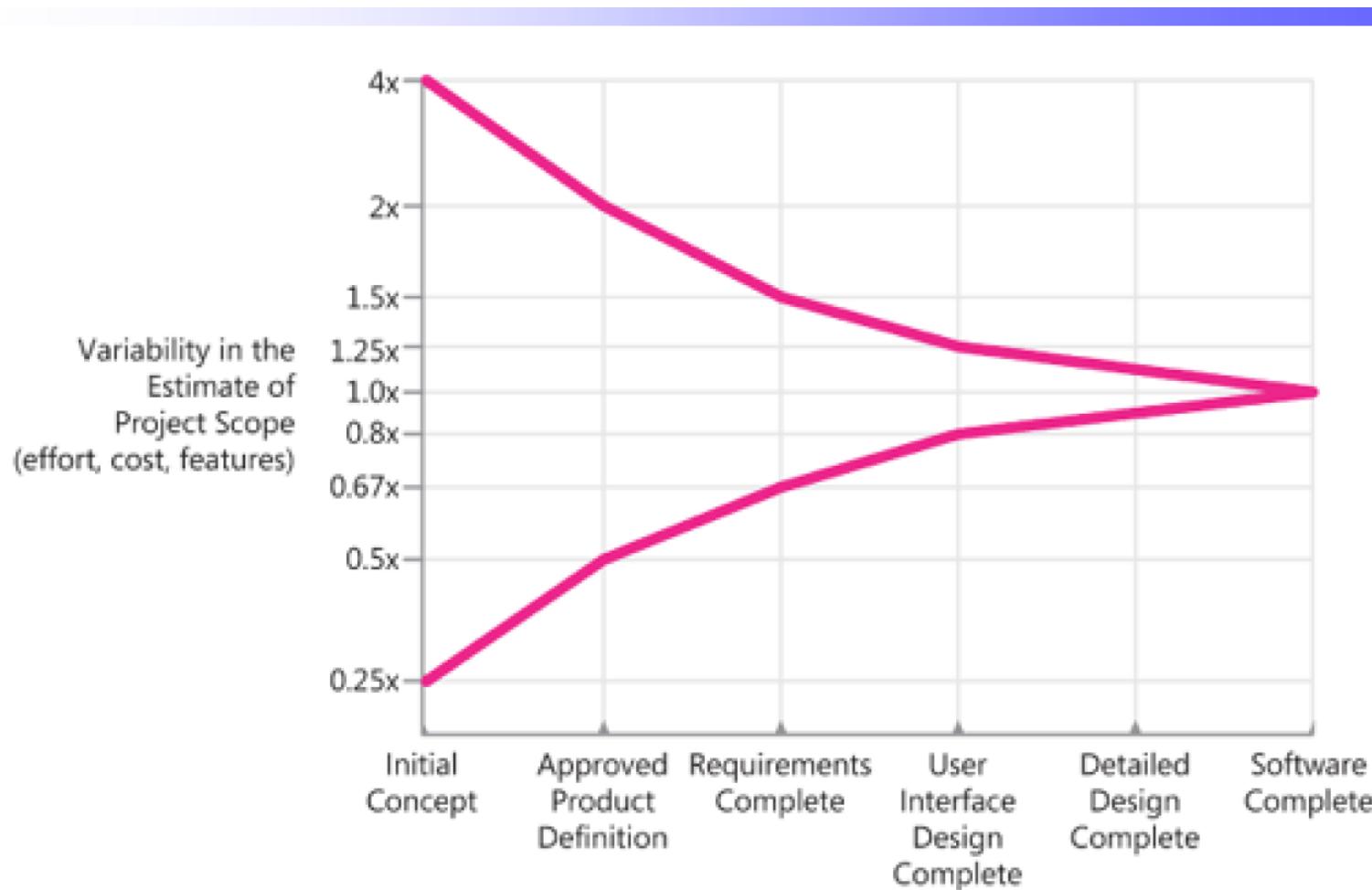
# O que estimar?

- I. **Tamanho/complexidade** do sistema, serviços ou resultado a ser desenvolvido (LOC, PF, PCU, ...)
- II. **Esforço da atividade**: quantidade de trabalho necessário para realizar uma atividade (pessoas-hora, pessoa-mês, ...)
- III. **Duração da atividade**: Quantidade de tempo que decorre entre o início e o término da atividade (horas, meses, ...)

## Esforço X Duração

- Esforço estimado: 16 pessoa-horas
- 2 funcionários alocados 4h/dia -> Duração: 2 dias

# Cone da incerteza



[S. McConnell. Software Estimation: Demystifying the Black Art. Microsoft Press, 2006]

# Técnicas para estimativas

- **Analogia:** baseado em dados históricos da organização comparando o projeto com projetos semelhantes.
- **Baseado em modelos algorítmicos/ paramétricos:** baseado em funções matemáticas de atributos de produto, projeto e processo .
  - Modelos genéricos (p.ex. COCOMO), PF, PCU
  - Modelos específicos com base em dados históricos da própria organização
- **Opinião Especializada:** especialistas utilizam sua experiência e intuição para estimar.
  - *Planning Poker*
  - *Wideband delphi*

# Opinião de especialistas

- ❑ Consultando um ou mais especialistas



- ❑ **Consenso em grupo:** experiência de uso de várias pessoas para chegar a uma estimativa

- ❑ *Planning Poker*

- ❑ *Wideband Delphi*

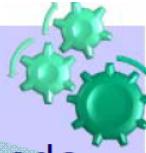
# Planning poker

## Planning Poker

- Técnica ágil de estimativa.
- Várias iterações para chegar num consenso.
- Evita influência de uma estimativa em outra.

[M. Cohn. Agile Estimating and Planning. Prentice Hall PT, 2005]

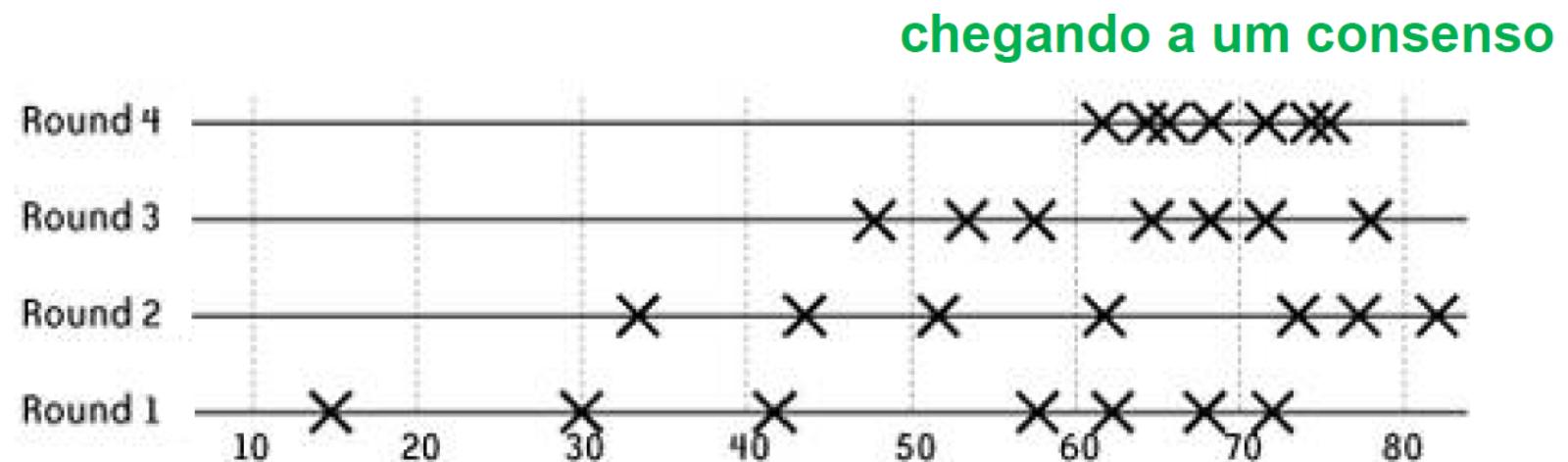
# Planning poker



## Como fazer?

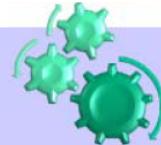
1. O desenvolvedor com maior conhecimento sobre uma determinada funcionalidade/história dá uma breve explicação da funcionalidade. Para a equipe é dada a oportunidade de fazer perguntas e discutir para clarificar premissas e riscos.
2. Cada indivíduo dispõe de uma carta virada para baixo que representa sua estimativa. Unidades utilizadas variam, pode ser dias de duração, dias ideais ou pontos de história.
3. Todos viram as cartas simultaneamente.
4. Pessoas com estimativas muito altas e baixas justificam as suas estimativas e a discussão continua.
5. O processo de estimativa é repetido até um consenso ser alcançado.

# Planning poker – estimativa por consenso



# Wideband Delphi

Como fazer?



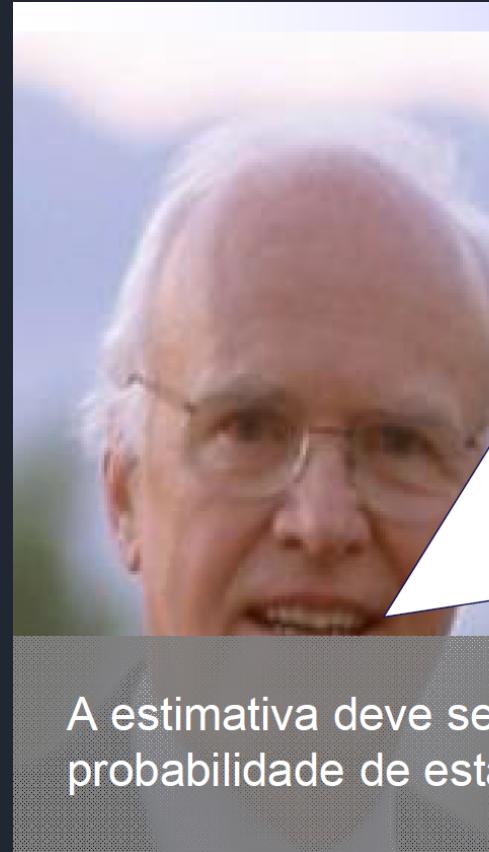
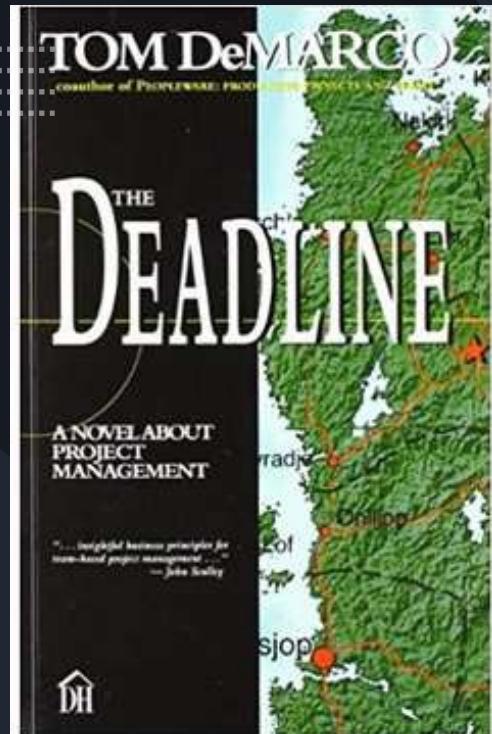
1. Coordenador apresenta aos especialistas o escopo e um formulário para as estimativas. Estimadores esclarecem dúvidas com o coordenador.
2. Cada estimador faz individualmente as estimativas e entrega para o coordenador de forma anônima.
3. Coordenador analisa as estimativas individuais.
4. Estimadores recebem como *feedback* um resumo dos resultados.
5. Estimadores se reúnem para discutir diferenças.
6. Repetido até chegar a um consenso.

[A. Stellman, J. Greene. Applied Software Project Management. O'Reilly, 2005]

# Exemplo simplificado: estimativa de esforço

ID	Lista de atividades	Atividade(s) precedente(s)	Estimativa de esforço (pessoas-horas)
3.1	Elicitar os requisitos do mod. de venda	---	40
3.2	Analizar os requisitos do mod. de venda	3.1	100
3.3	Revisar os requisitos do mod. de venda	3.2	40
3.4	Projetar arquitetura do mod. de venda	3.3	40
3.5	Modelar as classes do mod. de venda	3.4	80
3.6	Programar e testar o catalogo de jogos	3.5	200
3.7	Programar e testar a realização de pedidos	3.5	200
3.8	Integrar o mod. de venda	3.6, 3.7	20
3.9	Planejar e executar testes de integração do mod. de venda	3.8	40

# Nossas estimativas são tendenciosas?



Tom DeMarco

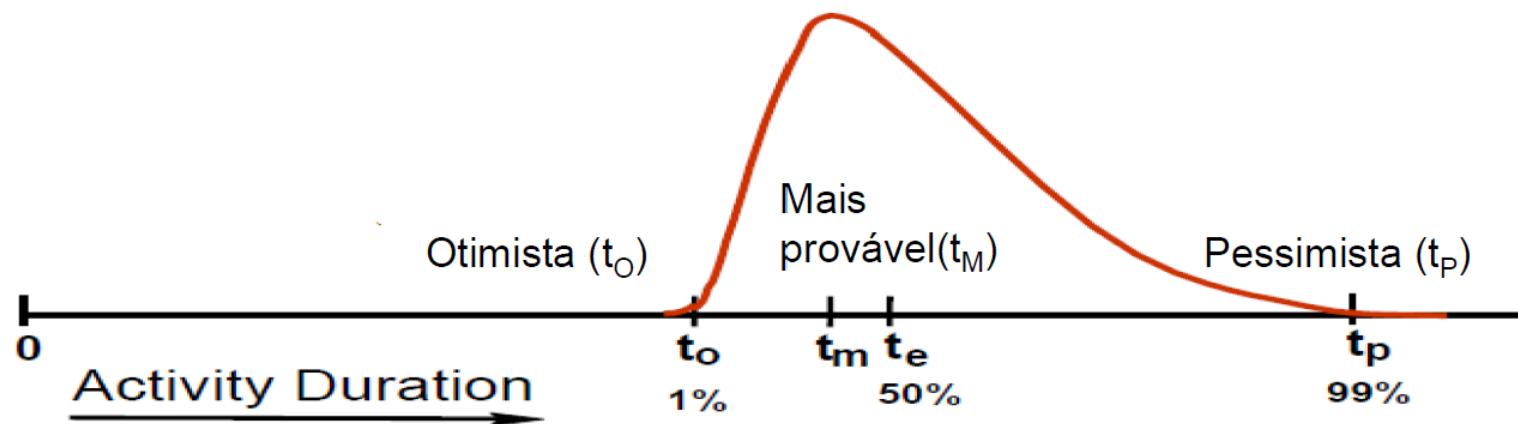
Imagine que você está solicitando uma estimativa de um de seus funcionários, e em seguida pergunta: "Qual é a probabilidade de terminar 20% mais tarde do que sua estimativa?" Neste caso, você receberá algum tipo de número razoável, mostrando que o estimador é uma pessoa do mundo real levando em conta os fatos tristes da vida. Agora pergunta: "E, qual é a probabilidade de terminar 20% mais cedo do que a sua estimativa?" Muito provavelmente, seu estimador olhará para você de forma confusa, como se você tivesse

...deveria recentemente de outro planeta.

A estimativa deve ser uma previsão, que tem a mesma probabilidade de estar acima ou abaixo do resultado real.

# Estimativas dos 3 pontos

- Para melhorar a precisão das estimativas considerando as incertezas e riscos.
- Origem na Técnica de Revisão e Avaliação de Programa (PERT)
- Usa três estimativas



Cálculo da duração esperada ( $t_e$ ) usando a média ponderada

$$t_e = \frac{t_p + 4t_m + t_o}{6}$$

# Exercício

□ Calcule a duração esperada dado as seguintes estimativas:

Atividade	$t_o$	$t_m$	$t_p$
A	14	27	47
B	41	60	89

# Estimar a duração das atividades

- ❑ Se a estimativa for esforço ou outra unidade, deve ser convertida em duração.
- ❑ Duração = Esforço / Recursos (às vezes)
- ❑ Levando em consideração:
  - ❑ Disponibilidade de recursos / calendários
  - ❑ Competência
  - ❑ Produtividade
- ❑ Análise de reservas: incluindo reservas para contingências (*buffers*) para considerar incertezas.
  - ❑ Pode ser uma porcentagem da duração estimada da atividade
  - ❑ Número fixo de períodos de trabalho
- ❑ Para mostrar incertezas, podem incluir indicações da faixa de resultados:
  - ❑ 2 semanas  $\pm$  2 dias
  - ❑ 15% probabilidade de exceder 3 semanas

# Exemplo: estimativa por analogia

A Alfa é uma empresa que fornece soluções de software a pedido dos vários clientes. Maria é a gerente do projeto PRI. Com um aumento da demanda por novos projetos na Alfa, Maria foi designada pelo escritório de projetos como gerente do projeto SEG, acumulando mais uma função. Embora o projeto PRI seja pequeno, o SEG tem objetivos distintos e parece estar crescendo a cada dia. Após algum tempo, Maria descobre que, no passado, a Alfa já havia atendido uma demanda semelhante à do projeto SEG.

Com o intuito de executar suas funções da melhor forma, Maria deve:

- A gerenciar os projetos PRI e SEG como parte do mesmo programa;
- B agrupar os projetos PRI e SEG como parte do mesmo portfólio;
- C reunir os stakeholders dos projetos PRI e SEG para comunicar um aumento de prazo;
- D unir as equipes dos projetos PRI e SEG e redefinir um cronograma único para ambos os projetos;
- ✓ buscar registros históricos do projeto semelhante ao SEG junto ao escritório de projetos.

# Desenvolver o cronograma

Planejar o gerenciamento de tempo

Definir as atividades

Seqüenciar as atividades

Estimar os recursos das atividades

Estimar a duração das atividades

Desenvolver o cronograma

# Desenvolver o cronograma

## Entradas

- .1 Plano de gerenciamento do projeto
- .2 Cronograma do projeto
- .3 Dados de desempenho do trabalho
- .4 Calendário do projeto
- .5 Dados do cronograma
- .6 Ativos de processos organizacionais

## Ferramentas e técnicas

- .1 Análise de desempenho
- .2 *Software* de gerenciamento de projetos
- .3 Técnicas de otimização de recursos
- .4 Técnicas de desenvolvimento de modelos
- .5 Antecipações e esperas
- .6 Compressão de cronograma
- .7 Ferramenta de cronograma

## Saídas

- .1 Informações sobre o desempenho do trabalho
- .2 Previsões de cronograma
- .3 Solicitações de mudança
- .4 Atualizações no plano de gerenciamento do projeto
- .5 Atualizações nos documentos do projeto
- .6 Atualizações nos ativos de processos organizacionais



# Desenvolver o cronograma

**Cronograma do projeto:** as datas planejadas para realizar as atividades do cronograma e para atingir os marcos do cronograma.

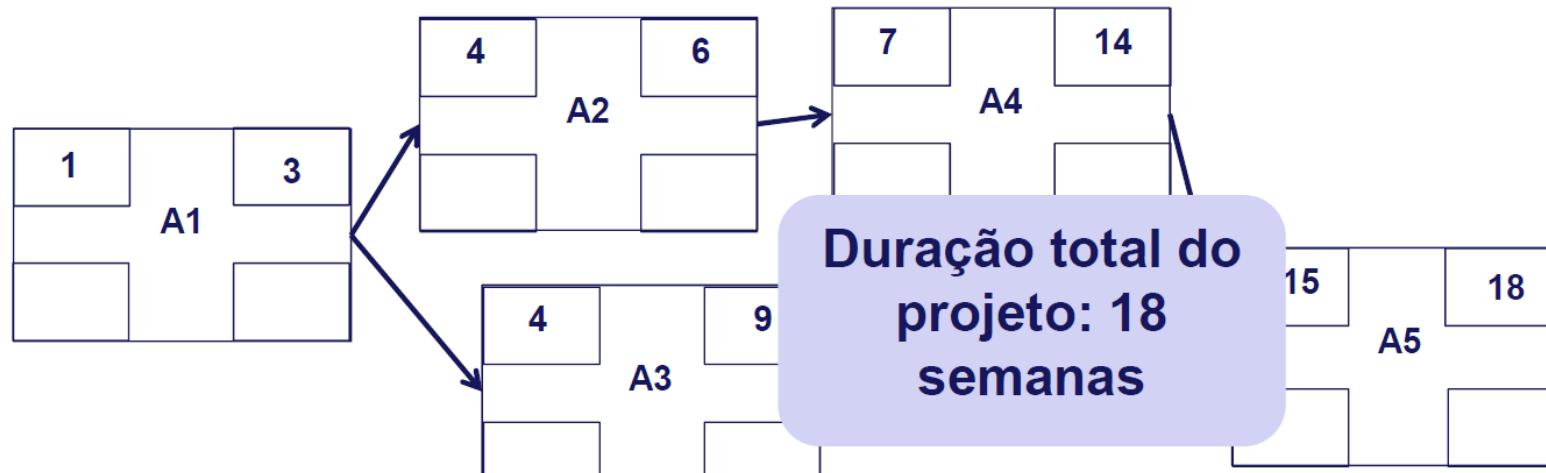
- Processo de análise de sequências das atividades, suas durações, recursos necessários e restrições ao cronograma visando criar o cronograma do projeto.
  - Análise da rede do cronograma
  - Método do caminho critico
  - Nivelamento de recursos
  - Método de corrente critica
  - Análise do cenário “E – se”

# Desenvolver o cronograma

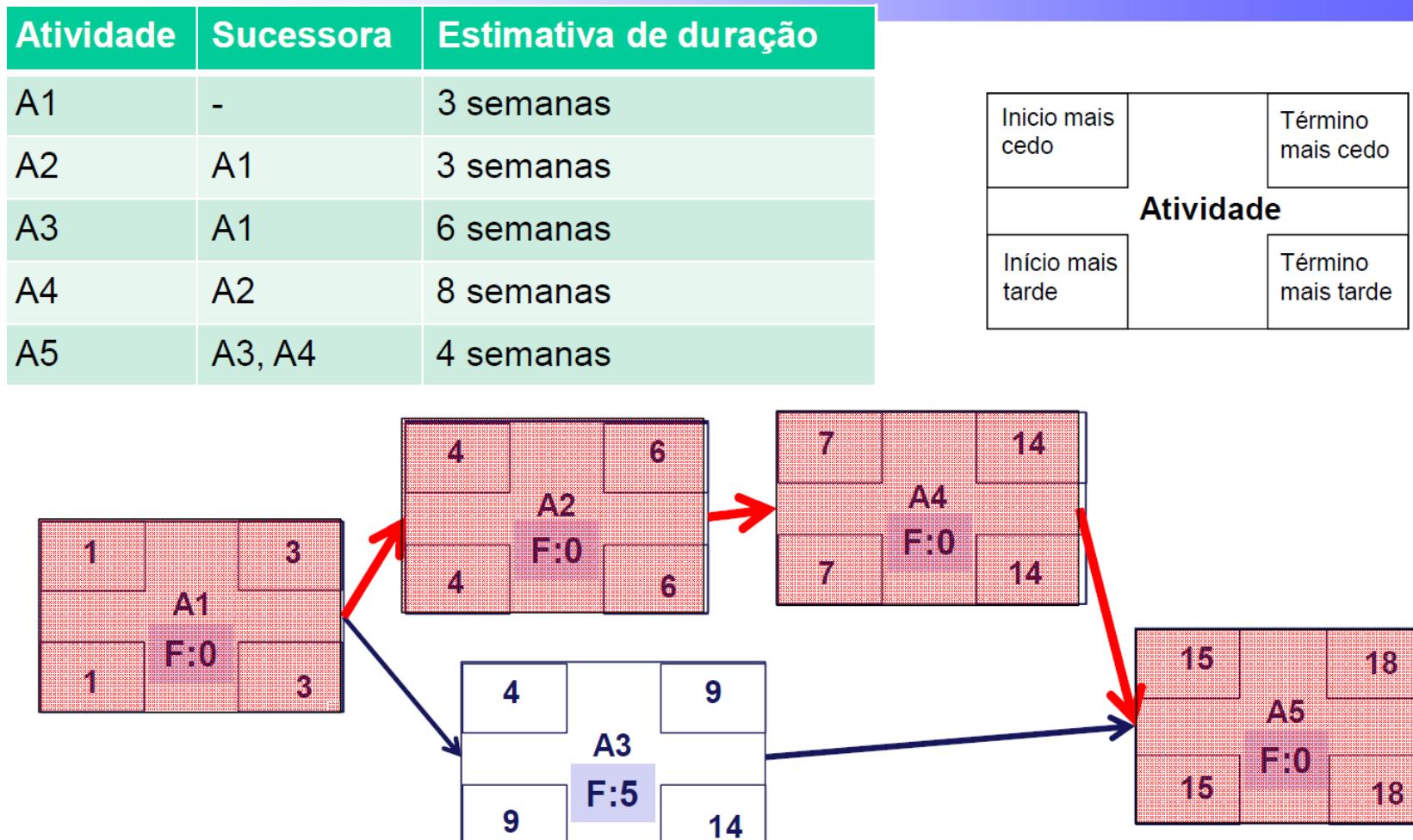


# Análise de rede do cronograma

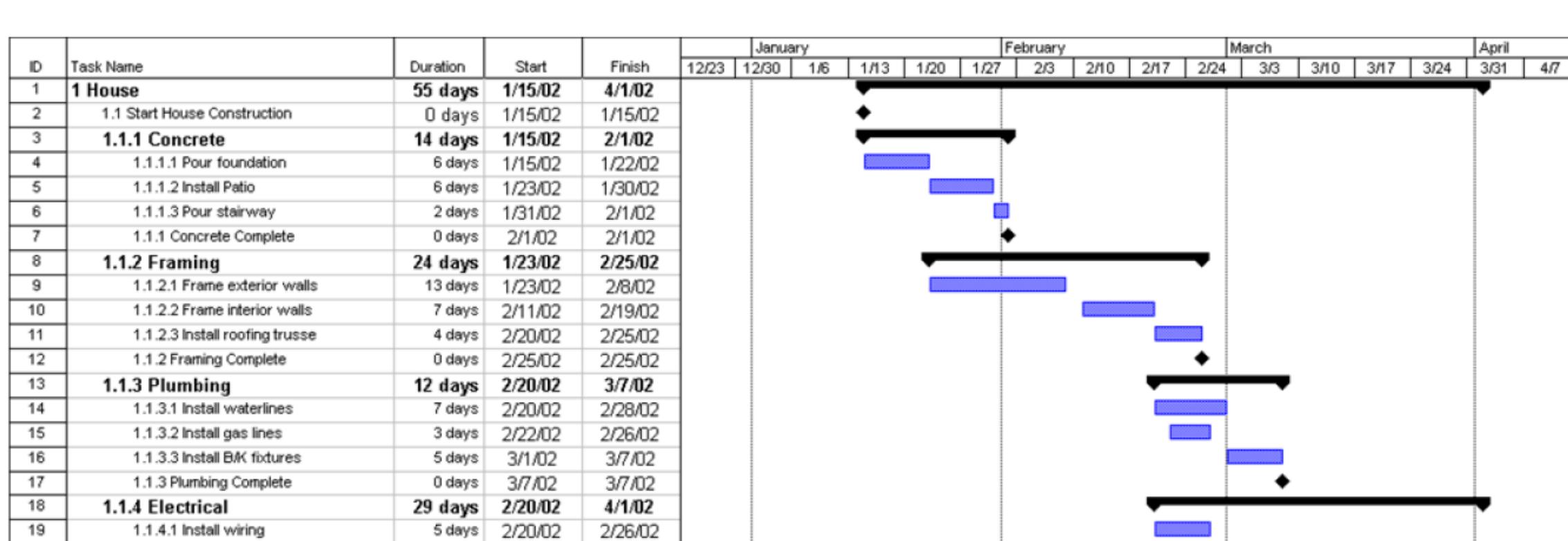
Atividade	Predecessora	Estimativa de duração	Atividade		
A1	-	3 semanas	Início mais cedo		Término mais cedo
A2	A1	3 semanas			
A3	A1	6 semanas	Início mais tarde		Término mais tarde
A4	A2	8 semanas			
A5	A3, A4	4 semanas			



# Folga e caminho crítico



# Desenvolver cronograma - resultado

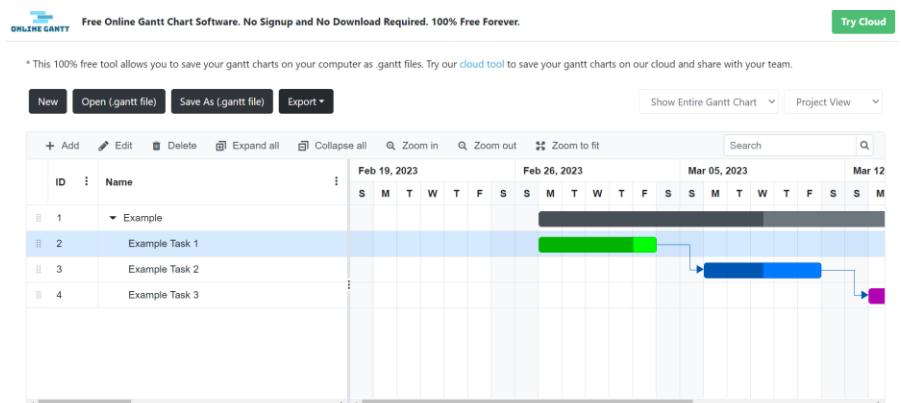


# Atividade: Construir o cronograma.

## Incluir:

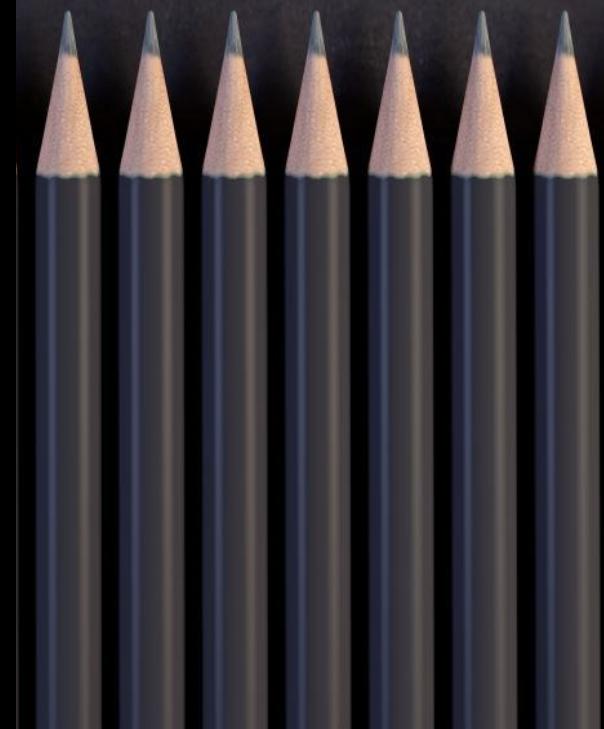
- Registrar a estimativa de esforço para cada atividade com base na estimativa de 3 pontos.
- Registrar a alocação de recursos humanos planejados para cada atividade.
- Registrar a estimativa de duração, com datas e início e fim, considerando a real disponibilidade dos recursos.
- Construir o gráfico de Gantt.

<https://www.onlinegantt.com/#/gantt>



# Aula 9

PMBOK – Planajemento de riscos



# Gestão de riscos



# Como está a adoção da gestão de riscos?

Uma pesquisa feita pelo PMI mostrou como que organizações praticam a gestão de riscos:

- 35% realizavam formalmente.
- 54% realizam informalmente.
- 11% não tratam os riscos em seus projetos.

# O que é um risco?

“Evento ou condição incerta que, se ocorrer, terá um efeito positivo ou negativo sobre pelo menos um objetivo do projeto”

(PMI - Project Management Institute)



# O que é um risco?

“Risco é a possibilidade de sofrer perdas”

(SEI - Software Engineering Institute)



**Software Engineering Institute**  
**Carnegie Mellon**

# O que é o gerenciamento de riscos?

É o processo (procedimento de trabalho) que busca reduzir o número de incertezas que podem se materializar em problemas e minimizar o efeito daquelas que venham a ocorrer.

# Gerenciamento de riscos no CMMI -DEV

Capability Maturity Model Integration (CMMI)

Categorias\Níveis	Nível 2	Nível 3	Nível 4	Nível 5
<b>Engenharia</b>	- Gerenciamento de requisitos.	- Desenvolvimento de requisitos. - Solução técnica. - Integração do produto. - Verificação. - Validação.		
<b>Ger. Projetos</b>	- Planejamento do projeto. - Controle e monitoramento do projeto. - Contratação e gestão de fornecedores.	- Gerenciamento integrado do projeto.  - Gerenciamento de riscos.	- Gerenciamento quantitativo do projeto.	
<b>Ger. Processos</b>		- Foco no processo organizacional. - Definição do processo organizacional. - Treinamento organizacional.	- Desempenho do processo organizacional.	- Inovação e implantação organizacional.
<b>Suporte</b>	- Medição e análise. - Garantia de qualidade de processo e produto. - Gerenciamento de configuração.	- Análise de decisão e resolução.		- Análise e resolução de causas.

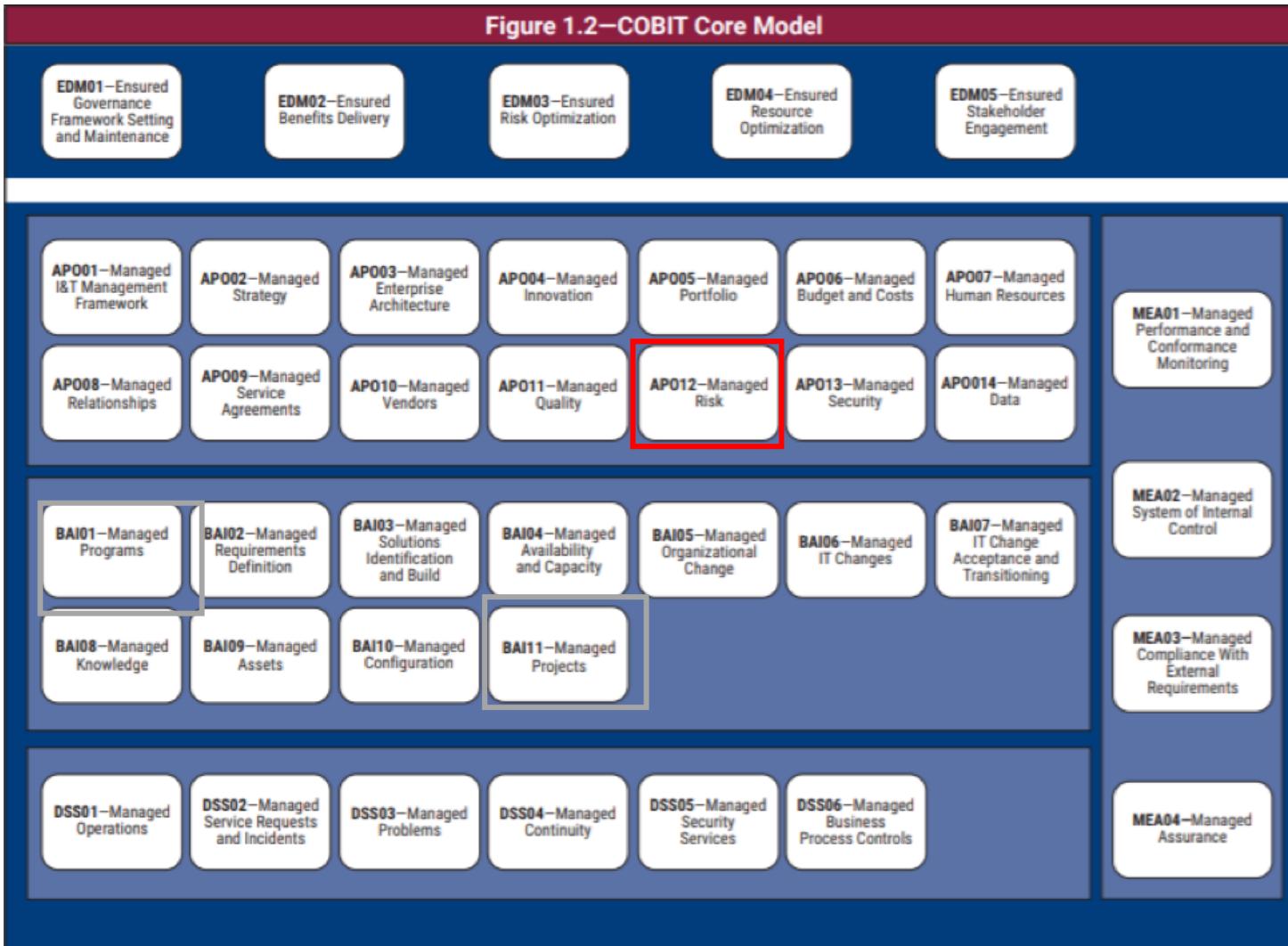
# Gerenciamento de riscos no MPS.BR



\*Compatível com CMMI e com modelo de avaliação ISO/IEC 15504

Nível	Processo	Capacidade
A (mais alto)	Inovação e Implantação na Organização	AP 1.1, AP 2.1, AP 2.2, AP 3.1 e AP 3.2
	Análise e Resolução de Causas	
B	Desempenho do Processo Organizacional	AP 1.1, AP 2.1, AP 2.2, AP 3.1 e AP 3.2
	Gerência Quantitativa do Projeto	
C	Análise de Decisão e Resolução	AP 1.1, AP 2.1, AP 2.2, AP 3.1 e AP 3.2
	Gerência de Riscos	
D	Desenvolvimento de Requisitos	AP 1.1, AP 2.1, AP 2.2, AP 3.1 e AP 3.2
	Solução Técnica	
	Integração do Produto	
	Instalação do Produto	
	Liberação do Produto	
	Verificação	
	Validação	
E	Treinamento	AP 1.1, AP 2.1, AP 2.2, AP 3.1 e AP 3.2
	Avaliação e Melhoria do Processo Organizacional	
	Definição do Processo Organizacional	
	Adaptação do Processo para Gerência de Projeto	
F	Medição	AP 1.1, AP 2.1 e AP 2.2
	Gerência de Configuração	
	Aquisição	
	Garantia da Qualidade	
G	Gerência de Requisitos	AP 1.1 e AP 2.1
	Gerência de Projeto	

# Gerenciamento de Riscos no COBIT



# Por que gerenciar riscos?

Gerenciar riscos é algo que pode ser feito em diferentes contextos:

## **Empresarial**

- Possibilidade de desvalorizar ações de mercado ou custos de matéria prima.
- Situações que causem impacto negativo na imagem da organização

## **Projetos**

- Possibilidade recursos humanos chave na equipe se desligarem do projeto.
- Possibilidade de solicitações de mudança de grande impacto.

## **Pessoal**

- Doença, acidente, perda de emprego, etc.

# Por que gerenciar riscos?

## Cenário A:

Um stakeholder que não fica confortável em correr riscos e buscará a alocação de custos em atividade de gerenciamento de riscos.

## Cenário B:

Um stakeholder não se importa em correr (ou mesmo conhecer) os riscos, e deve estar disposto a arcar com os custos adicionais na ocorrência de eventos que impactem negativamente no projeto.

Vamos conhecer alguns tipos de riscos....

- Conhecidos vs desconhecidos
- Técnicos
- Negócio

# Por que gerenciar riscos?

## Riscos conhecidos:

- Podem ser analisados e ter seu impacto identificado
- Estratégias de resposta podem ser planejadas
- No momento de sua materialização, a equipe saberá como agir.

## Riscos não conhecidos:

- Não se conhece seus gatilhos e nem seus impactos
- O „custo” para tratar o impacto do risco pode ser bastante elevado e a eficácia da resposta pode ser prejudicada.



Dica: Sempre estão presentes

# Riscos técnicos

Os riscos técnicos ocorrem, por exemplo, quando um problema é identificado de forma mais complexa do que se havia planejado.

- Ameaçam o pontualidade do projeto.
- Ameaçam a qualidade do software.
- Podem inviabilizar o produto ou em adequação de escopo quando possível.

# Exemplos: Riscos técnicos

- Utilização das versões mais recentes de determinada tecnologia (incompatibilidade dependências que se planejava utilização).
- Utilização de versões mais antigas de determinada tecnologia (incompatibilidade de conexão com tecnologias mais recentes (ex. Drivers de conexão) e vulnerabilidades de segurança.
- Impossibilidade de extração de dados de uma fonte externa devido a implantação de mecanismos de segurança (ex. protocolos de segurança).
- Identificação de ausência de alguma premissa, como ausência de módulo que considerava-se pronto para uso.

# Riscos do negócio

Os riscos do negócio ameaçam a viabilidade do projeto:

- Construir um excelente produto que ninguém realmente quer.
- Perder o apoio da alta administração devido à mudança de foco ou mudança de pessoas na direção.
- Perder o compromisso orçamentário.

# Exemplo: Riscos de negócio

Você trabalha em uma fábrica de café e a diretoria o incumbiu de lançar um novo produto nas prateleiras.

Chuvas em excesso podem prejudicar a colheita e afetar a produção, talvez insumos importados sejam encarecidos com a alta do dólar e diminuam a margem de lucro estimada.

# Exemplo: Riscos de negócio

Você trabalha em uma fábrica de café e a diretoria o incumbiu de lançar um novo produto nas prateleiras.

Há a possibilidade de que os testes com os grãos vão além da data prevista a fim de atender a qualidade estabelecida e um lançamento semelhante pela concorrência é capaz de comprometer suas vendas.

# Exemplos de riscos

- a) Requisitos pouco claros (superficiais).
- b) Baixo engajamento dos stakeholders (internos ou extenos).
- c) Integração com produtos externos com interfaces de conexão não definidas (ou não conhecidas).
- d) Utilização de tecnologias não conhecidas pela equipe.
- e) Novos integrantes na equipe.
- f) Mudanças de prioridades na empresa.
- g) Estimativas não factíveis.

# Impacto do riscos em projetos

Os riscos de projeto impactam o plano do projeto em diversos aspectos:

- Custo
- Tempo
- Pessoal
- Escopo (requisitos)

Gestão de riscos precisa estar  
fortemente integrada com a gestão das  
demais áreas de conhecimento.

# O que é gerenciamento de riscos?

## **Definição 1:**

É uma forma de conhecer, administrar e se preparar para os elementos incertos que podem impactar no planejamento do projeto.

# O que é gerenciamento de riscos?

## **Definição 2:**

O gerenciamento de riscos é um conjunto de ações tomadas a partir da identificação de possíveis impactos (positivos ou negativos) no projeto.

- Identifica as influências em potencial sobre os objetivos traçados.
- Estas influências são quantificadas e respostas planejadas.
- Orientações são repassadas à equipe quanto as possíveis incertezas (ações mitigatórias).

O Gerenciamento de Riscos segundo o  
PMBOK...

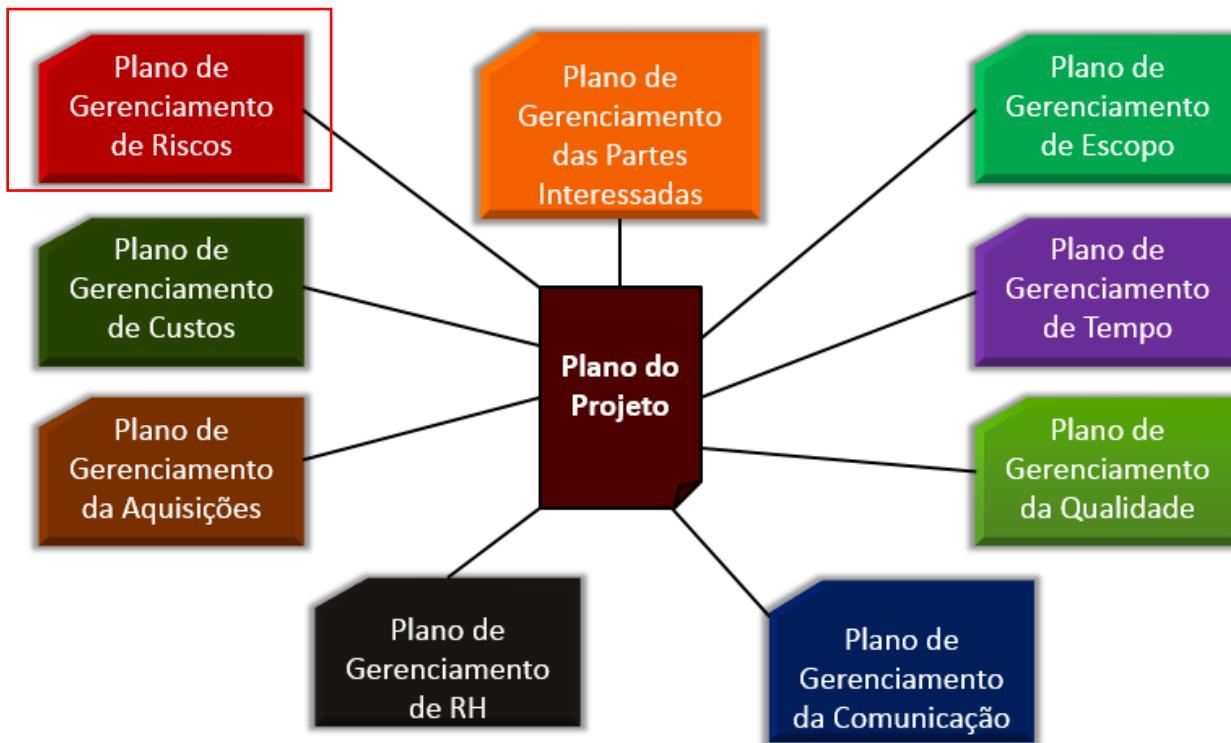
# O Gerenciamento de Riscos pelo PMBOK

Áreas de conhecimento	Grupos de processos de gerenciamento de projetos				
	Grupo de processos de iniciação	Grupo de processos de planejamento	Grupo de processos de execução	Grupo de processos de monitoramento e controle	Grupo de processos de encerramento
<b>11. Gerenciamento dos riscos do projeto</b>		11.1 Planejar o gerenciamento dos riscos 11.2 Identificar os riscos 11.3 Realizar a análise qualitativa dos riscos 11.4 Realizar a análise quantitativa dos riscos 11.5 Planejar as respostas aos riscos		11.6 Controlar os riscos	

- 6 processos
- 1 área de conhecimento
- 2 grupos de processos

# Plano de gerenciamento de riscos

É um dos planos auxiliares do plano de gerenciamento do projeto.



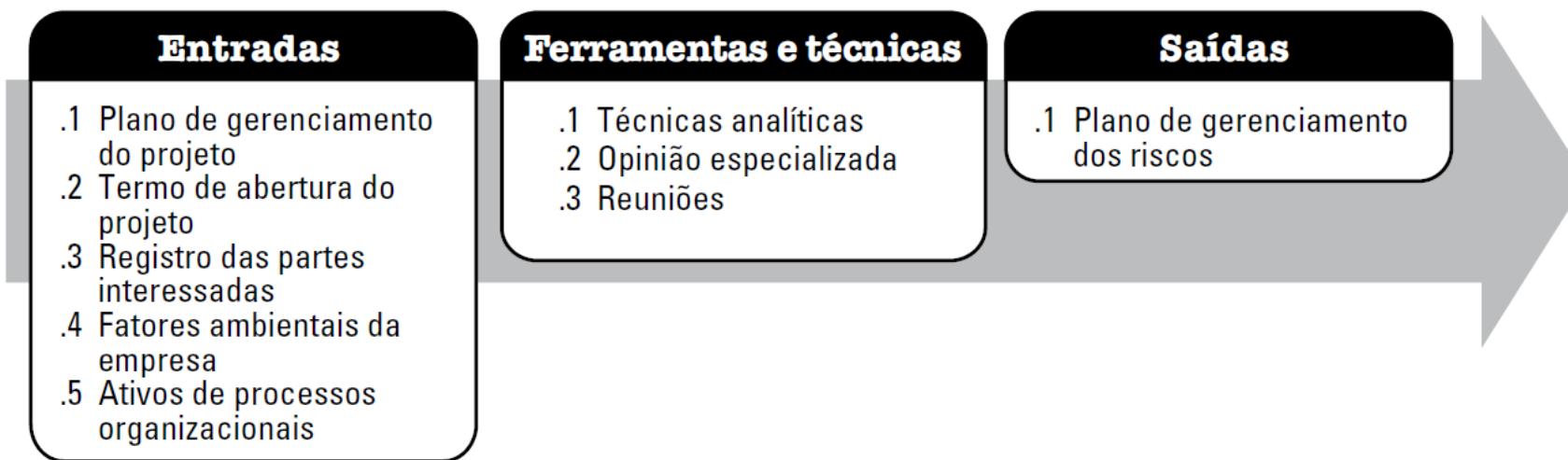
Fonte: <https://doprojetoaoportfolio.wordpress.com/2015/09/15/o-plano-de-gerenciamento-do-projeto/>

# Visão geral

Planejar o gerenciamento de riscos	Definir como conduzir as atividades de gerenciamento de riscos para o projeto
Identificar os riscos	Determinar quais riscos podem afetar o projeto e documentar suas características
Realizar a análise qualitativa de riscos	Avaliar a exposição ao risco para priorizar os riscos que serão objeto de análise ou ação adicional
Realizar a análise quantitativa de riscos	Efetuar a análise numérica do efeito dos riscos identificados nos objetivos gerais do projeto
Planejar as respostas aos riscos	Desenvolver opções e ações para aumentar as oportunidades e reduzir as ameaças do projeto
Monitorar os riscos	Monitorar os riscos durante o ciclo de vida do projeto

Fonte: <https://escritoriodeprojetos.com.br/gerenciamento-dos-riscos-do-projeto>

# Planejamento de gerenciamento de riscos



Como executaremos as atividades de gerenciamento de riscos?

# Fatores ambientais da empresa

Envolvem o ambiente onde o projeto será executado, podendo ser internos ou externos.

Exemplos:

- Estrutura e cultura organizacional
- Normas governamentais e de setor
- Infraestrutura existente
- Recursos humanos
- Administração de pessoal
- Autorização de trabalho
- Condições do mercado
- Tolerância a risco das partes
- Sistema de informações de gerenciamento de projetos

# Ativos de processos organizacionais

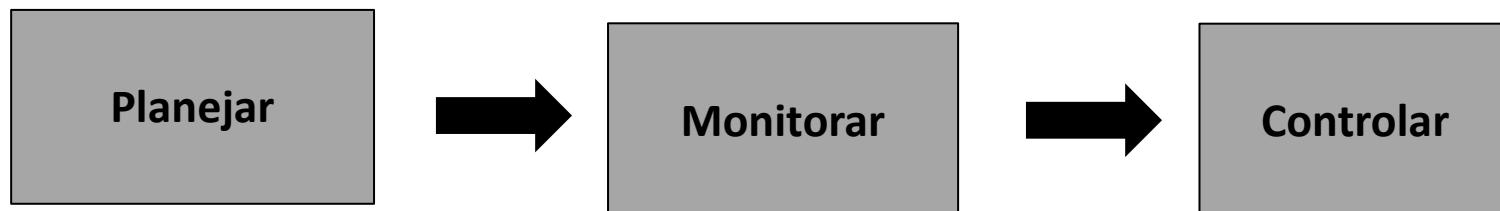
Os ativos de processos organizacionais são os ativos relacionados aos processos da empresa que impactam no projeto.

- Políticas, diretrizes e procedimentos.
- Procedimentos de qualidade: listas de verificação, instruções de trabalho.
- Requisitos de comunicação, controles financeiros.
- Base de conhecimento dos projetos passados: Lições aprendidas, informações históricas.

# Planejamento de gerenciamento de riscos

O planejamento do gerenciamento dos riscos é a primeira etapa da gerência de riscos.

- Como identificar os riscos (checklist, reuniões, analogias, etc.)?
- Quem serão as pessoas envolvidas?
- Como os riscos serão classificados (EAR –Estrutura Analítica dos Riscos)?
- Com qual frequência os riscos serão monitorados?
- Quanto (\$) será alocado para o tratamento dos riscos?

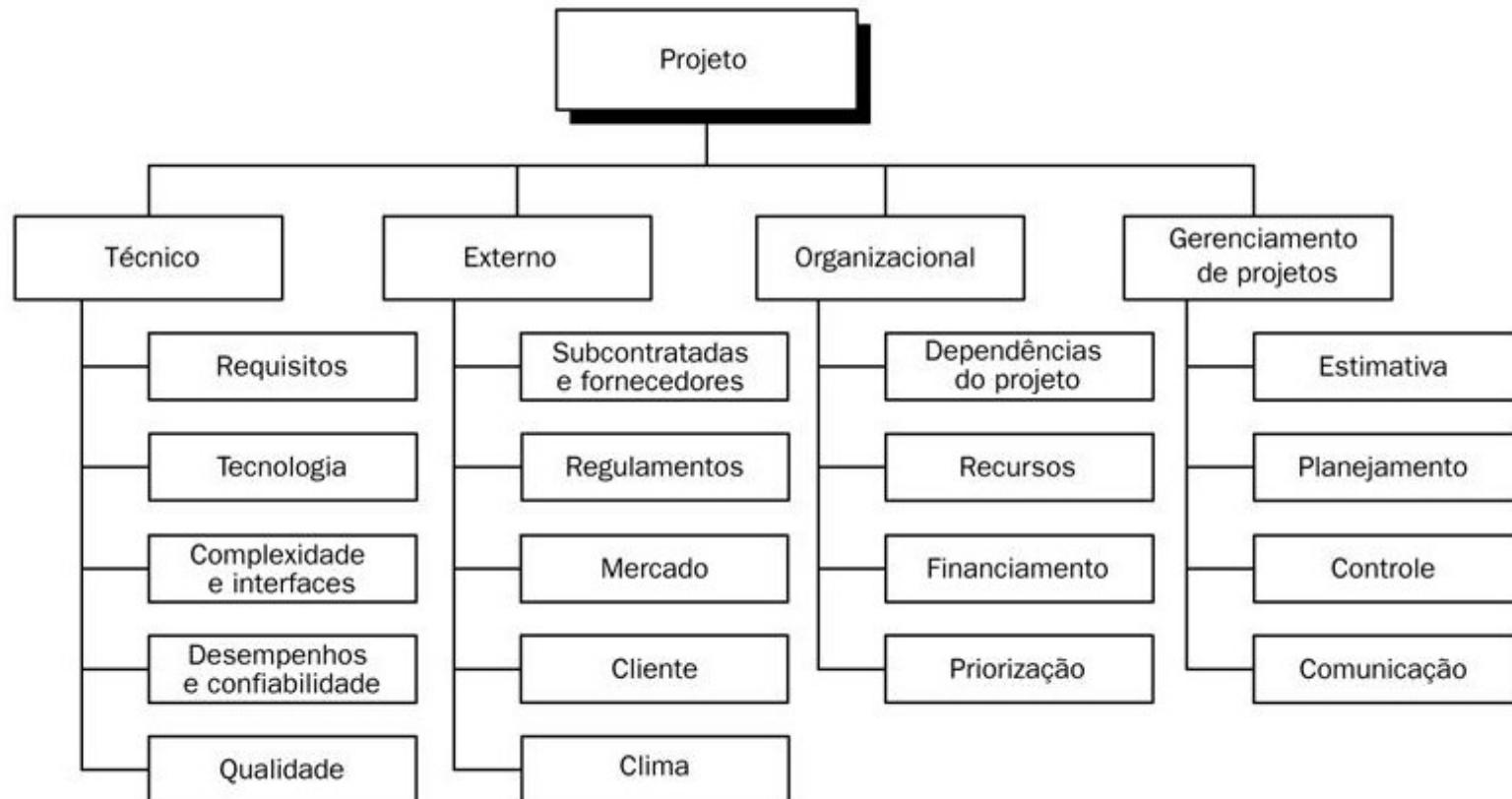


# Plano de gerenciamento de riscos

Informações que devem constar no plano de gerenciamento de riscos?

- Com qual frequência os riscos serão revisados?
- Quais são os níveis de probabilidade?
- Quais são os níveis de impacto?
- Define a matriz de probabilidade e impacto.
- Quais são os tipos de riscos (EAR)?
- Quanto será alocado para reserva de contingência (riscos identificados)?
- Quanto será alocado para reserva gerencial (riscos não identificados).

# EAR - Exemplo



Fonte:<https://wpm.wdfiles.com/local--files/artefato%3Aplano-de-gerenciamento-de-riscos/Figura11.4.JPG>

# Níveis de Probabilidade - Exemplo

Qualitativo	Quantitativo
<b>Probabilidade</b>	<b>% de certeza de materialização</b>
1-Muito baixa	0 a 20%
2-Baixa	20 a 40%
3-Média	40 a 60%
4-Alta	60 a 80%
5-Muito Alta	> 80%

# Níveis de Impacto - Exemplo

A análise do impacto depende da área impactada:

## Impacto

1-Muito baixo

2-Baixo

3-Médio

4-Alto

5-Muito Alto



Fonte:<https://movimentoimpactoglobal.com.br/tripla-restricao/>

# Níveis de Impacto - Exemplo

## Impacto

1-Muito baixo

2-Baixo

3-Médio

4-Alto

5-Muito Alto

	Muito baixo (Nota = 1)	Baixo (Nota = 2)	Médio (Nota = 3)	Alto (Nota = 4)	Muito alto (Nota = 5)
<b>Custo</b>	Até 2% no orçamento	De 2 a 5% no orçamento	De 5 a 8% no orçamento	De 8 a 10% no orçamento	Acima de 10% no orçamento
<b>Tempo</b>	Até 2% no prazo total	De 2 a 5% no prazo	De 5 a 8% no prazo	De 8 a 10% no prazo	Acima de 10% no prazo
<b>Escopo</b>		Mudança impactará no custo	Mudança impactará no custo e no tempo	Mudança impactará no custo, tempo e qualidade	

# Matriz de probabilidade x impacto - Exemplo

Porque precisamos definir a **importância (fator de exposição)** dos riscos?

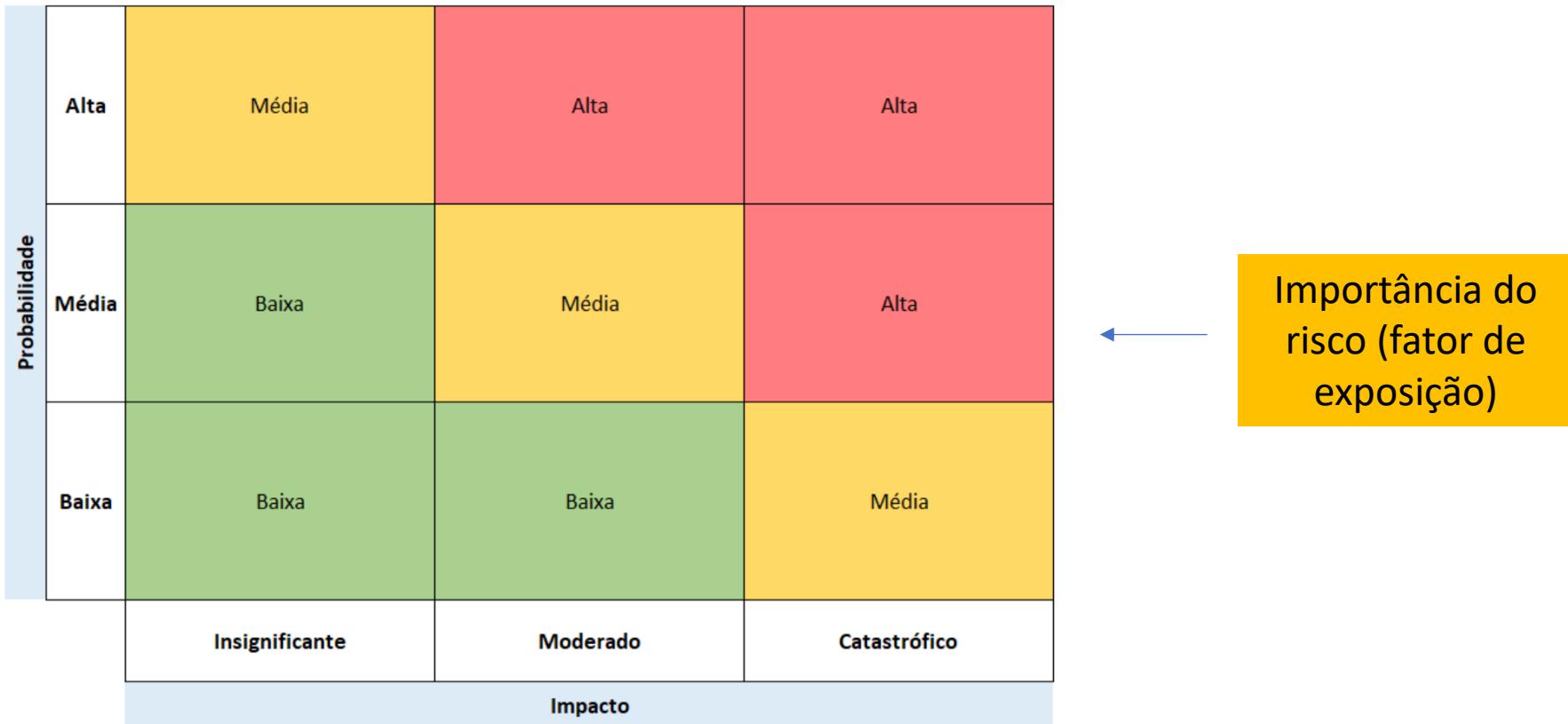
O gerenciamento de um determinado risco é caro (\$\$)!

- Planejam-se ações de mitigação
- Planejam-se ações de contingência
- Despende-se tempo no seu monitoramento e controle.



Buscando maior eficiência é necessário priorizar os riscos que serão monitorados.

# Matriz de probabilidade x impacto - Exemplo



Fonte:<https://ferramentasdaqualidade.org/matriz-de-riscos-matriz-de-probabilidade-e-impacto/>

# Matriz de probabilidade x impacto - Exemplo

<b>Probabilidade</b>	90%	Média	Média	Alta	Alta	Alta
	70%	Baixa	Média	Média	Alta	Alta
	50%	Baixa	Baixa	Média	Alta	Alta
	30%	Baixa	Baixa	Média	Média	Alta
	10%	Baixa	Baixa	Baixa	Baixa	Média
		<b>Muito Baixo</b>	<b>Baixo</b>	<b>Moderado</b>	<b>Alto</b>	<b>Muito Alto</b>
		<b>Impacto</b>				

Fonte:<https://ferramentasdaqualidade.org/matriz-de-riscos-matriz-de-probabilidade-e-impacto/>

# Matriz de probabilidade x impacto - Exemplo

Não podemos esquecer os riscos positivos... (oportunidades)

Probabilidade	Ameaças					Oportunidades					
	90%	Média	Média	Alta	Alta	Alta	Baixa	Baixa	Baixa	Média	Média
70%	Baixa	Média	Média	Média	Alta	Alta	Baixa	Baixa	Média	Média	Alta
50%	Baixa	Baixa	Baixa	Média	Alta	Alta	Baixa	Baixa	Média	Alta	Alta
30%	Baixa	Baixa	Baixa	Média	Média	Alta	Baixa	Média	Média	Alta	Alta
10%	Baixa	Baixa	Baixa	Baixa	Baixa	Média	Média	Alta	Alta	Alta	Alta
Impacto											
	Muito Baixo	Baixo	Moderado	Alto	Muito Alto	Muito Alto	Alto	Moderado	Baixo	Muito Baixo	

O que fazer quando uma determinada oportunidade se apresentar?

# Questão: gerenciamento de riscos

- Identifique abaixo uma ferramenta de gerenciamento de riscos que permite de forma visual detectar os riscos que devem receber mais atenção. Por se tratar de uma ferramenta para priorização de riscos, ela pode ser aplicada na etapa de avaliação de riscos

A Matriz de qualidade e tempo.

B Matriz de gerenciamento do tempo.

 C Matriz de probabilidade e impacto.

D Matriz de tempo e impacto.

E Matriz de priorização do tempo.

# Questão: Gerenciamento de riscos

- Segundo o PMBOK, auxiliar no planejamento estratégico, melhorar a comunicação entre os stakeholders, reduzir o estresse dos stakeholders e diminuir a ocorrência de surpresas desagradáveis, são alguns dos benefícios do gerenciamento de:

A iniciação.

B tempo.

C qualidade.

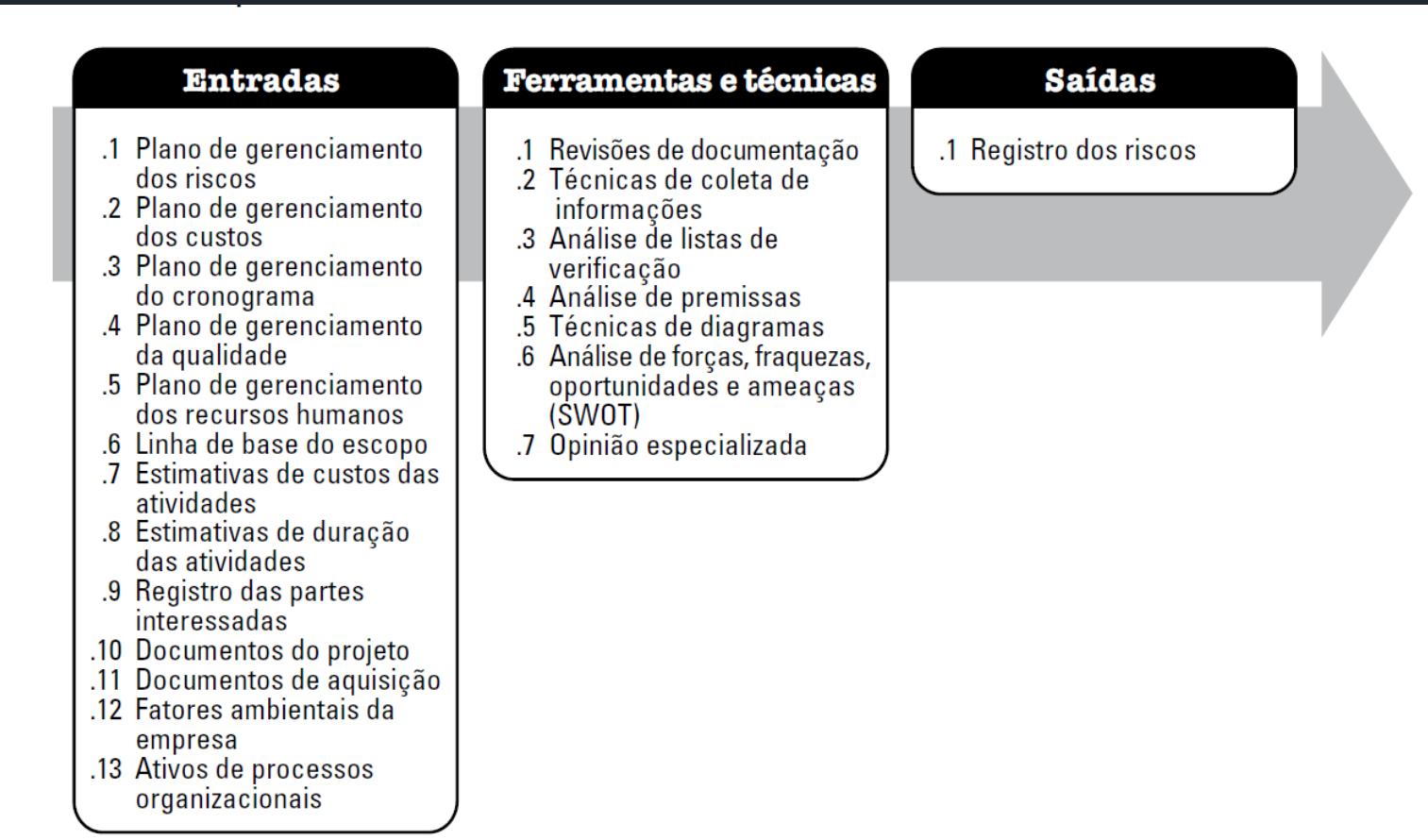
D integração.

risco.

# Atividade: Elaborar o planejamento do gerenciamento de riscos.

- Com qual frequência os riscos serão revisados?
- Quais são os níveis de probabilidade?
- Quais são os níveis de impacto?
- Define a matriz de probabilidade e impacto.
- Quais são os tipos de riscos (EAR)?
- Quanto será alocado para reserva de contingência (riscos identificados)?
- Quanto será alocado para reserva gerencial (riscos não identificados).

# Identificação dos riscos



# Identificação dos riscos

- A identificação dos riscos tem como objetivo determinar quais eventos (incertos) podem afetar o projeto e documentar suas características.
- É um processo iterativo que deve ocorrer durante todo o projeto.

# Identificação dos riscos

**Lembre-se que não é pelo fato do risco não estar sendo visto que ele não existe!**

- É melhor conhecer as consequências dos riscos para saber como agir diante de sua materialização.
- Ao conhecer as possibilidades do que pode acontecer em um projeto, e com um programa de gerenciamento de riscos, um gerente de projeto poderá adotar medidas que continuem garantindo o sucesso do trabalho mesmo sob condições desfavoráveis.

# Identificação dos riscos

- Nunca é possível conhecer todos os riscos qual o projeto está sujeito.
- Por mais que o processo de identificação de riscos seja bem executado, sempre podem ocorrer situações que não foram previamente mapeadas.

# Identificação dos riscos

## **Quais são as principais fontes de risco?**

As fontes de risco estão ligadas a diversos fatores externos ao projeto e a organização e também internos a estes.

# Identificação dos riscos

## Fontes externas dos riscos

### **Fatores Imprevisíveis.**

- Requisitos regulamentares.
- Desastres naturais.
- Fatores ambientais (Pandemia, greve de serviços, etc.)

### **Fatores Previsíveis**

- Riscos financeiros.
- Evolução da tecnologia
- Riscos legais (adequação LGPD?), etc.

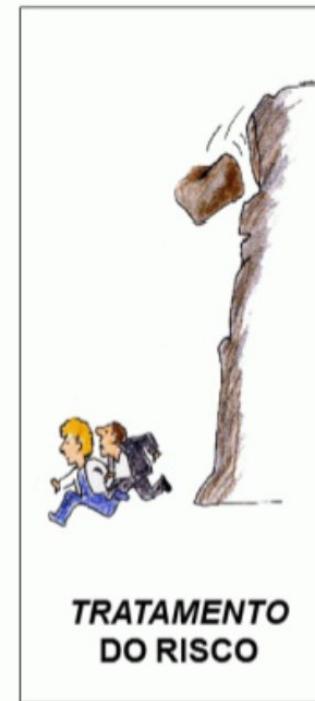
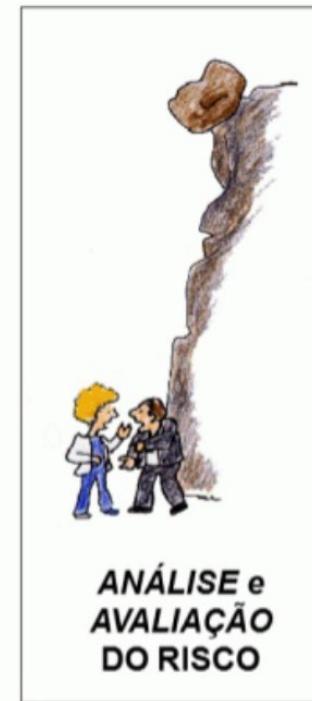
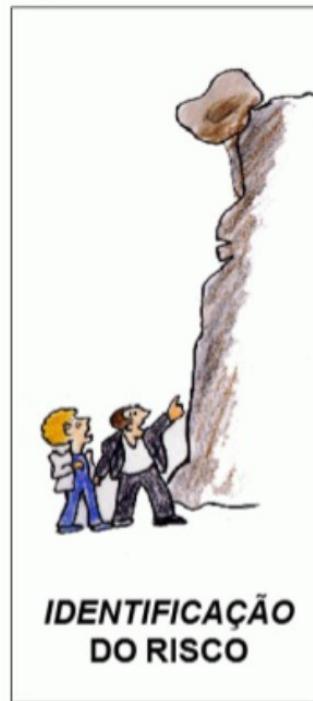
# Identificação dos riscos

## **Fontes internas de riscos**

- Excesso de projetos sendo desenvolvidos ao mesmo tempo.
- Um cronograma não realista (estimativas baseadas apenas no melhor caso)
- Solicitações de mudança descasadas da análise de impacto e construção de novas baselines.

# Identificação dos riscos

Como identificar os riscos?



# Riscos negativos vs. riscos positivos

## Riscos negativos

- Conhecidos como **ameaças**.
- Quando não gerenciadas podem acarretar em atrasos, estouro de orçamento, escopo incompleto.
- Deve-se programar ações para reduzir esses impactos.

## Riscos positivos

- Conhecidos como **oportunidades**.
- Podem impactar na redução de tempo e de custo.
- Pode-se planejar ações para maximizá-las.

# Riscos negativos - Exemplos

Exemplos ilustrados no guia PMBOK:

- Saída de um colaborador-chave da organização;
- Produtividade abaixo do planejado;
- Número de erros encontrados durante testes maior ou menor do que o esperado;
- Condições meteorológicas atípica

# Riscos negativos - Exemplos

- Obras de infraestrutura normalmente requerem licença ambiental, que podem ser negadas ou demorar além do esperado para serem concedidas.
- Obras de edificações podem ter sua conclusão postergada devido a  períodos de chuvas contínuas, devendo ser considerada a localização geográfica da edificação na elaboração do cronograma.

O mapeamento dos riscos orienta o gerente no cumprimento do escopo, custos, e tempo.

# Riscos Positivos

Muitos gerentes veem os riscos como elementos negativos, mas eles também podem ser positivos.

Exemplo:

O alto conhecimento técnico e a experiência da sua equipe atuam a favor do projeto, possibilitando rápido desenvolvimento e reutilização de muitos componentes.

Em um projeto cobrado por pontos de função a equipe terá grande velocidade e o. que maximizará os lucros.

# Riscos positivos – Exemplo

- Se o cliente desistir de integrar os dados com o Sistema XPTO, o Projeto poderá ser encerrado com 1 mês de antecedência.
- Se o cliente optar por um WebAPP ao invés de APPs nativos, o projeto pode necessitar de dois programadores a menos, reduzindo os custos e o tempo para conclusão.
- Se a lei 123/2020 for revogada, o sistema não precisará do modulo de auditoria, reduzindo a complexidade do produto, e assim o custo e tempo para conclusão do projeto.

# Quais técnicas podem ser utilizadas para identificar riscos?

- Uso de checklists (com perguntas ou listas de riscos já pré-definida pela organização)
- Reuniões e brainstormings com gerentes e equipes experientes na organização.
- Análise de cenários.
- Lições aprendidas em projetos anteriores.

# Exemplo checklist – Para riscos de pessoal

- Há pessoas suficientes para conduzir as atividades do projeto?
- As pessoas têm as competências adequadas para as atividades?
- A alocação dos membros da equipe está confirmada para o período que precisam atuar no projeto?
- Há membros da equipe alocados em mais de um projeto simultaneamente?

# Exemplo checklist – Para riscos de cliente

- Você já realizou outros projetos com o cliente?
- O cliente tem ideias sólidas dos requisitos?
- O cliente tem expectativas de tempo realísticas para o escopo solicitado?

# Identificação dos Riscos

A Matriz SWOT (Strengths, Weaknesses, Opportunities e Threats )

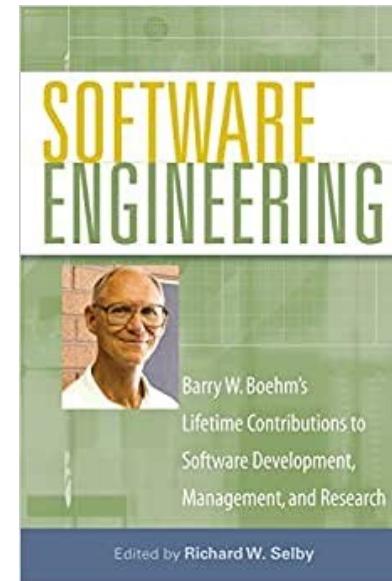
A Matriz SWOT é muito conhecida por ser utilizada em planos de negócio, análises de mercado e desenvolvimento de estratégias.



Fonte: <https://www.agendor.com.br/blog/matriz-swot-como-fazer/>

# Top 10 (Boehm)

1. Imprevistos de pessoal.
2. Cronogramas e orçamentos não realísticos.
3. Desenvolvimento das funções erradas.
4. Desenvolvimento da interface com o usuário errada.
5. Requisitos sofisticados, sem necessidade.
6. Fluxo contínuo de mudanças nos requisitos.
7. Imprevistos em serviços terceirizados.
8. Imprevistos em componentes terceirizados.
9. Imprevistos de desempenho em tempo real.
10. Capacidade de computação excedida.



Barry William Boehm  
Software Risk Management, 1989

<https://dl.acm.org/doi/book/10.5555/107446>

Como registrar os riscos  
identificados?

# Registro dos riscos

- Descrição do risco
- Causa (Por que o risco pode ver a ocorrer)
- Efeito/Consequência (Vai impactar onde? RH, escopo, tempo, custo, etc.)
- Gatilhos (como identificar que ocorreu?)
- Probabilidade
- Impacto
- Estratégia de resposta
- Responsável (owner)
- Ações mitigatórias
- Ações de contingência
- etc.

# Causa x Efeito

Uma boa prática é descrever o risco em relação a suas causas e efeitos (consequências).

Se <causa> então <efeito>

Risco: Fonte de dados inacessível no sistema XPTO.

**Se** a fonte de dados do XPTO se tornar inacessível **então** não será possível concluir o módulo integrador.

# Causa x Efeito

Não confundir efeito com possibilidade de resposta!

**Se** a fonte de dados do XPTO se tornar inacessível **então**  
não será possível concluir o módulo integrador por  
simples extração de dados.

**Se** a fonte de dados do XPTO se tornar inacessível **então**  
será necessário negociar com a empresa detentora dos  
dados outra forma alternativa de intercâmbio dos  
dados.

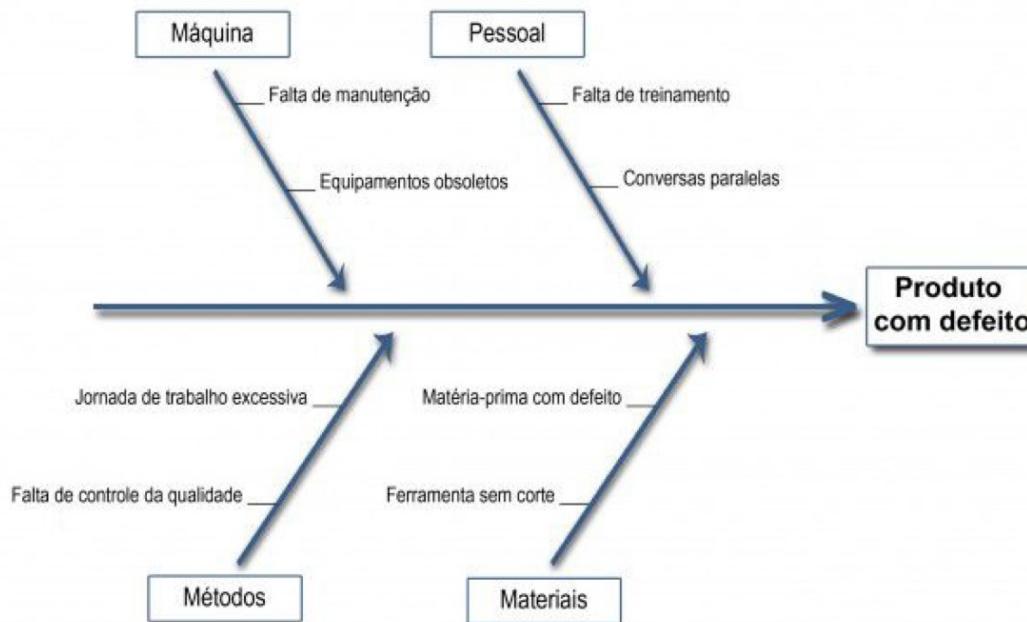
# Exemplo risco - descrição

“Se for necessário adotar um novo hardware para embarcar o software então erros inesperados de integração do sistema podem ocorrer, o que levaria a estouros dos custos do projeto”

# Causa x Efeito

## Diagrama de Espinha de Peixe (Diagrama de Ishikawa)

- Realiza uma análise de causa e efeito para um problema, a fim de identificar sua causa raiz.
- O diagrama quebra “o todo” em “partes”. Por isso, diz-se que a técnica direciona a equipe a olhar para as categorias e pensar em causas alternativas.



Fonte: <https://www.revistaespacios.com/a18v39n03/18390309.html>

# Registro dos riscos

A identificação de riscos é um processo **iterativo**:

- É comum atualizar o registro dos riscos conforme outros processos de gerenciamento dos riscos são conduzidos.
  - Seja com as informações complementares.
  - Seja com a inclusão de novos riscos.
- A identificação de novos riscos pode ocorrer durante a realização de outros processos da etapa de planejamento do projeto.
  - Exemplo: Está planejando contratar um módulo que atenderá parte das funcionalidades pretendidas pelo produto. Porém existe incerteza se o fornecedor irá entregar o módulo no prazo esperado. (Plano de aquisições)

# Atividade: Realizar a identificação dos riscos do projeto.

Informações a serem documentadas para cada risco:

- Descrição do risco
- Causa
- Efeito/Consequência
- Gatilhos

Mínimo 8 riscos!

# Análise qualitativa de riscos

Consegues elencar os riscos do projeto em ordem de prioridade?

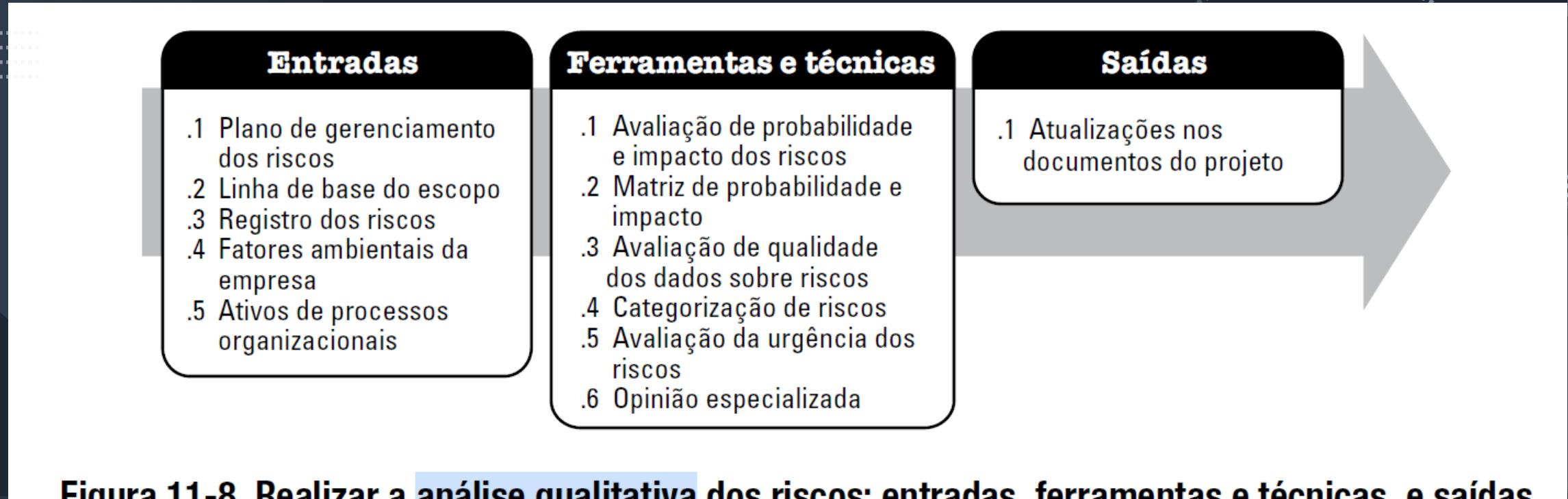


Figura 11-8. Realizar a análise qualitativa dos riscos: entradas, ferramentas e técnicas, e saídas

# Análise qualitativa de riscos

## Objetivo

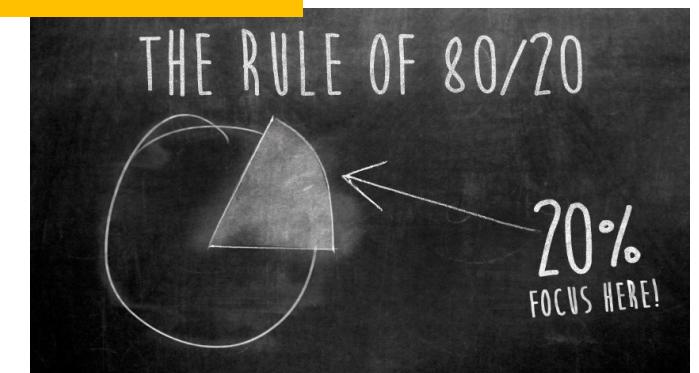
- Avaliar a exposição ao risco (importância) para priorizar os riscos que serão objeto de análise ou ação adicional.
- Os riscos com **maior probabilidade e impacto** são priorizados para posterior criação de um plano de respostas.
- Os riscos com **menor probabilidade e impacto** são mantidos nos registros dos riscos dentro de uma lista de observação (*watchlist*) para monitoramento futuro.

# Análise qualitativa de riscos

Por que realizar a análise qualitativa?

- O controle dos riscos gera custo ao projeto.
- Grandes projetos: 30 a 40 riscos.
  - Se cada risco precisar ser analisado e monitorado, isto terá custos elevados.
  - Por isto focar os esforços nos riscos de fato de maior importância.

Pareto



# Análise qualitativa de riscos

“Podemos não adivinhar o futuro  
mas é útil saber que nível de  
confiança temos nele”.

Nuno Nogueira



# Análise qualitativa de riscos

- O processo de análise qualitativa de riscos se inicia através de reuniões de grupo (opinião especializada).
  - Pode ser utilizado histórico do registro de riscos de projetos anteriores da organização.
- Nas reuniões devem ser estimados o nível de probabilidade e impacto de cada risco.
- Na abordagem qualitativa, os parâmetros de probabilidade e impacto serão expressos em uma escala ordinal definida do plano de gerenciamento de riscos.

# Análise qualitativa de riscos

## **Analise da probabilidade**

- Todo risco possui uma probabilidade de ocorrência que não pode ser zero ( certeza de inexistência ) e nem 100% ( certeza de ocorrência ).
- Estas duas situações a cima não são riscos e sim certezas!

# Análise qualitativa de riscos

## Analise da impacto

- Se o risco se materializar?

Identificar o que seria afetado em relação ao que foi previamente planejado?

- O cronograma iria atrasar?
- Os custos do projeto iriam aumentar?
- Inviabilizaria a completude do escopo do projeto?

Na análise qualitativa pode-se trabalhar em faixas, exemplo:

- pouco, médio, muito.
- Até 5 dias; até 15 dias, mais de 15 dias.
- Etc.

A quantificação ou qualificação dos níveis de impacto que é diferenciam a análise qualitativa da análise quantitativa

# Análise qualitativa de riscos

Quais riscos devem ser priorizados para o desenvolvimento do plano de resposta?

- Como base na análise de probabilidade e impacto podemos definir o nível de importância dos riscos.
- Utilizaremos como apoio a matriz de probabilidade e impacto.

# Matriz de probabilidade x impacto - Exemplo

	Alta	Média	Alta	Alta
Probabilidade	Média	Baixa	Média	Alta
Baixa	Baixa	Baixa		Média
	Insignificante	Moderado	Catastrófico	
Impacto				

Fonte:<https://ferramentasdaqualidade.org/matriz-de-riscos-matriz-de-probabilidade-e-impacto/>

# Análise qualitativa de riscos

Exemplo de análise qualitativa de riscos com base em matriz de probabilidade e impacto.



Aqui os riscos 11, 6, 12, 4, 1 e 4 são os de maior importância, evidenciando aqueles que precisam ser trabalhados.

Atividade: Realizar a análise qualitativa para todos os riscos identificados para o projeto.

- Qual o nível de probabilidade?
- Qual o nível de impacto?
- Qual o fator de exposição?
- Verifique se é necessário atualizar o plano de gerenciamento de riscos.

# Análise quantitativa de riscos

Para os riscos priorizados na análise qualitativa, quais serão seus impactos no plano do projeto em relação a tempo, custo, e escopo? (como ficaria a nova baseline após sua materialização?)

## Entradas

- .1 Plano de gerenciamento dos riscos
- .2 Plano de gerenciamento dos custos
- .3 Plano de gerenciamento do cronograma
- .4 Registro dos riscos
- .5 Fatores ambientais da empresa
- .6 Ativos de processos organizacionais

## Ferramentas e técnicas

- .1 Técnicas de coleta e apresentação de dados
- .2 Técnicas de modelagem e **análise quantitativa** dos riscos
- .3 Opinião especializada

## Saídas

- .1 Atualizações nos documentos do projeto



# Análise quantitativa de riscos

## Objetivos:

- Efetuar a análise numérica do **efeito** dos riscos identificados nos objetivos gerais do projeto.
- Por envolver alta complexidade é realizada somente nos riscos priorizados pela análise qualitativa.
- Sua realização apoia muito as atividades de controle de um projeto.

# Análise quantitativa de riscos

Na análise quantitativa é necessário mensurar o quanto a materialização do risco impacta no plano do projeto.

- Exemplo: a chuva atrasa a entrega da obra, mas não atrapalha uma atividade de desenvolver um aplicativo para celular. Um dia chuvoso equivale a quantos dias de atraso na entrega da obra?

# Exemplo análise quantitativa

**Exemplo:**

**Identificação do risco** Somente 70% dos componentes prontos poderão ser reutilizados. A funcionalidade restante deverá ser desenvolvida pela equipe do projeto.

**Probabilidade** 80%

**Impacto** Foram planejados 60 componentes reutilizáveis. Se apenas 70% são reutilizáveis, então 18 terão que ser desenvolvidos desde o início.

Considerando que cada componente tem 100LOC e dados iniciais indicam um custo de \$14/LOC, o custo a mais será:

- $18 \times 100 \times 14 = \$25.200$

Exemplo de Exposição ao risco na análise quantitativa (quanto a custo)=  
 $0,80 \times 25.200 = \$20.160$

Ainda considerar o tempo no cronograma para acomodar as atividades de desenvolvimento destes componentes.

# Análise quantitativa de riscos

A análise qualitativa também pode ser aplicada para os processos de identificar os níveis de probabilidade e impacto dos riscos.

Matriz de Probabilidade e Impacto					
Probabilidade	Ameaças				
	0,90	0,05	0,09	0,18	0,36
	0,70	0,04	0,07	0,14	0,28
	0,50	0,03	0,05	0,10	0,20
	0,30	0,02	0,03	0,06	0,12
	0,10	0,01	0,01	0,02	0,04
		0,05	0,10	0,20	0,40
		Muito Baixo	Baixo	Moderado	Alto
					Muito Alto

Porém estimar a probabilidade de forma quantitativa necessita de muitos dados históricos de projeto anteriores para saber qual a probabilidade de um mesmo risco, já existente em projetos anteriores ocorrer.

- Em quantos projetos este mesmo risco estava presente e em quantos ocorreu?

# Análise quantitativa de riscos

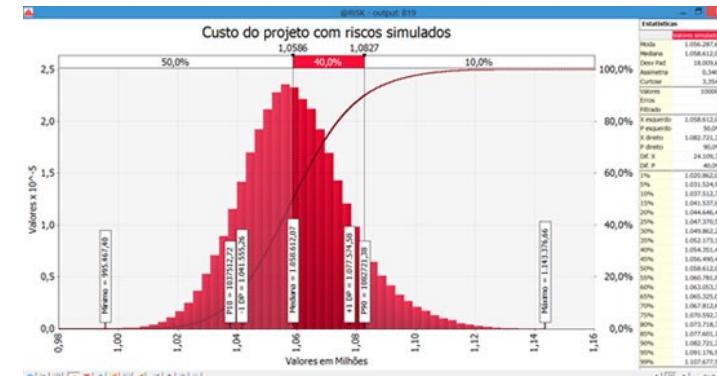
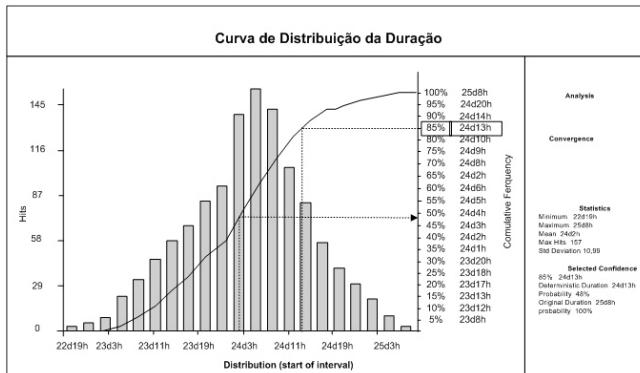
## SIMULAÇÃO DISCRETA:

- Com a análise quantitativa realizada, podem ser utilizadas técnicas de simulação estatística, por exemplo, método de **monte carlo**.
- Esta técnica possibilita simular a ocorrência dos riscos e seus impactos no projeto, podendo assim prever cenários de pior caso.

# Técnicas de Gestão de riscos

A Análise de Monte Carlo é uma técnica estatística que executa simulações de cenários para:

- Visualizar o impacto dos riscos para diferentes cenários (de forma individual e combinada)
- Fornece ao tomador de decisão uma série de resultados possíveis, bem como as probabilidades de ocorrência desses resultados de acordo com a ação a ser tomada.
- O processo de simulação é feito com um software, o qual executa os cálculos, apresentando em gráficos e tabelas, toda a gama de resultados possíveis e a probabilidade de cada um deles.



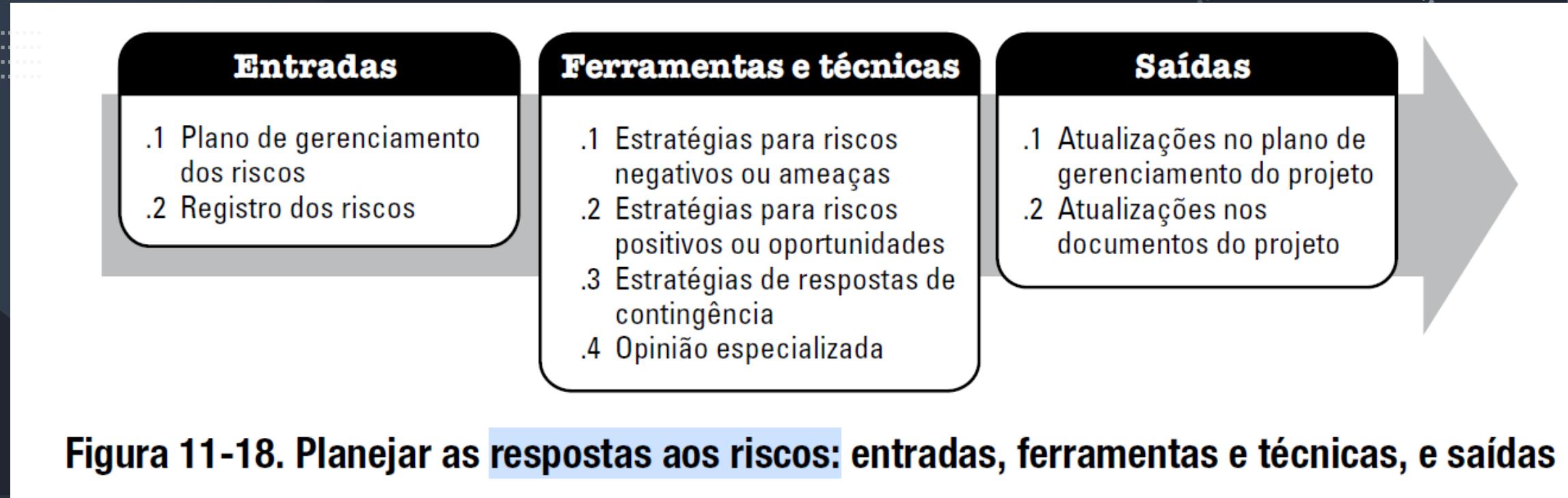
Fonte: <https://pmkb.com.br/artigos/aplicacao-da-simulacao-de-monte-carlo-em-projetos/>  
<https://beware.com.br/wp-content/uploads/2015/03/simulacao-de-monte-carlo.jpg>

# Atividade: Realizar a análise quantitativa.

- Para cada riscos priorizado (alto fator de exposição):
- avaliar nos planos de cronograma, custos, e escopo seus respectivos impactos.

# Planejamento das respostas aos riscos

- Como agir para cada risco? Faço algo antes dele ocorrer? Faço algo após ele ocorrer?



# Planejamento das respostas aos riscos

## **Objetivo**

Estabelecer uma estratégia de respostas e desenvolver um plano de ação buscando aumentar as oportunidades e reduzir os impactos das ameaças aos objetivos do projeto.

# Planejamento de respostas aos riscos

Para cada risco priorizado é necessário estabelecer uma estratégia de resposta.

Os tipos de estratégias existentes, são já pré estabelecidos pelo PMBOK.

Estratégia para riscos negativos	Estratégias para riscos positivos
Eliminar	Explorar
Transferir	Compartilhar
Mitigar	Melhorar
Aceitar	Aceitar

De acordo com a estratégia pode ser necessário planejar ações de resposta, que podem ser ações mitigatórias ou ações de contingência.

# Planejamento das respostas aos riscos

**Eliminar:** identificar e eliminar os fatores que geram os riscos.

**Transferir:** prever ação de terceirizar as ações de contingência do risco. É necessário planejar sua transferência. (ex. seguro)

**Mitigar (prevenir):** implementar e executar um plano como parte da gestão do projeto para identificar e prevenir os riscos antes que eles se tornem problemas.

**Aceitar:** reconhecer e assumir os riscos, sem tomar nenhuma ação sobre eles.

# Planejamento de respostas aos riscos

## Quando aceitar os riscos?

Embora possa parecer prejudicial ao projeto, os riscos normalmente são aceitos quando:

- Representam uma oportunidade.
- A resolução dos riscos é muito cara.
- Não é possível solucionar o risco.

# Planejamento de respostas aos riscos

## Positivos

- **Explorar:** tenta eliminar a incerteza do risco, fazendo que a oportunidade ocorra.
- **Compartilhar:** a organização poderá juntar-se a uma outra parte para potencializar os ganhos.
- **Melhorar:** é contrário de mitigar no risco negativo, é aumentar a probabilidade dos impactos positivos pela maximização dos principais acionadores do risco.
- **Aceitar:** aceitar a oportunidade e colher frutos dela.

# Planejamento de Respostas aos riscos

## Ações de mitigação:

O que fazer para evitar que o risco ocorra ou minimizar seus impactos quando materializados.

# Planejamento de Respostas aos riscos

**Exemplo: Como mitigar risco de rotatividade de pessoal?**

- Reunião com a equipe para identificar causas de rotatividade elevada.
- Condições de trabalho ruins, salários baixos, mercado de trabalho competitivo.
- Após o início do projeto, assumir que a rotatividade acontecerá e desenvolver técnicas para garantir a continuidade do projeto.
- Organizar a equipe de tal maneira que informações sobre atividades de desenvolvimento são amplamente conhecidas.
- Definir padrões para o desenvolvimento.
- Executar revisões em pares de todo o trabalho.

# Planejamento de Respostas aos riscos

**Ações de contingência:**

O que fazer logo após a materialização do risco?

# Planejamento de Respostas aos riscos

## Exemplo de risco com plano de respostas

**Probabilidade:** 80%

**Impacto:** alto

**Descrição** Somente 70% dos componentes de software programados para reutilização serão de fato integrados na aplicação. A funcionalidade restante deverá ser desenvolvida de maneira personalizada

## Mitigação/Monitoração

- Verificar como utilizar adequadamente os componentes já desenvolvidos
- Solicitar a empresa que componentes de diferentes produtos sigam padrão de publicação de interfaces de componentes como serviços. ocorra a padronização de interfaces de comunicação

## Gerenciamento/Contingência

- Desenvolver nova versão do cronograma para o desenvolvimento dos 18 componentes personalizados
- Reservar esse valor no custo de contingência do projeto.

# Planejamento das respostas aos riscos

Plano de riscos e o plano do projeto:

- **Plano de tempo (cronograma):** A execução das ações de prevenção devem constar no cronograma do projeto afim de prever sua execução.
- **Plano de custos:** As ações de prevenção e de contingência devem estar previstas no orçamento do projeto a fim de que haja viabilidade de sua execução.

# Atividade:

## Elaborar o plano de resposta aos riscos.

Incluir para cada risco priorizado:

- Estratégia de resposta
- Ações mitigatórias
- Ações de contingência.

# Atividade: Elaborar Plano do Projeto

- Dando continuidade à atividade do termo de abertura do projeto, agora será elaborado o plano do projeto. O plano deve conter:
  - Planejamento de escopo (declaração, requisitos, EAP, dicionário EAP, etc.)
  - Planejamento de cronograma (atividades, recursos estimados, esforço estimado, duração estimada, sequenciamento, gráfico de Gantt, etc.)
  - Planejamento de riscos (riscos identificados, análise qualitativa, priorização, plano de respostas, análise quantitativa, etc.)

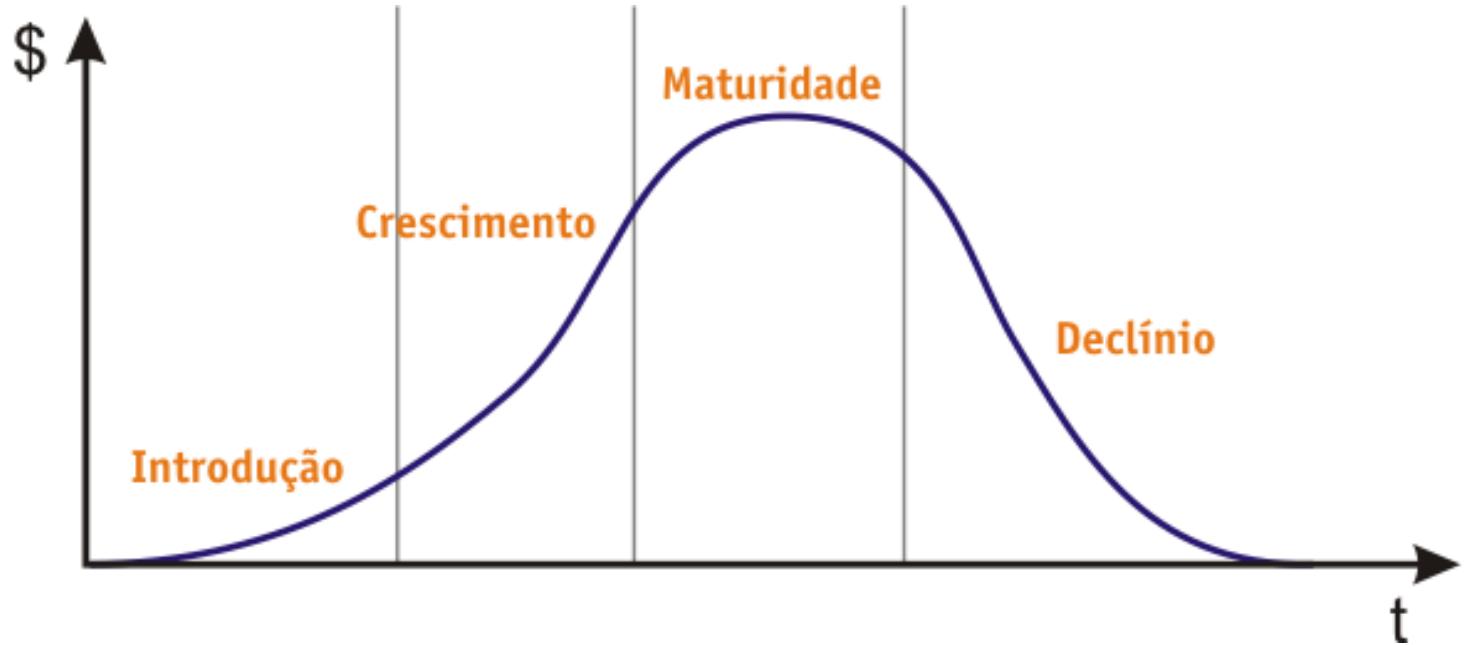
# Aula 10

ITIL



# Ciclo de vida do produto

- Um projeto é executado para produzir um produto.
- O produto é entregue, o mesmo entra em operação.
- Um produto de software em operação a disposição de usuários, entende-se como serviço de TI.



# ITIL

- O acrônimo ITIL se refere à **Information Technology Infrastructure Library** ou Biblioteca de Infraestrutura de Tecnologia da Informação.
- O ITIL pode ser considerado como um conjunto de práticas detalhadas para se fazer um bom gerenciamento de serviços habilitados pela tecnologia da informação.
- O propósito da ITIL é oferecer às organizações uma estrutura prática e flexível como suporte na jornada rumo a transformação digital, ajudando a alinhar os recursos humanos, digitais e físicos para competir no cenário contemporâneo.



# Quem usa o ITIL?

O ITIL pode beneficiar qualquer organização que forneça um produto ou serviço habilitado (ou não) por TI. Ele é usado por organizações em todo o mundo em todas as indústrias e setores:

- Grandes, médias e pequenas empresas
- Governos nacionais, estaduais e locais
- Universidades e instituições de educação
- Organizações não governamentais

Embora o ITIL seja usado em todo o mundo, é difícil encontrar uma lista definitiva e oficial de organizações que o utilizam.

A Axelos (quem atualmente mantém a ITIL ) publica regularmente informações sobre empresas que usam ITIL através de artigos..

# Versões da ITIL

## ITIL V1

- A primeira versão do ITIL foi lançada em meados de 1980 pelo CCTA (Central Computer and Telecommunications Agency), pertencente ao governo do Reino Unido.
- Devido à crescente demanda de TI, a agência passou a criar um conjunto de recomendações, a fim de criar uma **padronização** entre as agências governamentais.
- A maior preocupação era que os contratos entre estas agências e o setor privado, começassem a criar seus **próprios padrões e práticas de gerenciamento de serviços**, trazendo futuros problemas.
- Muitas das práticas baseada no modelo de processo de controle e gerenciamento de Edwards Deming, conhecido também como Ciclo de Deming ou ciclo PDCA (Plan, Do, Check and Act).

## ITIL V2

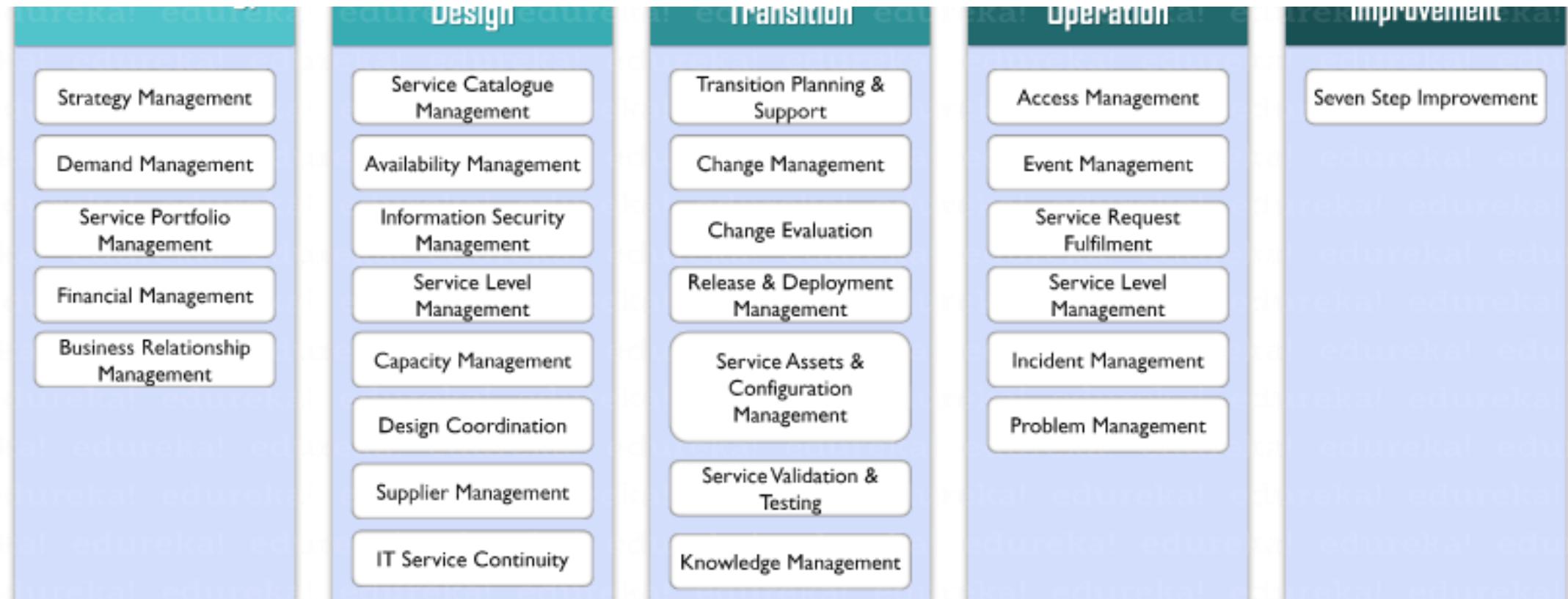
- Em 2001 foi lançada a versão 2 da ITIL
- Os livros da biblioteca foram consolidados em 9 publicações, que davam melhores diretrizes nos processos, atendendo de melhor forma os aspectos do gerenciamento de TI com seus aplicativos e serviços.
- Destas publicações, as mais utilizadas eram o Service Support (Suporte de Serviço) e o Service Delivery (Entrega de Serviço).

## ITIL V3

- Em maio de 2007 ocorre uma nova atualização.
- Composta por 26 processos e 4 funções distribuídos em **5 livros**, retrata um modelo conceitual conhecido como **Ciclo de Vida de Serviço** (estratégia, projeto, transição, operação, melhoria contínua).

## ITIL4 (e não ITIL v4)

- Em 18 de fevereiro de 2019, é lançada a quarta versão, que deixa de usar o acrônimo “v”, passando a se chamar simplesmente ITIL4.
- A estrutura sugere que as publicações não serão tão centralizadas quanto eram na versão anterior.
- No ITIL v3 haviam 5 publicações core, cada uma com algumas centenas de páginas. Qualquer atualização nesta estrutura geraria um grande esforço.

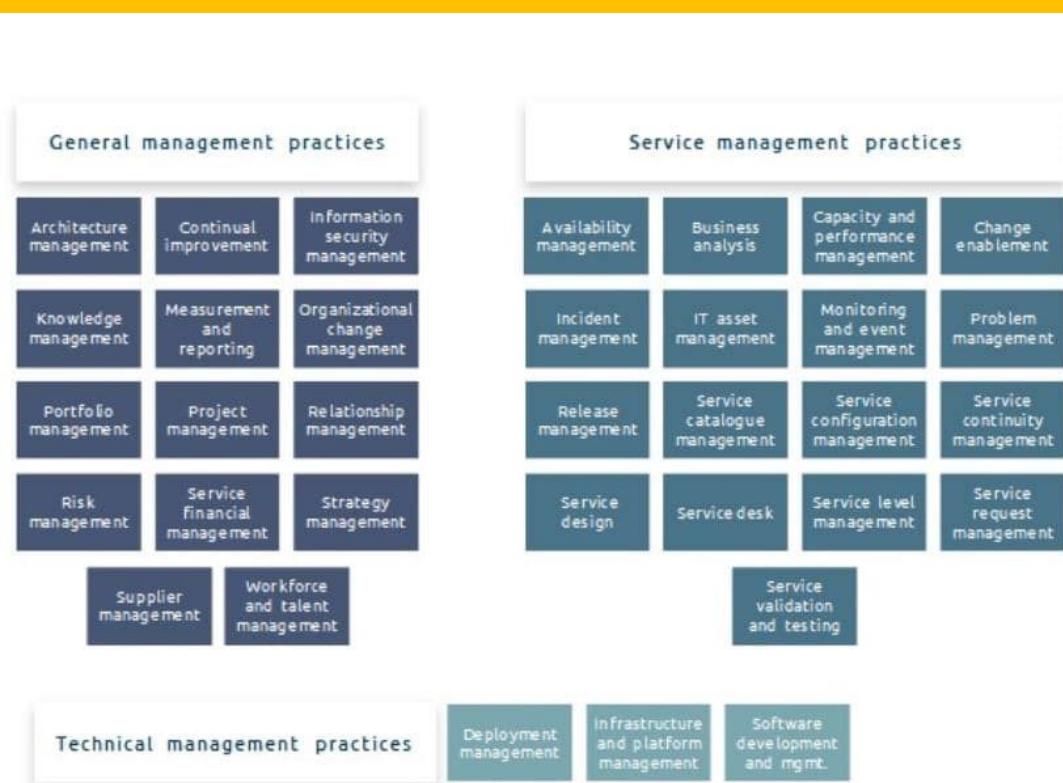


## Estrutura ITIL V3

- Ciclo de vida do serviço
- 26 processos organizado em 5 fases

# Estrutura ITIL 4

- 34 processos
- 3 áreas de competência



# Incidente, Problema e evento

Incidentes, problemas e eventos são conceitos diferentes!

Segundo o ITIL:

- Um evento é uma **mudança de estado** significativa para a gestão de um serviço de TI.
- Já um problema é a **causa raiz** de um ou mais incidentes.
- Incidente é a percepção **degradação ou interrupção do nível de serviço** (disponibilidade, desempenho, qualidade, etc.)

**Exemplo:**

- Imagine que você trabalha em uma grande empresa de metais. No meio da tarde, em pleno funcionamento da empresa, você começa a observar uma lentidão moderada na sua internet. Essa lentidão recebe o nome de **evento**, pois avisa que alguma coisa aconteceu.
- Apesar de ser algo significativo, a qualidade do serviço de internet estabelecida no Acordo de Nível de Serviço (SLA) ainda está dentro dos padrões.
- Um pouco mais para o fim da tarde, a internet cai e fica fora do ar por duas horas. Esse acontecimento caracteriza um **incidente**, pois quebra o fornecimento do serviço e a qualidade da entrega. Além disso, rompe o compromisso firmado no Acordo de Nível de Serviço, que estipula em 30 minutos o limite máximo para a falta de internet. Então, é aberto um chamado para a área de TI, que utiliza o gerenciamento de incidentes para priorizar e solucionar o chamado.

# Incidente x Problema

- A ITIL define assim: um **problema** é "uma causa ou uma possível causa, de um ou mais incidentes".
- E um **incidente** é um único evento não planejado que causa a interrupção do serviço.
  - Os **incidentes** são os episódios desagradáveis que os funcionários de plantão buscam resolver com a maior rapidez possível. E os **problemas** são a causa raiz desses eventos prejudiciais.
  - Um problema pode causar um só incidente ou vários incidentes. E um incidente pode ser rastreado por um só problema ou, às vezes, a vários problemas.

# Exemplos

- **Incidente:** A interrupção de cinco horas que custou US\$ 150 milhões à Delta Airlines em 2016 foi um incidente.
  - Problema: a falta de energia em um centro de operações e, talvez, a ausência de uma política de backup (alta disponibilidade) para casos de falta de energia.
- **Incidente:** a interrupção de 12 horas da App Store que custou à Apple cerca de US\$ 25 milhões foi um incidente.
  - Problema: falhas no serviço de DNS.



<https://money.cnn.com/2016/09/07/technology/delta-computer-outage-cost/>



<https://br.ign.com/apple/2173/news/apple-perde-us-2-milhoes-por-hora-com-app-store-fora-do-ar>

# Gerenciamento de problemas vs. gerenciamento de incidentes

- Problemas e incidentes estão intrinsecamente ligados. Um causa o outro e as equipes têm que prestar atenção nos dois.
- As diretrizes da ITIL exigem o gerenciamento separado de problemas e incidentes.
  - O **gerenciamento de problemas** é a prática concentrada na prevenção de incidentes ou na redução do impacto deles.
  - O **gerenciamento de incidentes** concentra-se em abordar os incidentes em tempo real.

## Prós e contras:

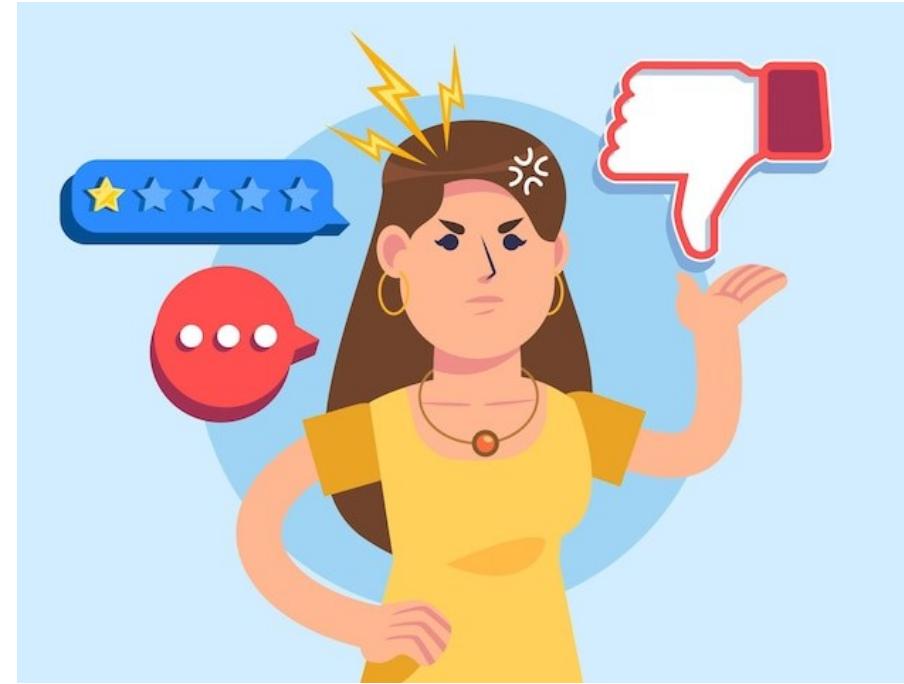
- O benefício da abordagem do ITIL ao separar essas práticas e atribuir igual importância às duas, as diretrizes estão tentando evitar o problema comum em que equipes de TI: estão **sempre apagando o incêndio dos incidentes sem lidar com a causa raiz deles**.
- Se o objetivo principal de um gerente de incidentes é a rápida resolução de incidentes e o objetivo principal de um gerente de problemas é a prevenção, a **combinação dessas funções pode significar que um desses objetivos, ambos vitais para uma empresa, pode ser prejudicado a favor do outro**.
- A desvantagem dessa abordagem é que a separação das duas práticas, pode criar lacunas de conhecimento e a quebra na comunicação entre a resolução do incidente e a análise da causa raiz que leva à causa inherent.



<https://www.atlassian.com/br/incident-management/devops/incident-vs-problem-management>

# Gerenciamento de incidentes

- Um incidente é a interrupção não planejada de um serviço de TI ou a redução da qualidade do serviço prestado.
- Quando os usuários e clientes não têm seus sistemas disponíveis eles perdem a confiança no trabalho da área de TI.
- Uma forma de evitar a perda de credibilidade do departamento é estabelecer um **bom processo de Gerenciamento de Incidentes**.
- São exemplos de incidentes: falta de acesso à internet, servidor fora do ar, mau funcionamento de computadores etc. Eles podem ser identificados pela equipe de TI, por sistemas de monitoramento ou, então, relatados pelos usuários e clientes.





# Exemplo: situação-problema

- Mesmo se você tivesse ficado sem internet por apenas 20 minutos, tempo que está dentro dos limites do SLA, a situação ainda assim caracterizaria um incidente, porque interrompeu um serviço. O Acordo de Nível de Serviço vai ser útil depois, na hora de medir a qualidade do serviço prestado e o atendimento a este SLA.
  - **IMPORTANTE:** usuários não relatam que estão sem acesso à internet. Eles relatam que o sistema não está funcionando!
- Agora, imagine: a internet cai todas as semanas. Nesse caso, é necessário investigar o porquê dessa queda e entender o problema que está por trás dos incidentes.
- O problema pode ser causado pela qualidade do provedor de internet, por pessoas usando a banda da internet para atividades que não deveriam (ex. consumo de multimídia por streaming, etc.) ou por um ataque de hackers (ex. malwares instalados).
- Por trás de uma simples queda de internet pode haver um grande **problema**. No nosso exemplo, a causa para a interrupção do serviço de internet é uma tentativa sistêmica de tentar extrair dados de clientes pela exploração de vulnerabilidades de segurança nas aplicações.

# Resumo de conceitos:

- Um **evento** é um alerta que alguma coisa aconteceu ou pode acontecer;
- Um **incidente** é o efeito, isto é, aquilo que impacta o serviço;
- Um **problema** é a causa raiz ou o porquê de o incidente ter acontecido;
- Todo incidente é um evento, mas nem todo evento é um incidente;
- Todo problema parte de um incidente e, consequentemente, de um evento. Mas o contrário não é verdadeiro.



# Processo de gestão de incidentes

## 1. Identificação de Incidentes

- O primeiro passo para gerenciar incidentes é reconhecer os incidentes. Os incidentes podem ser identificados pela Central de Serviços, por sistemas de monitoramento e pelos próprios usuários e clientes. Dessa forma, os chamados chegam por diversos canais, como chat, e-mail e telefone.

## 2. Registro de Incidentes

- Todos os incidentes devem ser registrados conforme a ferramenta de controle adotada pela organização, que pode ser uma planilha ou um sistema de chamados, por exemplo. O registro é muito importante porque cria um histórico que possibilita a consolidação de uma base de conhecimento. Dessa forma, sempre que os analistas de suporte receberem um chamado, eles poderão consultar a base e verificar se esse incidente já foi resolvido e qual foi a solução encontrada. O registro também facilita a comunicação na hora da “passagem de bastão”.

## 3. Categorização de Incidentes

- Nessa etapa do processo, a Central de Serviços irá classificar o chamado recebido. Qual é o tipo do chamado? É um incidente ou uma requisição? É um chamado de hardware ou software? Se não for um incidente, a Central de Serviços irá delegar o chamado para o processo adequado. Outra tarefa importante nesta etapa de categorização é definir a qual serviço do catálogo o incidente está relacionado.

## 4. Priorização de Incidentes

- É a etapa de definir se o incidente deve ser atendido agora ou pode esperar um pouco. Para isso, é preciso usar critérios relacionados à urgência e ao impacto. Um incidente urgente é aquele que precisa ser atendido imediatamente. Já um incidente impactante é aquele que pode gerar grandes riscos ao negócio. Os incidentes podem ser classificados de acordo com um dos seguintes graus de priorização: “Muito Baixo”, “Baixo”, “Normal”, “Alto” e “Muito Alto”.
- Também é muito comum a classificação através de uma matriz GUT, ferramenta que ajuda a priorizar os incidentes conforme sua Gravidade (intensidade dos impactos), Urgência (o quanto emergencial é a resolução do incidente) e Tendência (os rumos que a situação poderá tomar se não for imediatamente resolvida).

# Processo de gestão de incidentes

## 5. Diagnóstico Inicial de Incidentes

É a fase de entender, de fato, o incidente que foi reportado. Essa atividade compreende todo o processo de busca da Central de Serviços por uma solução que realmente resolva o chamado do usuário ou cliente. Normalmente, os atendentes do primeiro nível de suporte buscam respostas na Base de Conhecimento, em procedimentos técnicos da empresa, junto com os fornecedores ou com os próprios colegas. Também é importante ressaltar que, caso o atendente perceba que faltam informações para a resolução do chamado, ele deve solicitá-las ao usuário ou ao responsável.

## 6. Escalada de Incidentes

Caso o atendente de primeiro nível de suporte não tenha o conhecimento técnico necessário para resolver o incidente, ele delegará a tarefa ao segundo nível de suporte. Essa situação é chamada de escalada. É muito interessante ter essa divisão dos atendimentos em níveis porque gera uma melhor distribuição de tarefas de acordo com as competências da equipe. Afinal, você não vai querer que o seu programador mais caro perca tempo lidando com pequenos incidentes que poderiam ter sido resolvidos por outras pessoas, não é mesmo?

## 7. Resolução de Incidentes

É a fase em que os chamados são realmente solucionados, seja pelo primeiro nível de suporte ou posterior. Além de resolver o pedido do usuário ou cliente, o atendente também deve registrar todas as informações relevantes sobre o incidente e sua resolução. Outro ponto bem importante é garantir de verdade que o incidente foi resolvido, comunicando o cliente.

A resolução de um incidente pode apagar rastros e evidências que poderiam ser utilizados para resolver um problema (causa raiz). Por isso, é preciso cuidado nesse momento.

Por exemplo: para resolver um incidente, como a queda de um servidor, o técnico aperta o botão de “Reset” da máquina. Isto vai apagar registros que poderiam ser utilizados para tentar investigar uma causa raiz para a queda do servidor.

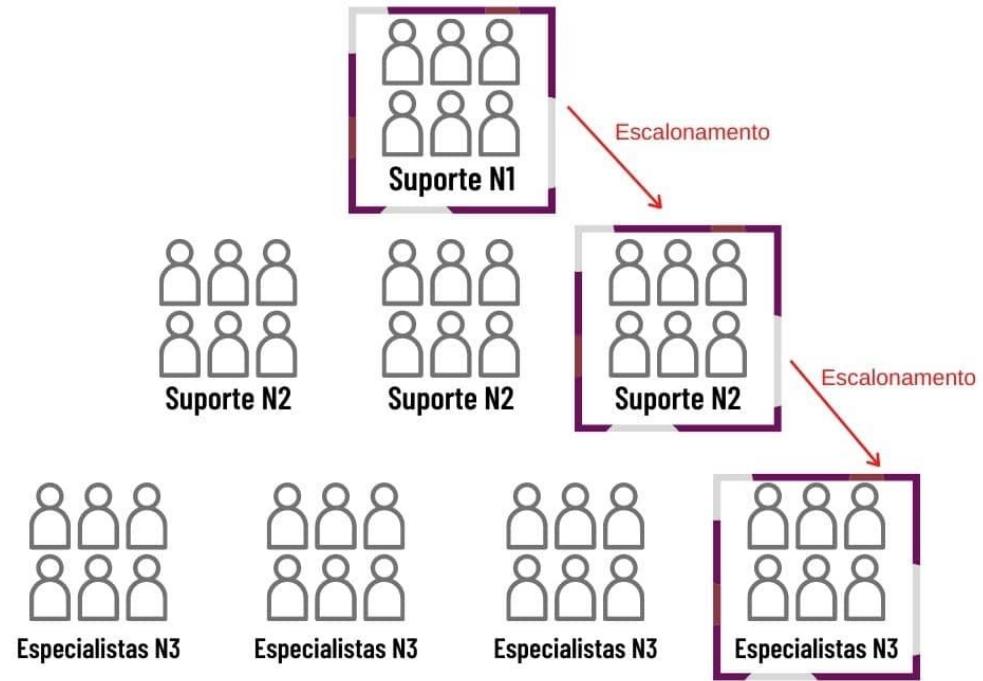
## 8. Fechamento de Incidentes

É o encerramento do chamado, que deve ser documentado para eventuais consultas. Também é preciso exportar as informações para a base de conhecimento, tornando-as acessíveis para outros atendentes. Caso essa base não seja constantemente reabastecida, corre-se o risco de perder tempo tentando encontrar a solução para um incidente que já foi resolvido anteriormente.

# Onde que isso se encaixa nos projetos de software?

A resolução de problemas pode demandar:

- Correções de bugs.
- Desenvolvimento de novas funcionalidades de apoio ao usuário.
- Tratamento de erros.



# Gerência de Configuração

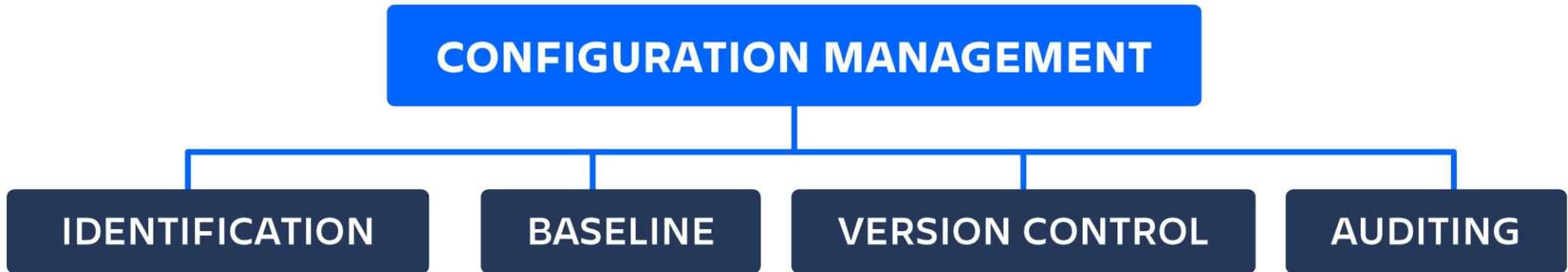
Quando precisamos realizar uma mudança em um produto de software, é importante seguir processos de gerência de configuração.

- Gestão de mudanças
  - Requisitos: novos, alteração, remoção.
  - Analise de impacto.
    - Matriz de rastreabilidade.
    - Tripla restrição: prazo e orçamento com base na mudança de escopo.
  - Comitê Gestor de Mudanças.

## Ferramentas de apoio:

- Service desk: origem da demanda, solicitante, aprovação, etc. (ex. Jira).
- GIT:
  - Commit, deploy, “chamado”, etc.
- Docker/infraestrutura
  - Versões, serviços, capacidades .

# Gerência de Configuração



- O gerenciamento de configuração é um processo de engenharia de sistemas para estabelecer a **consistência dos atributos de um produto ao longo da vida dele**.
- No mundo da tecnologia, o gerenciamento de configuração é um processo de gerenciamento de TI que monitora **itens de configuração individuais de um sistema de TI**.
- Os sistemas de TI são compostos por ativos de TI que **variam em granularidade**: pode representar um software, um servidor ou um cluster de servidores.

# Gerência de Configuração

Roger Pressman, em seu livro Software Engineering: A Practitioner's Approach, especifica que a gerência de configuração de software (GCS) é o:

“ conjunto de atividades projetadas para controlar as mudanças pela identificação dos produtos do trabalho que serão alterados, estabelecendo um relacionamento entre eles, definindo o mecanismo para o gerenciamento de diferentes versões destes produtos, controlando as mudanças impostas, e auditando e relatando as mudanças realizadas.



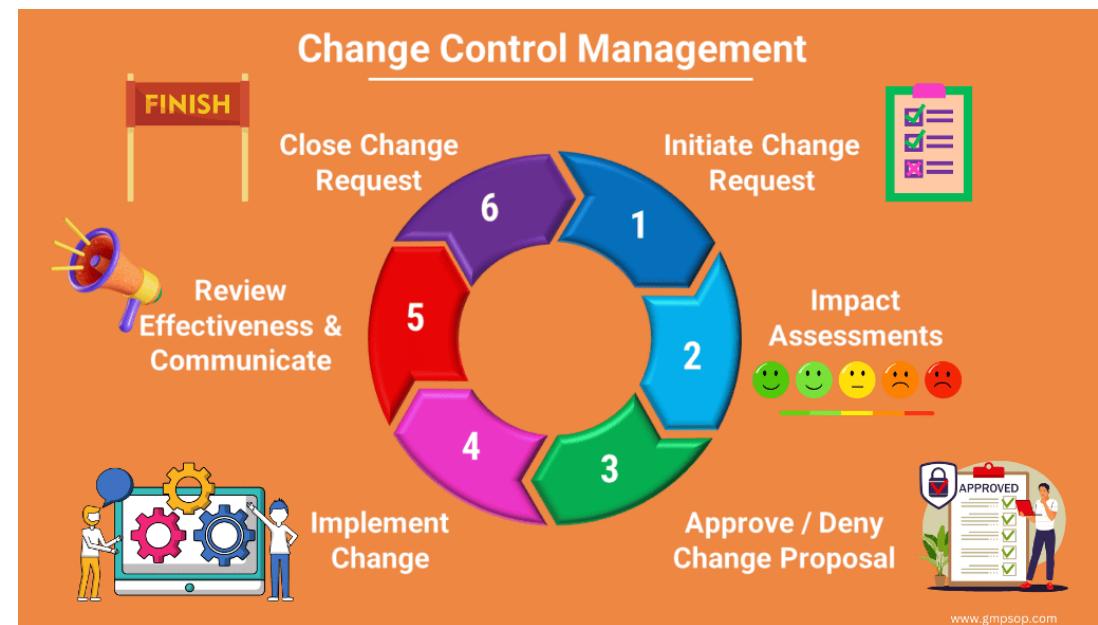
# Quem autoriza a realização de uma mudança?

O Comitê de Controle de Mudanças (CCM) é um grupo de pessoas que toma decisões sobre as mudanças no projeto.

- O grupo é composto de pessoas de autoridade e poder de decisão na organização, e também conta com a participação de pessoas técnicas, até mesmo clientes, que fornecerão assessoria ao grupo.
- Geralmente a última palavra sobre a aprovação ou não da mudança é do patrocinador.

Os projetos de baixa ou média complexidade podem não possuir o CCM e as avaliação do impacto das mudanças é feita pelo patrocinador, juntamente com o gerente do projeto.

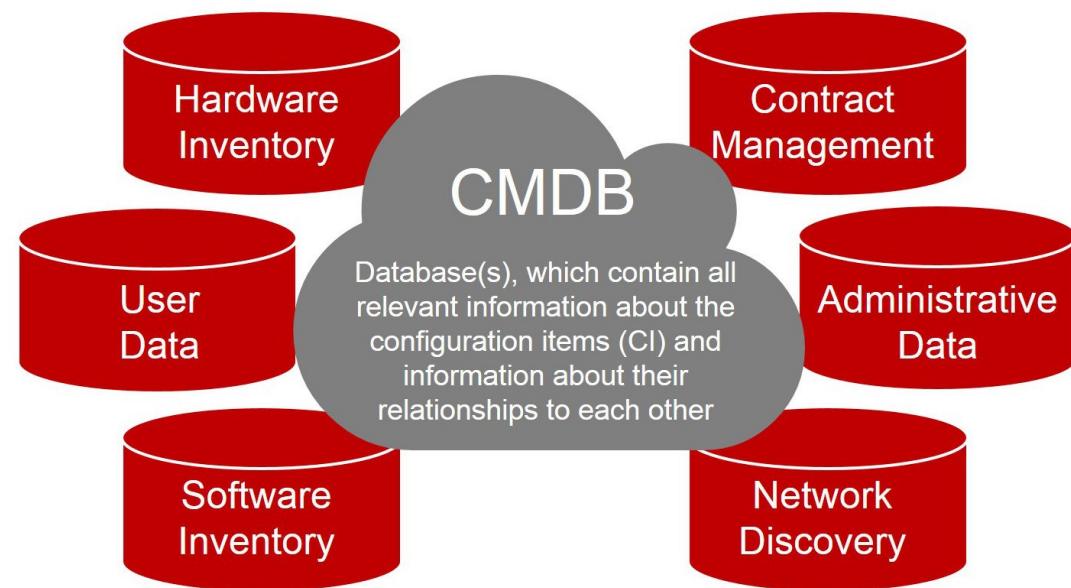
As decisões do CCM devem ser documentadas e comunicadas às partes interessada, a título de informação e para as ações de acompanhamento.



# Onde registrar a configuração dos itens de configuração?

## Configuration Management DataBase (CMDB)

- A configuration management database (CMDB) is a file -- usually in the form of a standardized database -- that contains all relevant information about the hardware and software components used in an organization's IT services and the relationships among those components.
- Within the context of a CMDB, components of an information system are referred to as configuration items (CIs). CIs can be any conceivable IT components, including software, hardware, documentation and personnel.



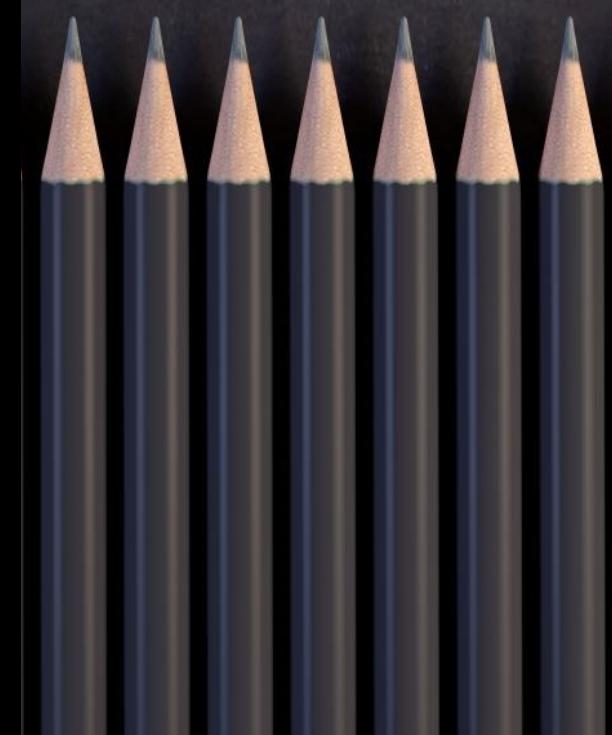
Source:

<https://www.techtarget.com/searchdatacenter/definition/configuration-management-database>  
<https://www.neteye-blog.com/2015/10/is-a-cmdb-a-deciding-factor-for-service-oriented-companies/>



# Aula 11

Aula prática para produção dos  
artefatos da M2



# Aula prática: Produzir os artefatos da M2

Utilize essa aula para elaborar:

- Plano do projeto (escopo, tempo e riscos)
- Documento de arquitetura detalhado
  - Detalhamento dos casos de uso (fluxos alternativos e de exceção)
  - Diagrama de deployment
  - Diagrama de componentes
- Prototipação funcional (sistema em funcionamento pleno: os 3 casos de uso)

# Aula 12

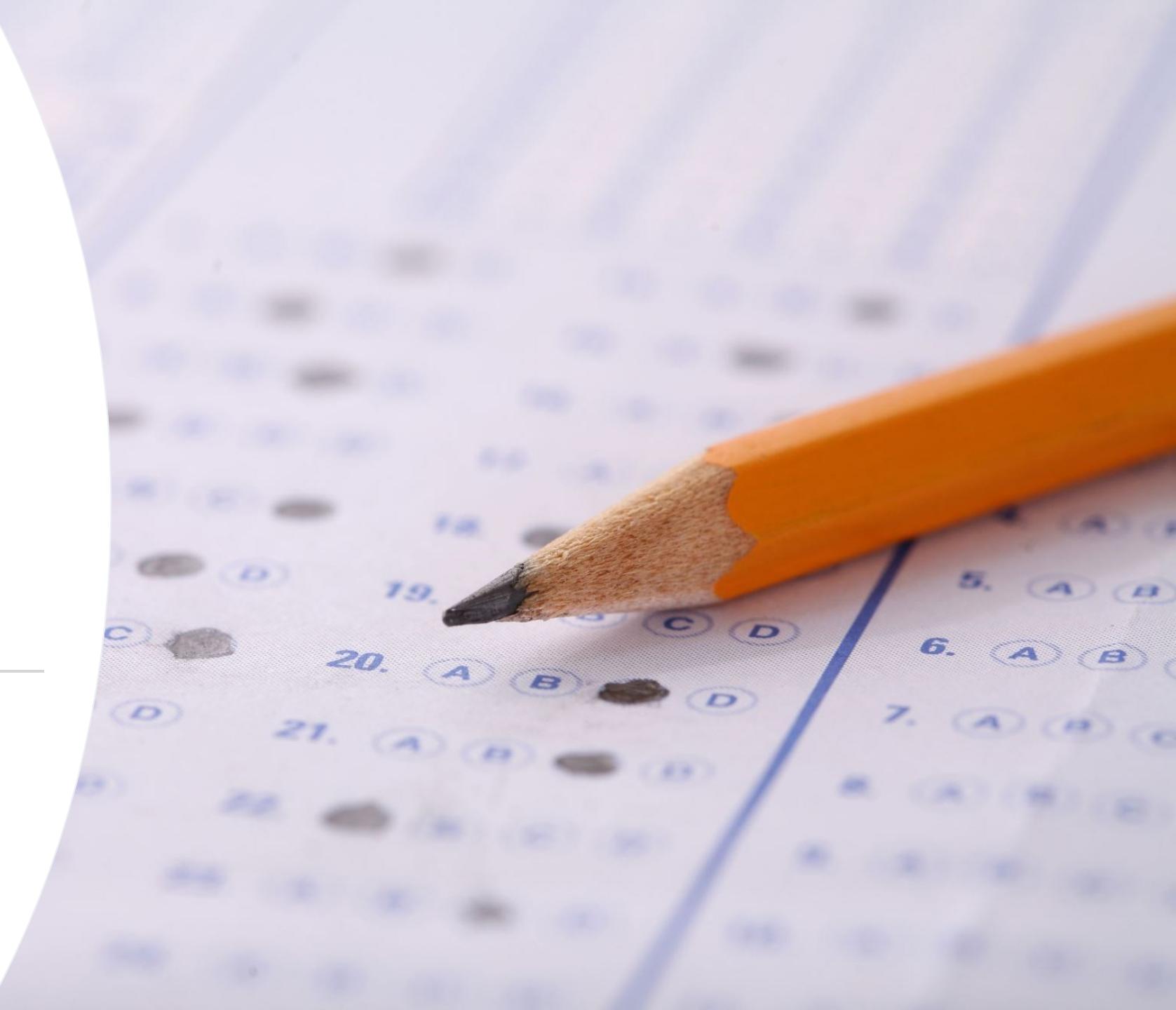
Trabalho M2 –  
Apresentação e entrega



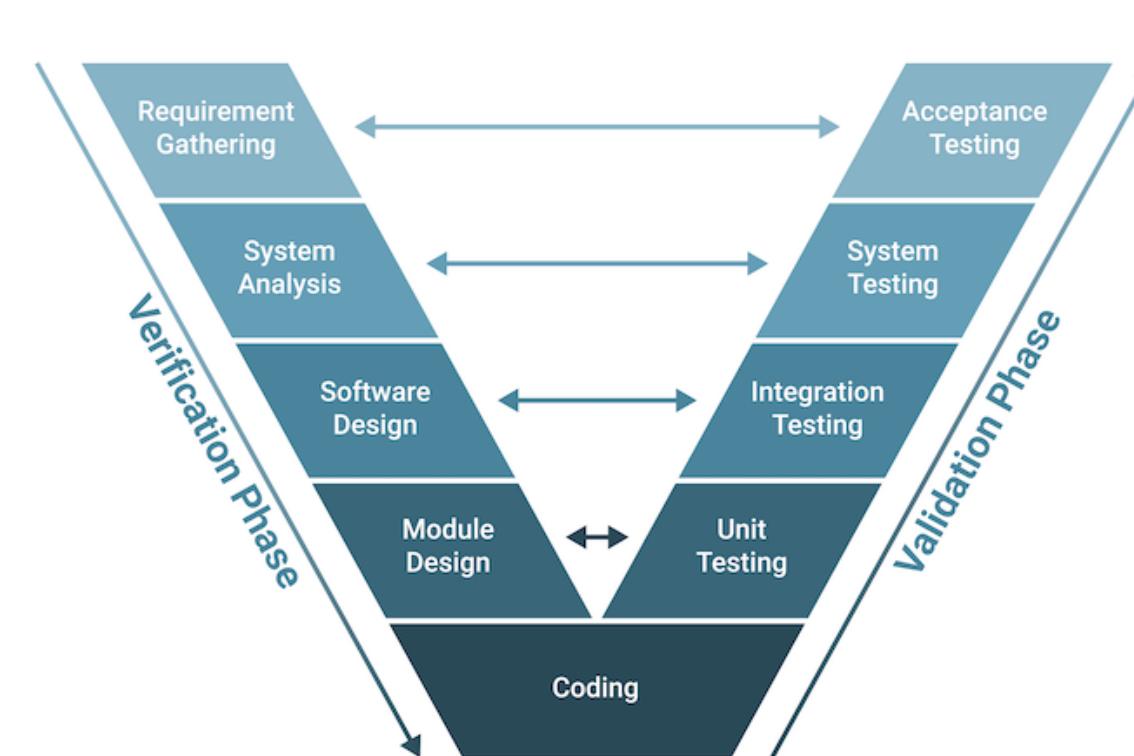
# Aula 13

---

Casos de teste e homologação de software



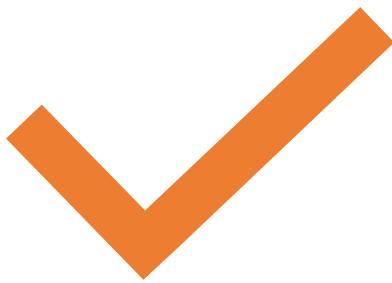
# Verificação e Validação



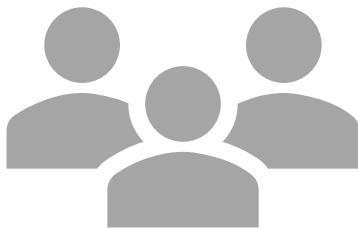
Uma representação gráfica do ciclo de desenvolvimento de sistema.

- Ao lado esquerdo a decomposição de requisitos em elementos de projeto.
- Ao lado direito a integração dos elementos de projeto para atender aos requisitos.

# Diferença entre verificação e validação



Verificação: observa se um produto atende as especificações técnicas previamente estabelecidas entre as partes.



Validação: observa se um produto atende a necessidade dos clientes e demais stakeholders que demandaram o projeto.

# Técnicas de Verificação e Validação

## Verificação

- Testes de unidade
- Teste de caixa branca, preta, cinza
- Cobertura de requisitos
- Automatização de testes

## Validação

- Homologação
- Aceite do produto
- Validação de requisitos no projeto

# Casos de teste

## Objetivo:

avaliar o atendimento aos requisitos do projeto.

- Uma **verificação** abrangente dos requisitos, demanda ao menos um caso de teste para cada fluxo de um caso de uso.
- A relação é: um caso de uso está relacionado a um ou mais casos de teste.

# O que é um caso de teste?

- Em engenharia de software, caso de teste é um **conjunto de condições** usadas para teste de software.
- O caso de teste deve especificar os **valores de entrada** e os resultados esperados (**saídas**) do processamento.
- Ele pode ser elaborado para identificar **defeitos na estrutura interna do software** por meio de situações que **exercitem adequadamente todas as estruturas utilizadas** na codificação;
- O desafio da especificação de casos de testes é encontrar o **menor** subconjunto dos casos de teste possíveis com a **maior** probabilidade de encontrar a maioria dos erros (para cobrir a maior parte possível dos cenários de execução do software).

# Como desenvolver um caso de teste?

Os casos de teste são artefatos criados a partir dos requisitos do software (casos de uso), sendo elaborados em formato de texto , contendo uma sequência de passos para execução.

Os resultados gerados através da sua efetuação são analisados e se o retorno estiver correto, o caso de teste passou, caso contrário falhou.

O analista de testes é responsável por criar o **plano de testes**, composto pelos casos de testes, para validar cada funcionalidade da aplicação.

Os indicadores obtidos deste processo, permitem documentar e avaliar cada funcionalidade do projeto, além de propiciar uma visão completa da qualidade do software construído.

# Casos de teste – template e exemplo

Seção	Descrição
Resumo	Contém uma descrição do caso de teste, descrevendo a finalidade ou o objetivo do teste e o escopo.
Pré-condições	Para cada condição de execução, descreve o estado obrigatório do sistema antes do início do teste.
Entradas	Para cada condição de execução, enumera uma lista dos estímulos específicos a serem aplicados durante o teste. Em geral, eles são denominados entradas do teste e incluem os objetos ou os campos de interação e os valores de dados específicos inseridos durante a execução deste caso de teste.
Ação	Para a execução do teste, são as ações que o usuário deve fazer para que o sistema possa cumprir com o que será testado.
Resultados esperados	É o estado resultante ou as condições observáveis esperadas como resultado da execução do teste. Observe que isso pode incluir respostas positivas e negativas (como condições de erro e falhas).
Pós-condições	Para cada condição de execução, descreve o estado ao qual o sistema deverá retornar para permitir a execução de testes subsequentes.

Caso de Teste		
Projeto:	QualiGO	
Plano de Teste:	PT - Validação Geral	
Suite de Teste:	Validação do módulo (Projetos)	
<b>Caso de Teste:</b> 1	<b>Resumo:</b> Validar Clique no botão (Novo)	
<b>Prioridade:</b> Média	<b>Status:</b> Aprovado	<b>Executor:</b>
<b>Data de criação:</b>		
<b>Pré-condição:</b> Estar logado no sistema		
<b>Passos:</b>	1 - Acessar o Sistema. 2 - Preencher campo (E-mail) de forma válida. 3 - Preencher campo (Senha) de forma válida. 4 - Clicar no botão (Acessar). 5 - Clicar no botão (Novo).	
	<b>Resultado Esperado:</b> Exibir modal para cadastro de novo Projeto	
	<b>Observações:</b> sem observações	
	<b>Comentários:</b> sem comentários	

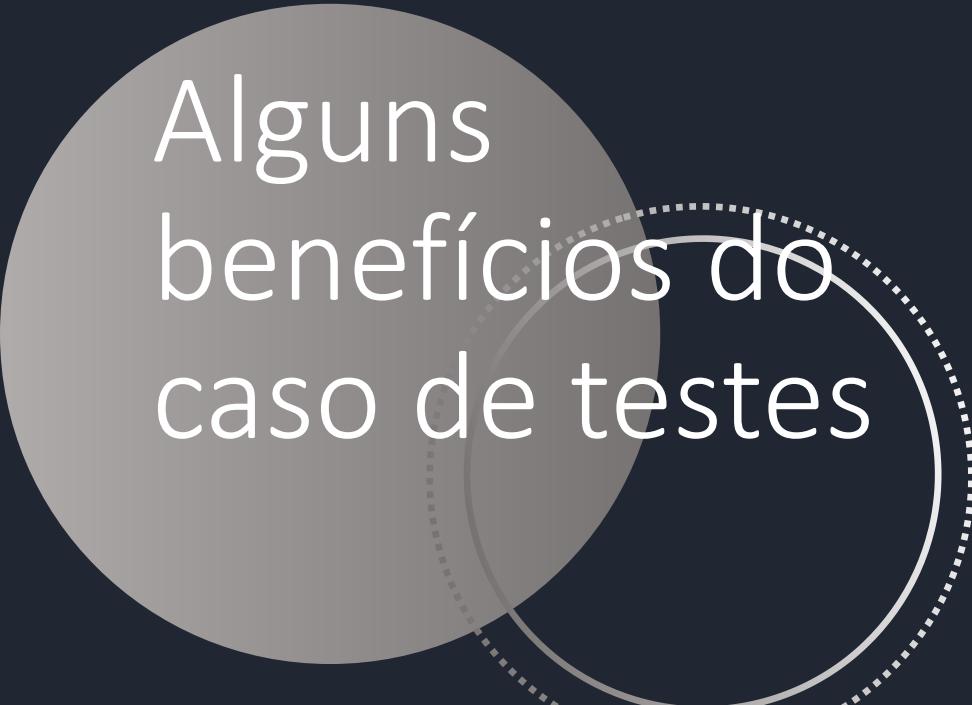
# Exemplos de caso de teste.

Project Name:	Google Email
Module Name:	Login
Reference Document:	If any
Created by:	Rajkumar
Date of creation:	DD-MMM-YY
Date of review:	DD-MMM-YY

TEST CASE ID	TEST SCENARIO	TEST CASE	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_LOGIN_001	Verify the login of Gmail	Enter valid User Name and valid Password	1. Need a valid Gmail Account to do login	1. Enter User Name 2. Enter Password 3. Click "Login" button	<Valid User Name> <Valid Password>	Successful login	Gmail inbox is shown		
TC_LOGIN_001	Verify the login of Gmail	Enter valid User Name and invalid Password	1. Need a valid Gmail Account to do login	1. Enter User Name 2. Enter Password 3. Click "Login" button	<Valid User Name> <Invalid Password>	A message "The email and password you entered don't match" is shown			
TC_LOGIN_001	Verify the login of Gmail	Enter invalid User Name and valid Password	1. Need a valid Gmail Account to do login	1. Enter User Name 2. Enter Password 3. Click "Login" button	<Invalid User Name> <Valid Password>	A message "The email and password you entered don't match" is shown			
TC_LOGIN_001	Verify the login of Gmail	Enter invalid User Name and invalid Password	1. Need a valid Gmail Account to do login	1. Enter User Name 2. Enter Password 3. Click "Login" button	<Invalid User Name> <Invalid Password>	A message "The email and password you entered don't match" is shown			

- A realização de casos de teste envolve a existência de uma massa de dados de teste.
- Os casos de teste não são executados, em regra, com os dados de produção. É necessário conhecer os dados da base para saber quais seriam as entradas e saídas esperadas.

Test Scenario ID	Login-1		Test Case ID	Login-1A		Test Priority	High		
Test Case Description	Login – Positive test case			Post-Requisite	NA				
Pre-Requisite	A valid user account		Test Execution Steps:						
S.No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments		
1	Launch application	https://www.facebook.com/	Facebook home	Facebook home	IE-11	Pass	[Priya 10/17/2017 11:44 AM]: Launch successful		
2	Enter correct Email & Password and hit login button	Email id : test@xyz.com Password: *****	Login success	Login success	IE-11	Pass	[Priya 10/17/2017 11:45 AM]: Login successful		



Alguns  
benefícios do  
caso de testes

**Identificar bugs**

**Automatizar testes**

**Realizar testes regressivos**

**Validar regras de negócio**

# Homologação

- É realizada pelo **cliente!**  
Não pela equipe técnica do projeto.
- É a denominação dada pelo aceite da entrega pelo cliente.
- Para reduzir a subjetividade dos critérios de aceite, os mesmos devem ser documentados nos artefatos do projeto .
  - No **termo de abertura** há uma descrição de alto nível dos critérios de aceite.
  - Os critérios de aceite são detalhados nos **casos de teste**.

# Ferramentas de automação de testes E2E

Os Testes End-to-End (E2E) são um tipo de teste de software que avalia a funcionalidade de uma aplicação em sua totalidade, simulando a interação de um usuário do início ao fim.



O **Selenium** é uma das ferramentas para teste automatizado voltada à testes de aplicações web. É uma ferramenta gratuita, open source que fornece recursos de reprodução e gravação para este tipo de teste. Os testadores podem escrever em várias linguagens de programação.  
<https://www.selenium.dev>



O **Ranorex** é uma das ferramentas para teste automatizado. Possui IDE própria e APIs abertas para especialistas em automação. Suporta testes de ponta a ponta em desktop, web e dispositivos móveis. A ferramenta possui apenas versão paga, mas oferece versão teste grátis.  
<https://www.ranorex.com/>



O **TestComplete** é uma plataforma que automatiza testes do celular, desktop e aplicações web. Ele também permite a utilização de diversas linguagens, como, por exemplo: JavaScript, VBScript e Python.  
<https://smartbear.com/product/testcomplete/>



O **Cypress** é um framework de testes de código aberto. Possui um painel próprio que exibe o que exatamente está acontecendo durante a execução do teste. O Cypress utiliza o node JS como servidor e interpretador de sua linguagem JavaScript, permitindo testes E-2-E e com um conjunto completo de frameworks, como, por exemplo: Mocha, Chai, Jquery, SinonJS. Possibilita a visualização de relatórios dos testes automatizados.  
<https://www.cypress.io/>



# Ferramenta de automação de testes (nodejs)



Jest is a testing framework that provides an easy-to-use and comprehensive testing solution for JavaScript codebases. Developed by Meta, Jest offers features such as parallel test execution, code coverage, built-in matchers for assertions, mocking, and snapshot testing, making it a powerful and versatile testing framework.



Mocha is a flexible testing framework that can be used for both synchronous and asynchronous testing. It supports a variety of assertion styles and provides a variety of hooks for running setup and teardown tasks. Mocha also includes a command-line interface and a web-based user interface for viewing test results.

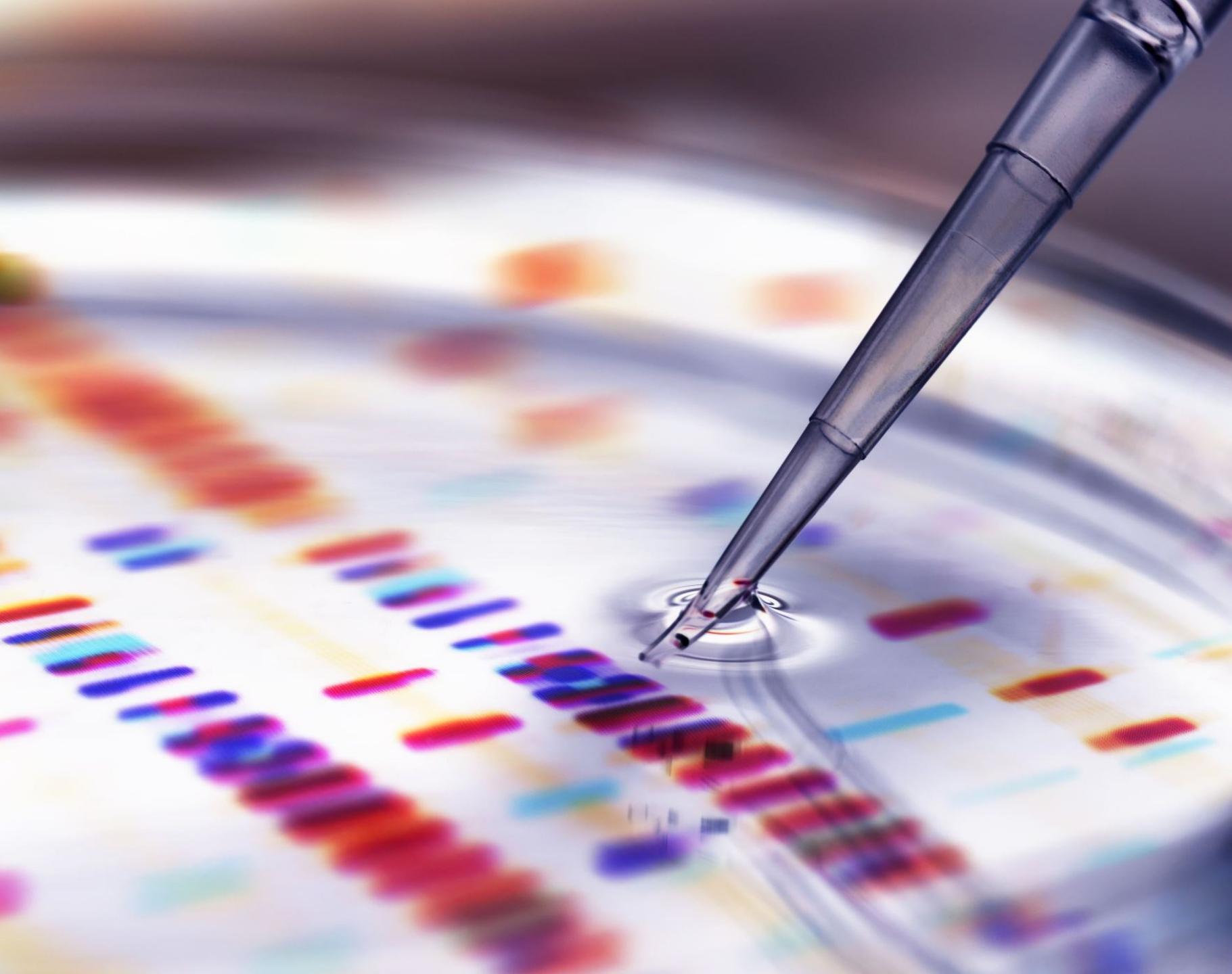


Ava is a JavaScript testing framework that focuses on performance and concurrency. It is designed to run tests concurrently, which allows for faster test execution times. Ava also comes with a built-in test runner and assertion library, making it a self-contained solution for testing JavaScript applications.



Jasmine is a behavior-driven development (BDD) testing framework for JavaScript applications. It provides a clean and expressive syntax for writing tests that closely resemble natural language, making it easy to understand and write tests for technical and non-technical people alike. Jasmine also comes with a built-in test runner and assertion library, making it a comprehensive solution for testing JavaScript applications.

# Teste de Software



# Questão: Qualidade do processo e do produto

Em relação a qualidade de software e análise estática de código-fonte, julgue o item subsequente.

A adoção de um processo de desenvolvimento de software de qualidade garante a qualidade do produto de software desenvolvido.

C

Certo



Errado

# Questão: modelo V

- Ao planejar um projeto de sistema seguindo um ciclo de vida linear, um gerente de projeto resolveu instituir uma estratégia global de teste de software. Considerando-se uma ordem do mais específico para o mais geral, ou seja, terminando-se com o teste de ordem superior, qual a ordem dos testes a serem realizados?

A Teste de integração, teste de validação, teste de sistema, teste de unidade

B Teste de sistema, teste de validação, teste de unidade, teste de integração

C Teste de validação, teste de integração, teste de unidade, teste de sistema

D Teste de validação, teste de sistema, teste de unidade, teste de integração

 Teste de unidade, teste de integração, teste de validação, teste de sistema

# Questão: critérios de avaliação

- Em relação a qualidade de software e análise estática de código-fonte, julgue o item subsequente.  
Os critérios utilizados para avaliar a qualidade de software variam de acordo com o tipo de aplicação a ser avaliada.

**Certo**

# Questão: tipos de testes

- A estratégia de teste software cujo objetivo principal é verificar como um dado software se comporta em um cenário que exige recursos computacionais em quantidades, frequência ou volumes anormais é o teste de



estresse.

B integração.

C regressão.

D unidade.

E usabilidade.

# Exemplo

- Em relação a qualidade de software e análise estática de código-fonte, julgue o item subsequente.
- De acordo com o princípio de clean code, o uso de polimorfismo deve ser evitado, uma vez que esse método dificulta o entendimento e, consequentemente, a manutenção do código.

**Falso**



# Atividade

Monte seu plano de testes!

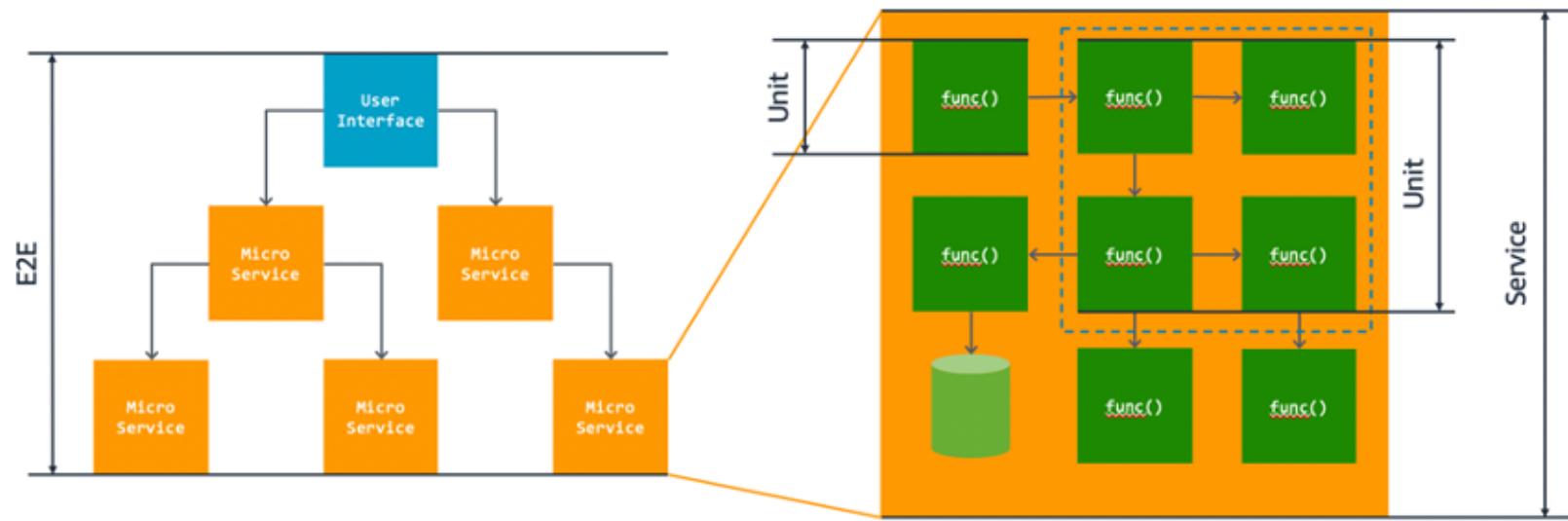
- Estabeleça quais testes de unidade cada caso de uso necessita para ser verificado.
- Utilize como base exemplos de modelos apresentados na aula.

# Aula 14

---



# O que é uma unidade?



- Um *teste de unidade* é um bloco de código que verifica a precisão de um **bloco menor e isolado de código de aplicação**, normalmente uma **função ou um método**.
- Verifica se o bloco de código é executado conforme o esperado, de acordo com a lógica teórica do desenvolvedor por detrás dele.
- O teste de unidade só é capaz de interagir com o bloco de código por meio de entradas e saídas declaradas (verdadeiras ou falsas).

# O que são testes de unidade

- Testes de unidade são o processo em que você testa a **menor unidade funcional** de código.
- Testes de software ajudam a garantir a qualidade do código e são parte integrante do desenvolvimento de software.
- É uma prática recomendada de desenvolvimento de software **escrever software como unidades pequenas e funcionais** e, em seguida, escrever um **teste de unidade** para cada unidade de código.
- Uma vez escrito o teste de unidade, este pode ser automaticamente executado sempre que ocorrer alterações no código do software.
- Se um teste falhar, você poderá isolar rapidamente a área do código que apresenta o bug ou o erro.
- Testes de unidade reforçam paradigmas de pensamento modular e **melhoram a cobertura** e a qualidade dos testes

# Estratégias de testes de unidade

Para criar testes de unidade, você pode seguir algumas técnicas básicas para garantir a cobertura mais ampla de cenários possíveis.

## Verificações lógicas

- O sistema executa os cálculos corretos e segue o caminho certo no código com uma entrada correta e esperada?
- Todos os caminhos do código são cobertos pelas entradas fornecidas?

## Verificações de limites

- Para as entradas fornecidas, como o sistema responde? Como ele responde a entradas típicas, casos extremos ou entradas inválidas?
- Digamos que você espera uma entrada de número inteiro entre 3 e 7. Como o sistema responde quando você usa 5 (entrada típica), 3 (caso extremo) ou 9 (entrada inválida)?

## Tratamento de erros

- Quando há erros nas entradas, como o sistema responde?
  - O usuário foi solicitado a inserir outra entrada? O software trava?

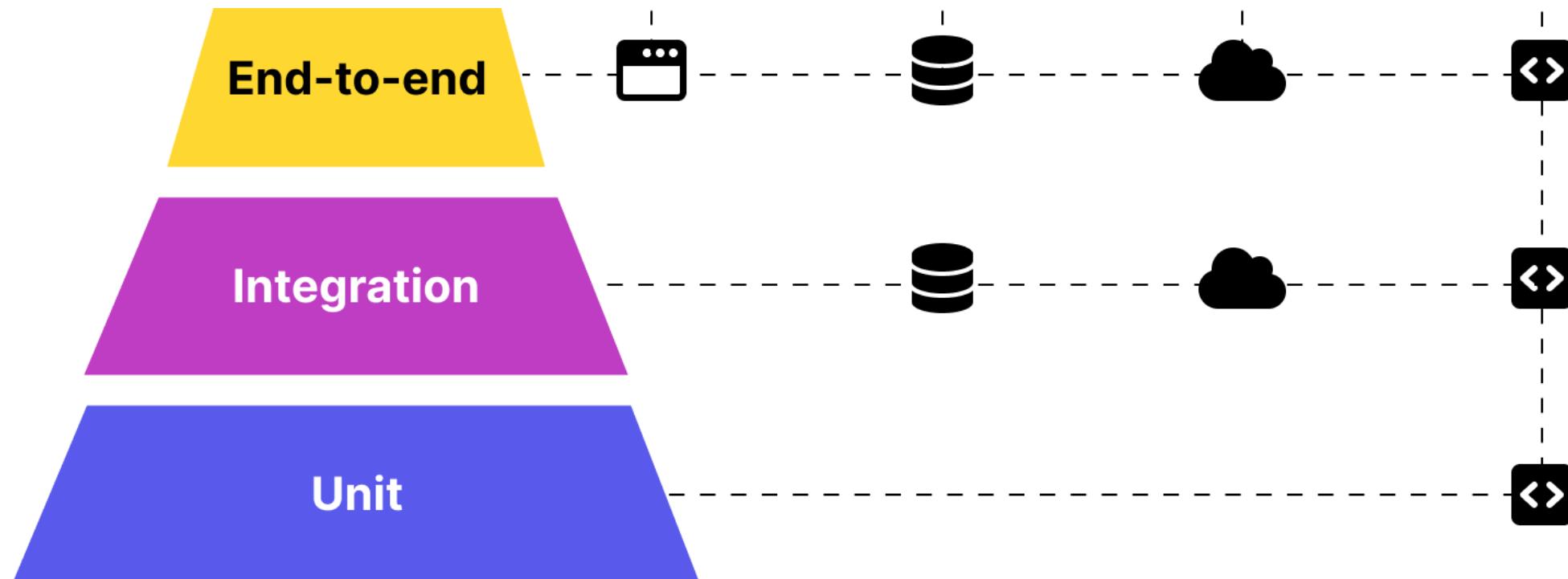
## Verificações orientadas por objetos

- Se o estado de algum objeto persistente for alterado ao executar o código, esse objeto é atualizado corretamente?

# O benefício da documentação

## Documentação

- É importante documentar o código para saber exatamente o que ele deveria estar fazendo. Neste sentido os testes de unidade também funcionam como uma forma de documentação.
- Outros desenvolvedores leem os testes para ver quais são os comportamentos esperados desse código ao ser executado. Eles usam as informações para modificar ou refatorar o código. A refatoração do código o torna mais eficiente e bem composto. Você pode executar testes de unidade novamente para verificar se o código funciona conforme o esperado após as alterações.



Quando necessito de dados externos?

- **O teste de unidade precisa ser executado isoladamente.**
- Quando um bloco de código exige que outras partes do sistema sejam executadas (fora da unidade), talvez estejamos no escopo de um teste de integração.
- É mais fácil escrever testes de unidade para blocos de código pequenos e logicamente simples.



# Quando escrever casos de teste?

---

## Use uma framework de testes de unidade

- Existem frameworks de teste automatizadas para todas as linguagens de programação populares.

## Afirme uma vez (assert)

- Para cada teste de unidade, deve haver apenas um resultado verdadeiro ou falso. Certifique-se de que haja apenas uma declaração de afirmação no seu teste.

## Implemente testes de unidade em novos produtos

- Pode ser difícil reescrever testes de unidade com cobertura ampla em produtos legados, qual não adotaram a prática de automatização de testes.
- Nada impede que novas funcionalidades iniciem esta prática, ainda mais considerando produtos que podem ainda ter anos de evolução pela frente.
- Mas, se a automatização de testes é desejável, é certamente mais fácil mantê-la desde o início do projeto.



# Quando aceitável não escrever testes de unidade

---

## Aplicações de UI/UX (ex. Hotsites, páginas estáticas, etc.)

Quando o sistema principal se preocupa com a aparência e não com a lógica, é possível que não haja muitos testes de unidade para executar. Outros tipos de testes, como testes manuais, são uma estratégia melhor que os testes de unidade nesses casos.

## Bases de código antigas

Escrever testes para envolver um código legado existente pode ser quase impossível, dependendo do estilo do código escrito. Como os testes de unidade exigem dados fictícios, também pode ser muito demorado escrever testes de unidade para sistemas altamente interconectados com muita análise de dados.

## Quando o tempo é limitado (cuidado!!)

- Mesmo com frameworks de apoio à escrita de testes de unidade, esta atividade consome uma quantidade significativa do tempo .
- Depois que se começa a escrever testes, podem ser observadas oportunidades de refatoração , que se forem feitas pode resultar em prazos de desenvolvimento prolongados e problemas de orçamento.



## Unit test – com node.js

---

### **npm test**

Este comando aponta para um script que inicia os procedimentos de teste da aplicação.

O script pode variar de acordo com o biblioteca (framework) de testes escolhida. Este deve ser definido do arquivo package.json

```
"scripts": {  
  "start": "node ./bin/www",  
  "test": "bin/wdio-test"  
},
```

# JEST config

1. Criar o projeto: `npm init -y`
2. Instalar o JEST: `npm install --save-dev jest`
3. Configurar o JEST no npm: `package.json => test: "jest"`
4. Executar os testes: `npm test`  
    `//busca todos os arquivos que tem .test. no nome`
5. Análise de cobertura: `npm test -- --coverage --collectCoverageFrom=".src/**"`

Arquivo fonte "original"

```
src/index.js
class Product{
    static products = [];
    name = "";
    price = 0.0;

    constructor(name, price){
        this.name = name;
        this.price = price;
    }

    static addProduct = (product) => {
        Product.products.push(product);
    }

    static getProducts = () =>{
        return Product.products;
    }
}

module.exports = Product;
```

▼ TERMINAL

```
PASS  test/index.test.js
  ✓ adds 1 + 2 to equal 3 (3 ms)
  ✓ adds Product

-----|-----|-----|-----|-----|-----|
File   | %Stmts | %Branch | %Funcs | %Lines | Uncovered Line #
-----|-----|-----|-----|-----|-----|
All files | 100 | 100 | 100 | 100 |
index.js | 100 | 100 | 100 | 100 |
-----|-----|-----|-----|-----|-----|
Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        0.815 s, estimated 1 s
Ran all test suites.          npm test -- --coverage --collectCoverageFrom=".src/**"
```

Arquivo fonte de teste

```
/test/index.test.js
const Product = require("../src/index");
test('adds 1 + 2 to equal 3', () => {
    expect(1+2).toBe(3);
});

test('adds Product', () => {
    const p = new Product("Bottle of water", 3.5);
    Product.addProduct(p);
    const list = Product.getProducts();
    expect(list.length).toBe(1);
});
```

# Aula 15

---



# Integration test

---

Testes de Integração: têm por objetivo encontrar falhas de integração entre as unidades.

Aqui parte do pressuposto que as unidades já foram individualmente testadas.

Teste de integração é uma etapa do processo de desenvolvimento de software em que módulos ou componentes são combinados e testados em grupo. Esse tipo de teste visa verificar a **corretude, a eficiência e a segurança** da comunicação entre sistemas

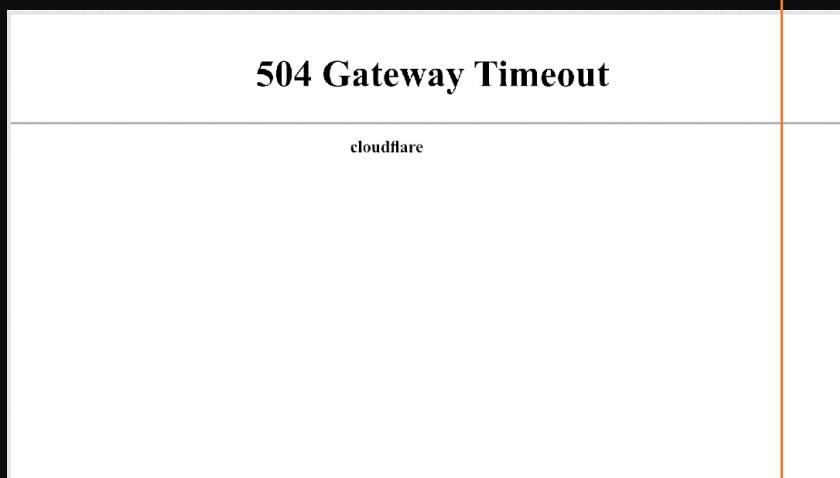
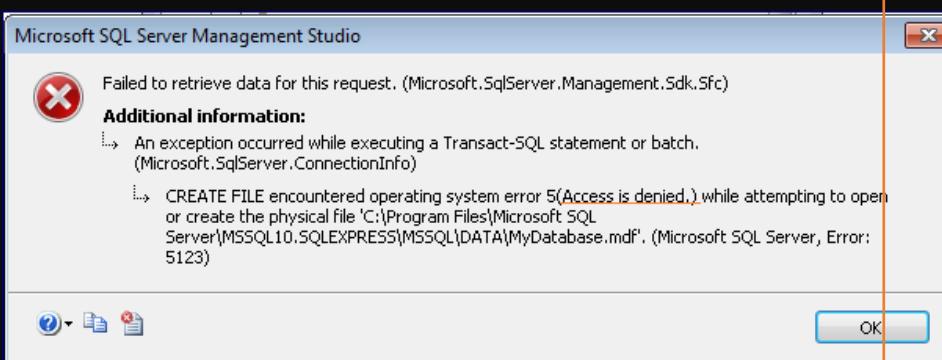


# Teste de integração

- O teste de integração é geralmente realizado em fases, começando com os módulos mais simples e progredindo para os módulos mais complexos.
- Eles podem ser classificados em duas categorias: vertical e horizontal.
  - O teste vertical verifica a interação entre os componentes de um mesmo módulo ou camada (se aprofunda nas diferentes partes de um mesmo "módulo").
  - O teste horizontal testa a interação entre diferentes módulos ou sistemas (transborda para outros "módulos" ou até sistemas).
- O teste de integração ajuda a identificar e corrigir problemas de interface (comunicação), que podem surgir quando os módulos de software são combinados. Esses problemas podem causar falhas, erros e desempenho insatisfatório.

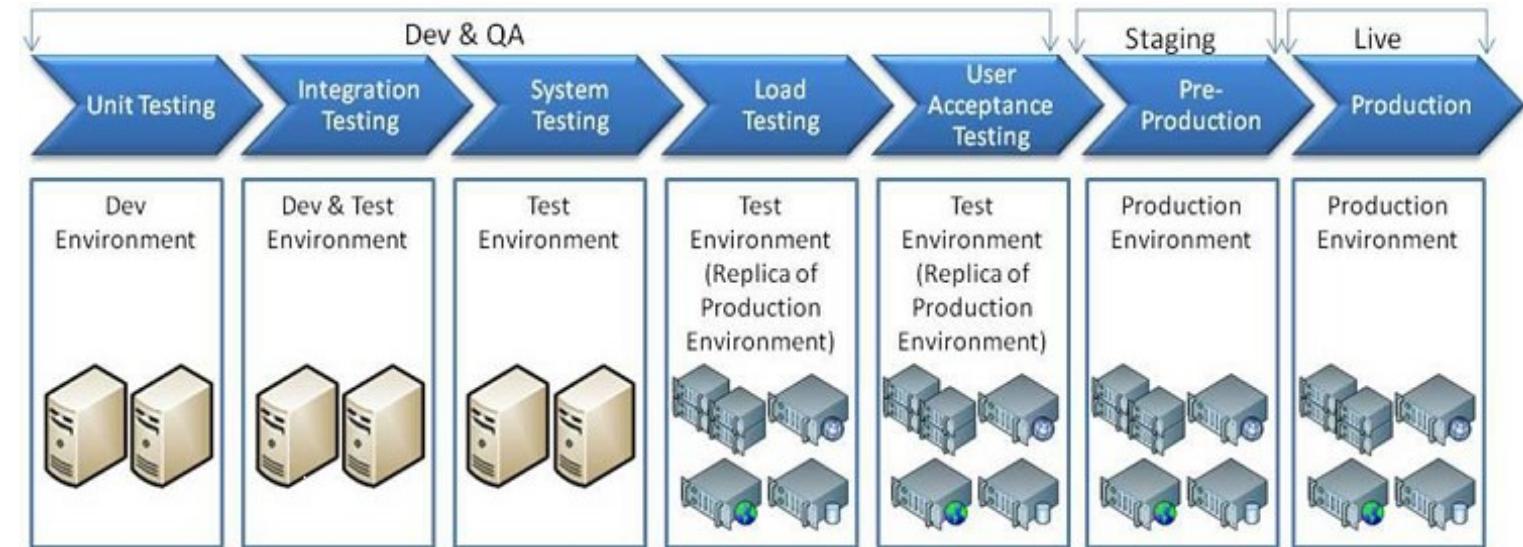
```
113 this.$http.post(Urls.users.register(), this.formItem).then(function (response) {
114   debugger
115   console.log(response)
116
117   this.loading=false
118 }.bind(this)).catch(function (error) {
119
120 //   this.loading=false
121 debugger
122 if (error.response) {
123   console.log(error.response.data);
124   console.log(error.response.status);
125   console.log(error.response.headers);
126 }
```

The screenshot shows a browser's developer tools console. Line 122 contains a conditional statement: `if (error.response) {`. A red arrow points from the word `loading` in the error message below to the `loading` variable in the code at line 117. The error message reads: `! > ReferenceError: Can't find variable: loading` followed by a stack trace: `> error ←` and `< {username: "A user with that username already exists."} = \$1`.



Apoio a encontrar erros  
na mudança de ambiente,  
exemplo:  
desenvolvimento, "qa" e  
homologação.

# Teste de integração



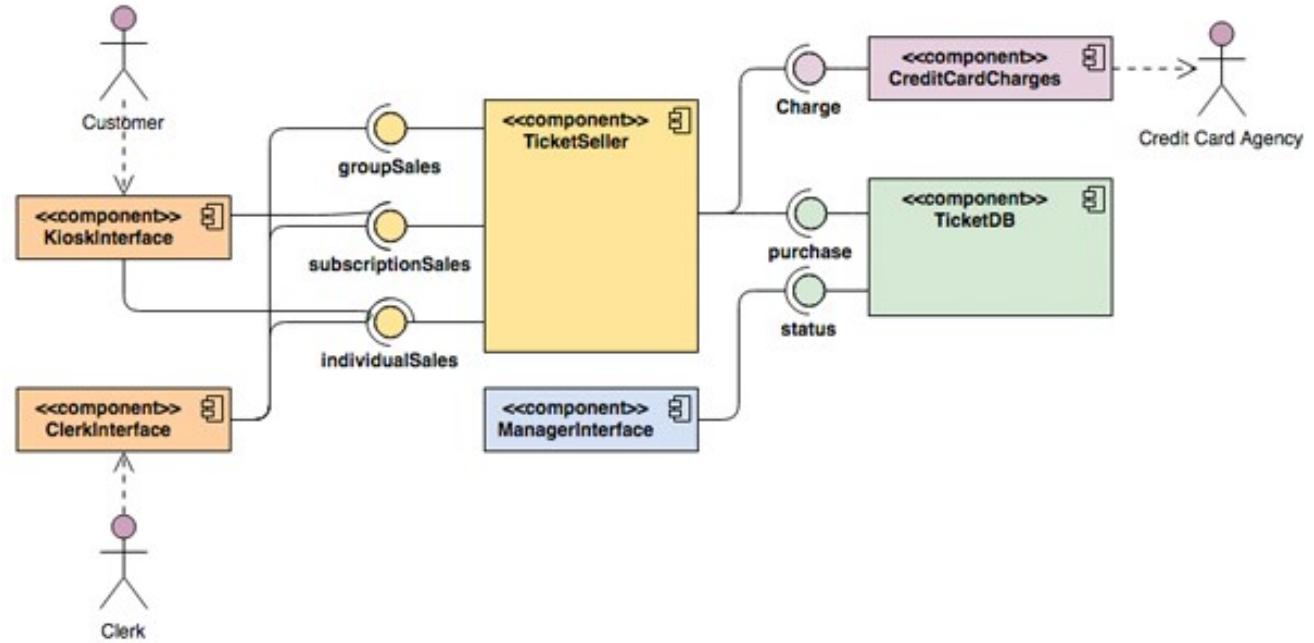
# Teste de integração - Tipos de problemas

- Problemas de comunicação: Os dados podem ser corrompidos durante a transmissão. (ex. Charset)
- Erros de programação: Os módulos podem não estar usando os dados corretamente. (ex. Conversão de tipos)
  - Diferentes versões em diferentes ambientes (ex. versão do JAVA, versão do SQL Server, etc.)
- Restrições de segurança: firewall, antivírus, IPs inacessíveis, etc.
- Erros de dados: Os dados que estão sendo usados no teste podem estar incorretos (em relação aos dados previstos na especificação dos testes).



# Como saber quais os módulos do projeto?

- O conceito de modularização vem do objetivo de segregar um sistema em partes, quais façam sentido existir individualmente, mas que necessitem trabalhar em conjunto para entregar o todo ao usuário.
- Aos módulos podem ser mais claramente identificados pela **segregação física** do "pedaço de software", seja por serem programados em tecnologia distinta ou ser dependência externa.
- Porém muitos módulos podem ser definidos por **segregações lógicas (teste vertical)**. E nesse sentido ferramentas como o diagrama de componentes da UML auxiliam na identificação destas partes.



# O processo do teste de integração

O teste de integração funciona combinando dois ou mais módulos de software para verificar se eles funcionam conforme o esperado. O teste de integração podem verificar se os módulos:

- Comunicam-se corretamente entre si;
- Transferem dados corretamente entre si;
- Produzem os resultados esperados.

Sugestão de processo para escrever os testes de integração:

- **1) Identificação dos módulos**

A primeira etapa para realizar testes de integração é identificar os módulos do sistema que precisam ser testados. É importante ter uma compreensão clara de como esses módulos se relacionam entre si e como as informações são passadas de um módulo para outro.

- **2) Definição de cenários de teste**

Uma vez que os módulos foram identificados, é preciso definir os cenários de teste. Isso envolve determinar quais funções ou operações devem ser testadas em cada módulo e como elas se relacionam com os outros módulos.

- **3) Criação de casos de teste**

Envolve a criação de um conjunto de dados que possa ser usado para testar cada cenário de teste. Os casos de teste devem incluir entradas específicas e resultados esperados para cada cenário de teste.

- **4) Execução dos testes**

Durante a execução dos testes, a equipe de testes deve monitorar cuidadosamente o sistema para garantir que tudo esteja funcionando corretamente. Se um teste falhar, a equipe deve identificar a causa do problema e trabalhar para corrigi-lo.

- **5) Verificação e documentação dos resultados**

# Atividade

Escreva os testes de integração do seu projeto.

- As funções de integração de dados (CRUD) podem ser testadas neste momento.
- Os serviços web do projeto podem ser testados neste momento.

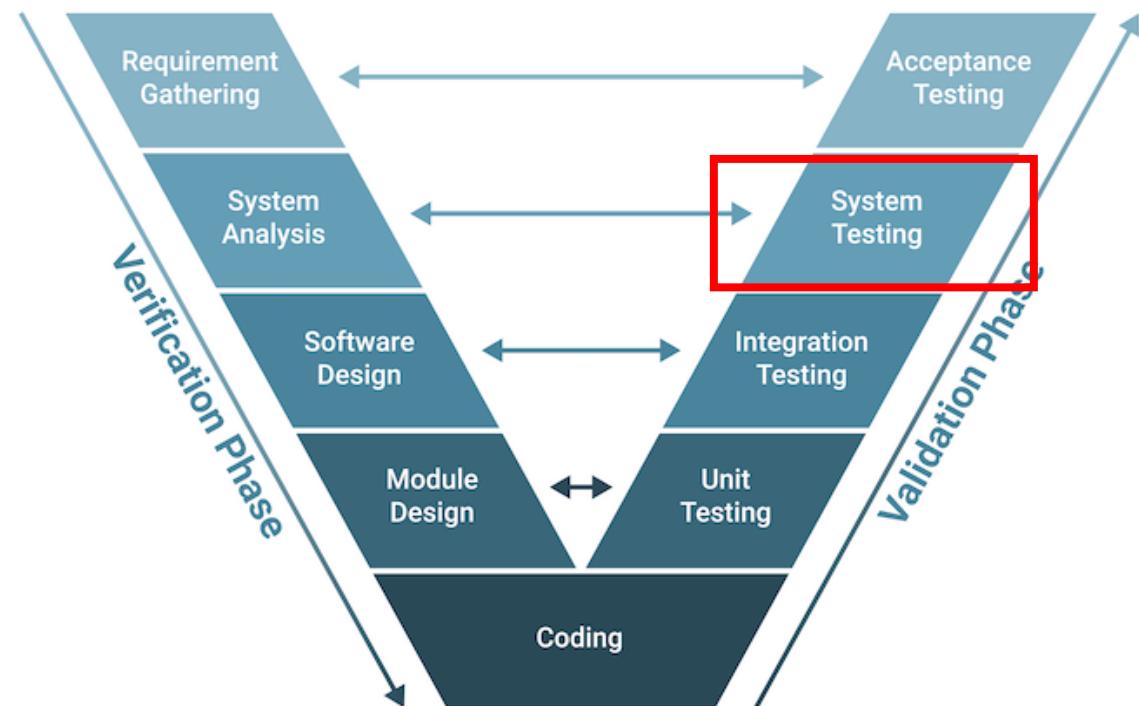
# Aula 16

---



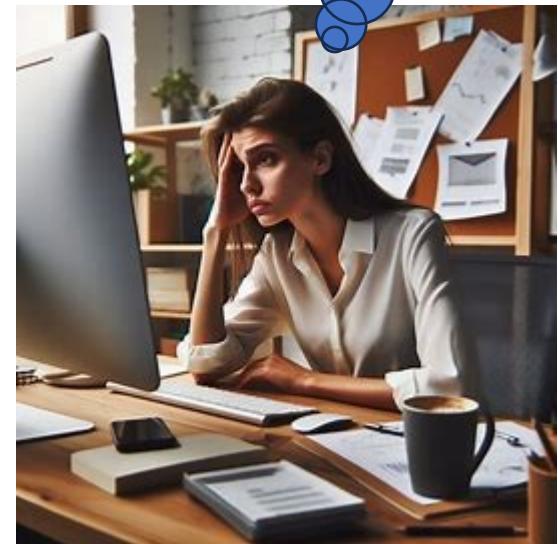
# System test

- O teste de sistema é a última etapa que precede a entrega para validação do usuário.
- Nesta etapa a equipe do projeto busca verificar se o sistema atende aos requisitos especificados, já sob a ótica de usuário (com acesso por telas) e não mais interagindo com código da aplicação.



# Teste de sistema

- É visto como um atributo negativo a entrega de software sem uma análise de funcionamento prévia da equipe.
- Testar sob a ótica do usuário pode demandar atenção e **envolvimento** com o projeto e regras de negócio.





Delegar o teste ao usuário

# Teste exploratório

Testes exploratórios é a abordagem cujo foco está na descoberta e depende da orientação do testador individual para descobrir defeitos que não são abrangidos no escopo de outros testes.

A prática de testes exploratórios ganhou impulso nos últimos anos. Testadores são incentivados a incluir testes exploratórios como parte de uma estratégia de cobertura de teste.

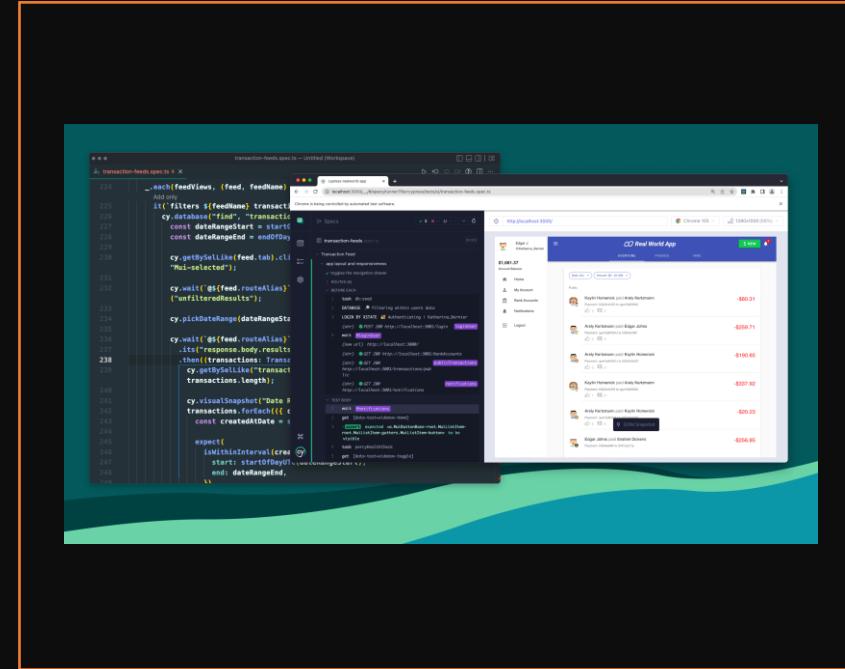
Atenção!

Deve ser utilizada como uma técnica complementar, após a execução dos testes sistematizados.

Este teste pode encontrar problemas no software, como naveabilidade, mensagens de retorno, entre outros erros mais explícitos.

# Automatizando testes de sistema

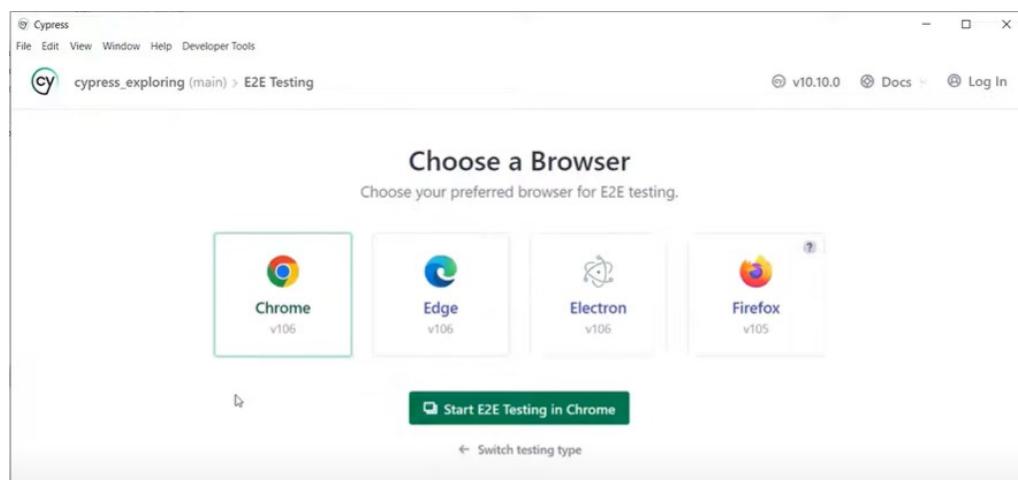
<https://www.cypress.io/>



# Instalando o Cypress

Dentro de um projeto node (pode-se utilizar um projeto existente (ex. cliente ou servidor) ou criar um projeto próprio.

```
npm install cypress  
npx cypress open
```



The screenshot shows the VS Code interface with the title bar 'File Edit Selection View Go Run Terminal Help'. The 'EXPLORER' pane on the left shows a file tree with a red bracket highlighting the 'cypress' and 'e2e' folders. The 'TERMINAL' pane at the bottom shows the command 'GET /\_cypress/iframes/cypress%5Ce2e%5Cspec-copy-2.cy'.

```
File Edit Selection View Go Run Terminal Help ← →  
EXPLORER ... JS spec-copy-2.cy.js U X  
CYPRESS_EXPLORING cypress > e2e > JS spec-copy-2.cy.js > ...  
|> describe('empty spec', () => {  
|>   it('passes', () => {  
|>     cy.visit('https://example.cypress.io')  
|>   })  
|> })  
< 3 JS spec-copy-1.cy.js U JS spec-copy-2.cy.js U JS spec.cy.js M  
< 3 fixtures support JS cypress.config.js  
< 3 TERMINAL PROBLEMS OUTPUT  
GET /_cypress/iframes/cypress%5Ce2e%5Cspec-copy-2.cy
```

# Escrevendo o teste automatizado com cypress

**Exemplo de código cliente:** o formulário que deve ser preenchido pelo usuário.

File: <http://127.0.0.1:5500/index.html>

```
<html>
  <body>
    <form action="doit.html">
      <input type="text" name="query" />
      <input type="submit" value="Send" id="bt_submit" />
    </form>
  </body>
</html>
```

**Exemplo de código cliente (retorno esperado):**

File: doit.html

```
<html>
<body>
  <h1 id="result">
    Your data has being received.
  </h1>
</body>
</html>
```

**File: spec.cy.js > arquivos “spec” podem ser criados automaticamente pela interface web.**

O primeiro arquivo é recomendável que seja. Os demais podem ser criados manualmente.

```
describe('empty spec', () => {
  it('passes', () => {
    cy.visit('https://example.cypress.io')
  })
})

it('Tesing form submission' ,()=>{
  cy.visit('http://127.0.0.1:5500/index.html')
  .get('[name="query"]')
  .type("green")
  .get('#bt_submit')
  .click()
  .wait(1000)
  .location()
  .get('#result')
  .should('be.visible');
})
```

# Cypress: exemplo 2

```
<html>
  <head>
    <script>
      function submitForm(){
        let form = document.getElementById("form_actor");
        let formData = new FormData(form);
        let xhr = new XMLHttpRequest();
        xhr.onreadystatechange=function(){
          if(this.status==200 && this.readyState==4){
            let divMsg = document.getElementById("success_message");
            divMsg.style.display = "block";
          }
        }
        xhr.open('POST',form.action , true);
        xhr.send(formData);
      }
    </script>
  </head>
  <body>
    <h1>Formulário de registro de novo ator</h1>
    <form id="form_actor" action="http://localhost:3333/actor" method="POST">
      <label for="firstName">First name</label>:<br/>
      <input name="first_name" id="firstName" /> <br/>
      <label for="lastName">Last name</label>:<br/>
      <input name="last_name" id="lastName" /><br/><br />
      <input type="button" id="bt_submit" onclick="submitForm()" value = "Registrar" />
    </form>
    <div id="success_message" style="display:none">
      Requisição processada!!
    </div>
  </body>
</html>
```

```
it('Testing new actor from submission' ,()=>{
  cy.visit('http://127.0.0.1:5500/aula7_client_html_new_actor.html')
  .get('[name="first_name"]')
  .type("Charlie")
  .get('[name="last_name"]')
  .type("Chaplin")
  .get('#bt_submit')
  .click()
  .wait(1000)
  .get('#success_message')
  .should('be.visible');

});
```

```
const express = require('express');//npm install express
const mysql = require('mysql');//npm install mysql
const bodyParser = require('body-parser'); // npm install body-parser

const app = express();
app.listen(3333);//initialize web server
//objects to handle POST message body
const jsonParser = bodyParser.json() // create application/json parser
const urlencodedParser = bodyParser.urlencoded({ extended: false })
//initialize mysql connection
const MYSQL_IP="localhost";const MYSQL_LOGIN="root"; const MYSQL_PASSWORD="root";

let con = mysql.createConnection({
  host: MYSQL_IP,
  user: MYSQL_LOGIN,
  password: MYSQL_PASSWORD,
  database: "sakila"
});

con.connect(function(err) {
  if (err){
    console.log(err); throw err;
  }
  console.log("Connection with mysql established");
});

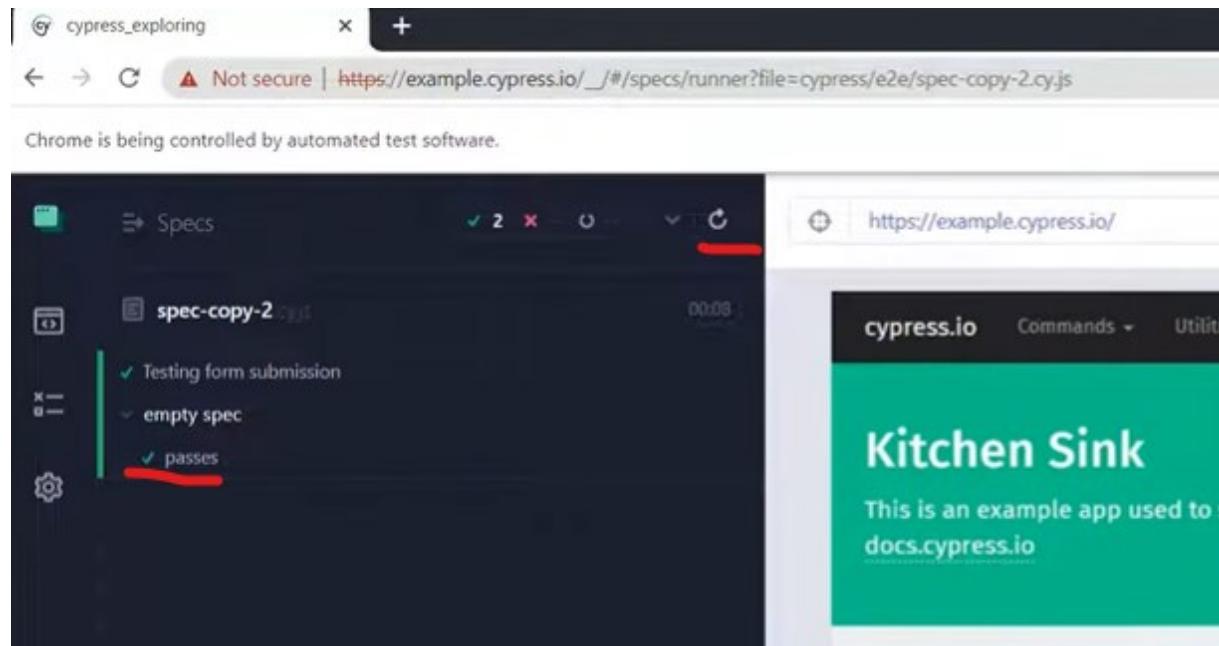
app.post('/actor', urlencodedParser, function (req, res) {
  const firstName = req.body.first_name;
  const lastName = req.body.last_name;
  const parameters = [firstName, lastName];
  let sql = "insert into actor (first_name, last_name) values (?,?)";
  con.query(sql, parameters, function (err, result) {
    if (err){
      res.status(500);
      res.send(JSON.stringify(err));
    }else{
      res.status(200);
      addCorsHttpHeaders(res);
      res.send("<div id='success_message'>Novo ator inserido com sucesso!</div><br/>" + JSON.stringify(result));
    }
  });
});

function addCorsHttpHeaders(httpResponse){
  httpResponse.setHeader("Access-Control-Allow-Origin", "*");
  httpResponse.setHeader("Access-Control-Allow-Methods", "POST,GET,OPTIONS,PUT,DELETE,HEAD");
  httpResponse.setHeader("Access-Control-Allow-Headers", "X-PINGOTHER,Origin,X-Requested-With,Content-Type,Accept");
  httpResponse.setHeader("Access-Control-Max-Age", "1728000");
}

console.log("node express is running");
```

# Executando o teste automatizado

```
npx cypress open
```



# Aula 17: Aula prática - Produzir os artefatos da M3

---

## a) Escrever o plano de testes

Escrever para todos os 3 casos de uso, com factibilidade de serem testados manualmente (base de testes precisa estar disponível)

- Todos os fluxos de todos os casos de uso tem ao menos um caso de teste para cobertura?

## b) automatização dos testes:

Escrever para 1 caso de uso, incluindo os 3 fluxos.

- **Unidade:** Não necessários get/sets. Se não houver função de negócio a ser testada, testar as funções de repositório/DAO com apoio de "mock" (dados simulados – fixos em código)
- **Integração:** com banco de dados, com servidor http.
- **Sistema:** execução de cada fluxo do caso de uso escolhido, de ponta a ponta (E2E).

# Aula 18

Trabalho M3 – Apresentação e entrega

