# PARTICIPANTES, FERRAMENTAS E O PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

**Projeto de Programas – PPR0001**

# Componente Humano / Participantes

- **Analista**
  - Deve ter conhecimento do domínio do negócio
  - Não precisa ser um especialista, mas deve ter um conhecimento básico na área de domínio para se comunicar com especialistas
  - Entender as necessidades dos clientes e repassar a equipe
  - Deve ter conhecimento relativos à modelagem de sistemas:
    - Tradutor: linguagem dos especialistas do domínio e dos desenvolvedores
  - Ter bom relacionamento interpessoal (+ importante que tecnológico)

  - ❖ Analista de negócios: entender o que o cliente faz, por que faz, e como o processo pode ser otimizado por um sistema
  - ❖ Analista de sistemas: traduz necessidades do usuário em características de um produto de software

# Componente Humano / Participantes

- **Projetistas**
  - Projetar alternativas de solução do problema resultante da análise, i.e. adicionam aspectos tecnológicos a tais modelos
  - Gerar especificações de uma solução computacional detalhada
  - Na prática podem existir diversos tipos:
    - Projetistas de Interface
    - Projetistas de Rede
    - Projetistas de Banco de Dados
    - (...)

# Componente Humano / Participantes

- **Arquitetos de software**
  - Elaborar a arquitetura de um software como um todo
  - Define quais serão as subdivisões do sistema e como serão as interfaces entre eles
  - Deve ser capaz de tomar decisões técnicas detalhadas
    - E.g. decisão sobre um aspecto em relação ao desempenho do sistema
  - Geralmente presentes em grandes equipes e projetos complexos

# Componente Humano / Participantes

- **Programadores**
  - Responsáveis pela implementação do sistema
  - Proficientes em uma ou mais LPs e capazes de ler modelos de projeto
  - Participam principalmente dos processos finais do desenvolvimento
  - !! um bom programador não necessariamente é um bom analista
  - !! um bom analista não necessariamente é um bom programador

# Componente Humano / Participantes

- ***Testers***
  - Avaliam se as funcionalidades do software estão em acordo com as especificações realizadas (geralmente utilizam *checklists*)
  - Podem realizar testes de qualidade (desempenho e confiabilidade)
- **Avaliadores de Qualidade**
  - Avaliam desempenho e confiabilidade do software
  - Avaliam qualidade e adequação durante o desenvolvimento

# Componente Humano / Participantes

- **Gerente de projeto**
  - Gerência ou coordenação do projeto
  - Define quem faz o quê e quando
  - Estipula orçamento e tempo
  - Definir processo de desenvolvimento / metodologias
  - Define e busca recursos de hardware e software
  - Realizar o acoplamento das atividades

# Componente Humano / Participantes

- **Especialistas do domínio**
  - Pessoas que possuem muito conhecimento no domínio do projeto
  - São os clientes:
    - Cliente contratante: indivíduo(s) que solicitaram o desenvolvimento
    - Cliente usuário: indivíduo(s) que utilizará o sistema
  - Em projetos que são desenvolvidos para a massa, são escolhidos representantes para clientes:
    - Equipe de marketing
    - Usuários comuns do tipo de software sendo desenvolvido

# Ferramenta para Modelagem

- Evolução de computadores => necessidade de evolução na modelagem de sistemas e processos de desenvolvimento
  - Analogia: construção de um software monoprocessado é igual a produção de um software com processamento paralelo?

- Histórico:
  - **1950/60**: sistemas simples; técnicas de modelagem mais simples; desenvolvimento *ad hoc* (direto ao assunto); fluxogramas e diagramas de módulos
  - **1970**: computadores avançados e acessíveis (expansão do mercado computacional); sistemas mais complexos; surgimento da programação estruturada; modelos mais robustos começavam a surgir

# Ferramenta para Modelagem

- Histórico:
  - **1980**: computadores mais avançados e baratos; interface gráfica; consolidação da análise estruturada
  - **1990-1996**: surgimento do paradigma de orientação a objetos. Várias propostas de técnicas surgiram:
    - OOAD – *Object-Oriented Analysis and Design*
    - *Booch Method*
    - OOSE – *Object-Oriented Software Engineering*
    - OMT – *Object Modeling Technique*
    - *Responsibility Driven Design*
    - *Fusion*
      - Problema: diferentes notações gráficas para representar uma mesma perspectiva

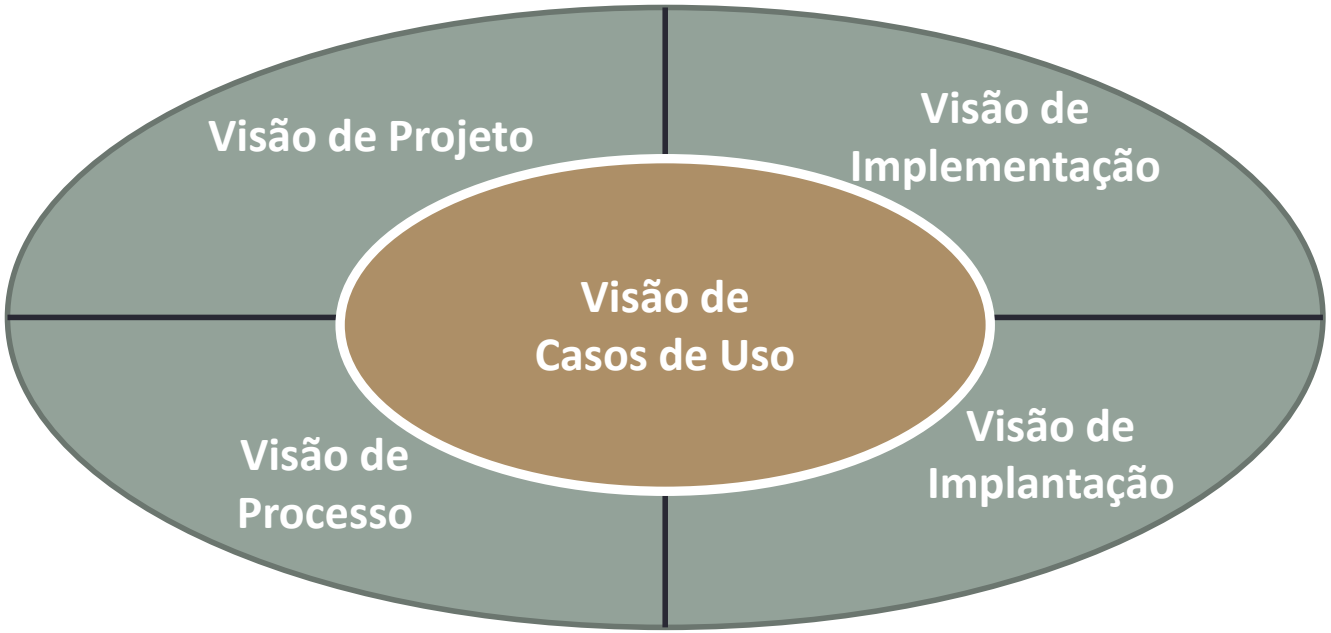
# Ferramenta para Modelagem

- Histórico:
  - **1996-2000**: percebeu-se a necessidade de um padrão de modelagem para indústria e academia; surge a UML.

- UML (Linguagem de Modelagem Unificada)
  - Grady Booch, James Rumbaugh e Ivar Jacobson (“três amigos”)
  - Aproveitou as melhores notações existentes na época
  - Aprovada como padrão em 1997
  - Em desenvolvimento / evolução: versão atual 2.5 (2015)
  - É uma linguagem visual para modelar sistemas O.O
  - Pode representar diversas perspectivas de um sistema

# Ferramenta para Modelagem

- UML (Linguagem de Modelagem Unificada)
  - Cada elemento possui uma sintaxe e uma semântica
    - Sintaxe: forma de desenho / Semântica: significado do elemento
  - É extensível: pode se adaptar às características específicas de cada projeto



# Ferramenta para Modelagem

**Visões de um sistema**

- Visão de Casos de Uso: descrição do sistema do ponto de vista externo; conjunto de interações entre o sistema e os agentes externos; visão inicial que direciona o desenvolvimento das outras visões
- Visão de Projeto (design): ênfase nas características do sistema - estrutura e comportamento - e nas funcionalidades visíveis
- Visão de Processo: ênfase nas características de concorrência, sincronia e desempenho do sistema
- Visão de Implementação: gerenciamento de versões do sistema, e do agrupamento dos módulos/componentes
- Visão de Implantação: distribuição física do sistema em seus subsistemas e conexão entre as partes

# Ferramentas no Processo de Desenvolvimento

- O processo de desenvolvimento de software é complicado e altamente cooperativo
- O uso de ferramentas auxiliares podem ajudar:
  - Na construção de modelos do sistema
  - Na integração do trabalho da equipe
  - No gerenciamento do andamento de desenvolvimento
  - (...)

- Softwares para suporte ao ciclo de desenvolvimento:
  - Ferramentas CASE
  - Ambientes de Desenvolvimento – IDE (*Integrated Development Environment*)

# Ferramentas no Processo de Desenvolvimento

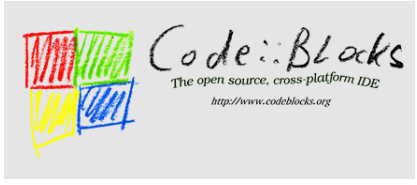
**Ferramentas CASE:**

- *Computer Aided Software Engineering* – Engenharia de software auxiliada por computador
  - Criação de diagramas (XMI – XML Metadata Interchange) [UML]
  - Engenharia *Round-Trip*: interage com o código-fonte do sistema
    - Engenharia Direta: diagramas → código
    - Engenharia Reversa: código → diagramas
  - Rastreamento de requisitos: facilitar a localização de artefatos criados com base em um dado requisito.
  - Ferramenta: JUDE/ASTAH

# Ferramentas no Processo de Desenvolvimento

**Ambientes de Desenvolvimento:**

- Possibilitam codificação com diversos funcionalidades adicionais que facilitam este processo:
  - Depuração de código-fonte (facilita procura por erros de lógica)
  - Pré-Compilação (procura-se por erros no momento da escrita)
  - Refatoração: modificações no código que não alterem seu comportamento (renomear variáveis, classes, ...)
  - Ferramenta: Dev-C++, CodeBlocks, NetBeans, Eclipse



# Ferramentas no Processo de Desenvolvimento

**Outras Ferramentas:**

- Realização de testes automatizados
- Gerenciamento de versões de documentos (dropbox, SVN)
- Monitoração e averiguação de desempenho (tempo de execução, uso de memória, tráfego de dados
- Tarefas de gerenciamento (jxProject, OpenProj, Project Planner)

# O que é Software

- O que é um software?

# O que é Software

- O que é um software?
  - *"Software são os programas de computador e a documentação associada" (Sommervile, 2003)*

- Diversos tipos de produtos possuem documentação associada:
  - Casas e edifícios: planta
  - Remédios: bula
  - Eletrodomésticos: especificações de montagem, manual do consumidor
  - Software: especificações de implementação, manual do usuário

  - E.g.: um jogo é produzido através de especificações, principalmente do game designer, e pode vir acompanhado de um manual de uso.

# Sobre o desenvolvimento de Software

- O desenvolvimento de software é um processo complexo
- Dados levantados pelo *Standish Group* (Chaos Report, 2017):

| | 2011 | 2012 | 2013 | 2014 | 2015 |
|---|---|---|---|---|---|
| SUCCESSFUL | 29% | 27% | 31% | 28% | 29% |
| CHALLENGED | 49% | 56% | 50% | 55% | 52% |
| FAILED | 22% | 17% | 19% | 17% | 19% |

- Motivos (Goldratt, 2002):
  - Expansão de funcionalidades
  - “Lapidação a ouro” (quando tenta-se fazer detalhes muito elaborados no design ou na implementação que muitas vezes nem foram requisitados)
  - Negligência ao controle de qualidade
  - Cronogramas “super” otimistas

# Sobre o desenvolvimento de Software

- Motivos (Goldratt, 2002) (...):
  - Trabalho em muitos projetos ao mesmo tempo
  - **Planejamento pobre**
  - Projetos orientados a pesquisa (resultados da pesquisa são incertos)
  - Pessoal não qualificado para as atividades
  - Distância dos *stakeholders*

- Tentativa de minimizar problemas => estudos sugerem uso de processos ou metodologias de desenvolvimento de software:

  - ICONIX
  - EUP (*Enterprise Unified Process*)
  - RUP (*Rational Unified Process*)
  - XP (*Extreme Programming*)
  - OPEN (*Object-Oriented Process, Environment and Notation*)

# Sobre o desenvolvimento de Software

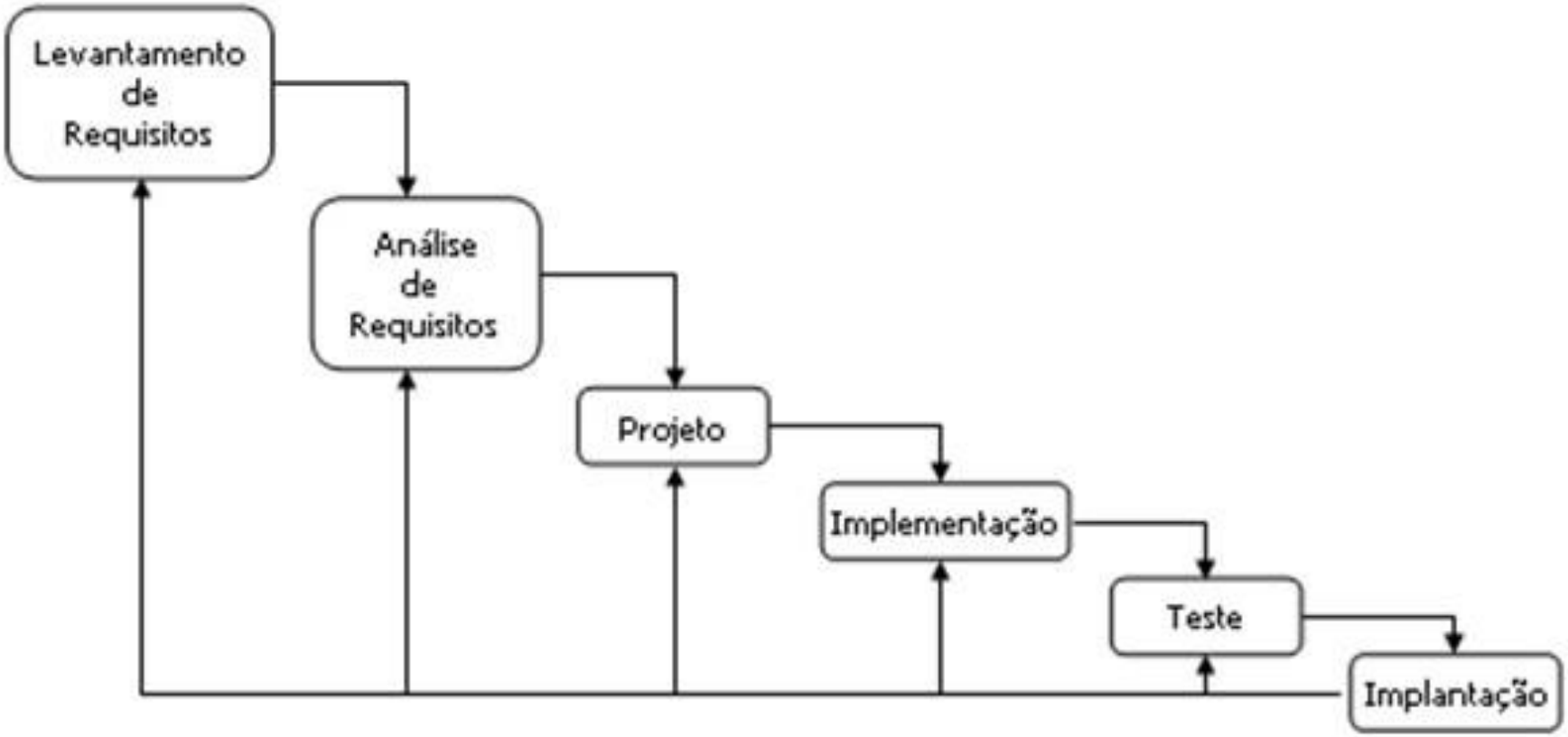
- O objetivo das metodologias de desenvolvimento de software é definir os pontos chaves de um projeto:
  - Quais atividades serão executadas
  - Quando cada atividade será executada
  - Quem executará cada tarefa
  - Definir pontos de controle para verificar andamento do desenvolvimento
  - Definir um padrão para o desenvolvimento

# Desenvolvimento de Software

- Atividades típicas no desenvolvimento de software:
  - Levantamento de Requisitos
  - Análise
  - Projeto
  - Implementação
  - Testes
  - Implantação

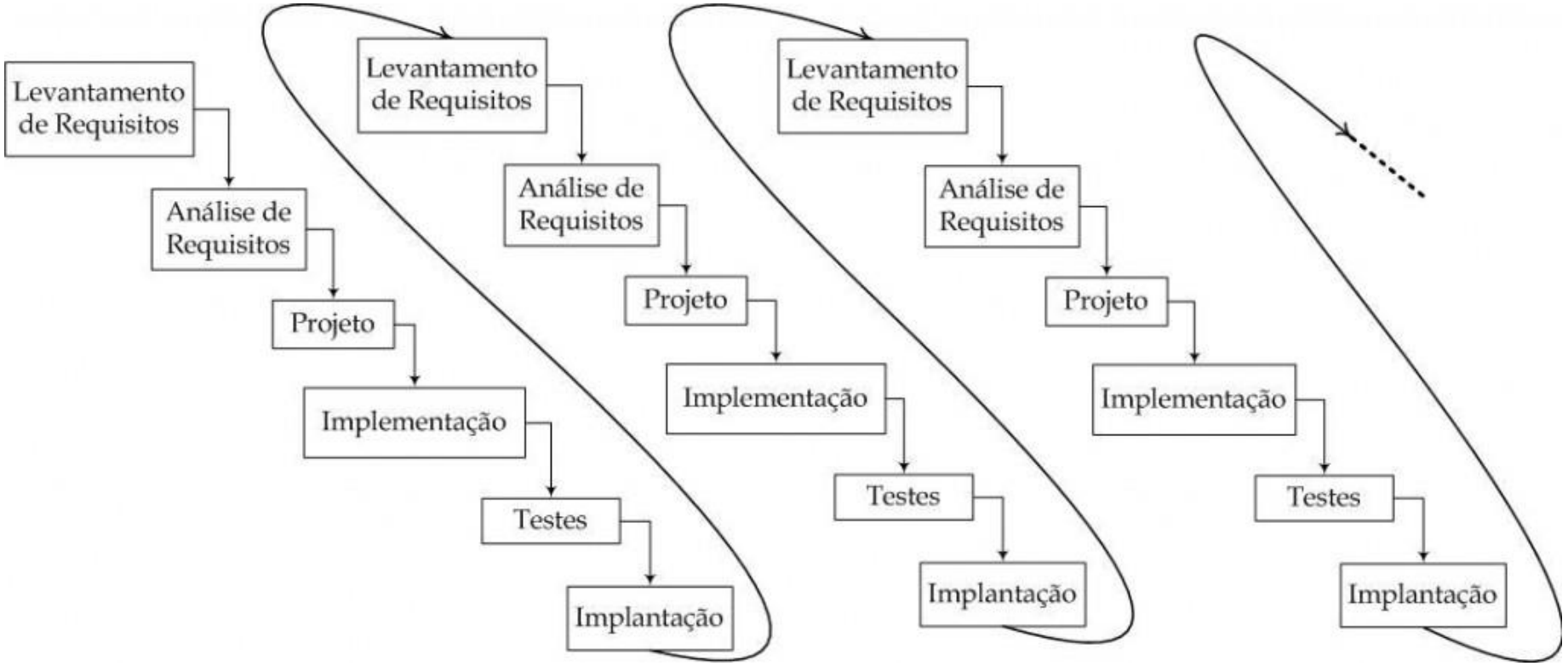
# Metodologia de Desenvolvimento

- **Modelo Cascata**



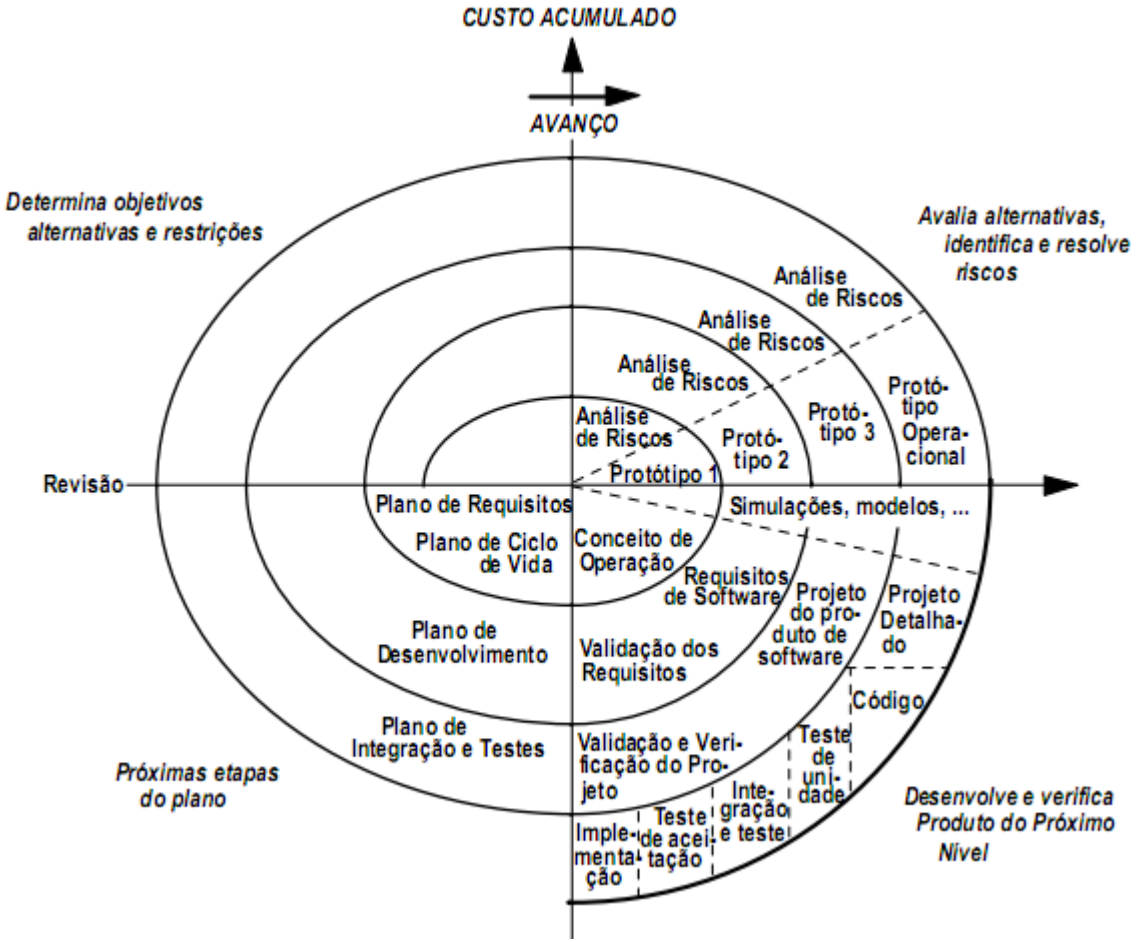
# Metodologia de Desenvolvimento

- **Modelo Iterativo Incremental**



# Metodologia de Desenvolvimento

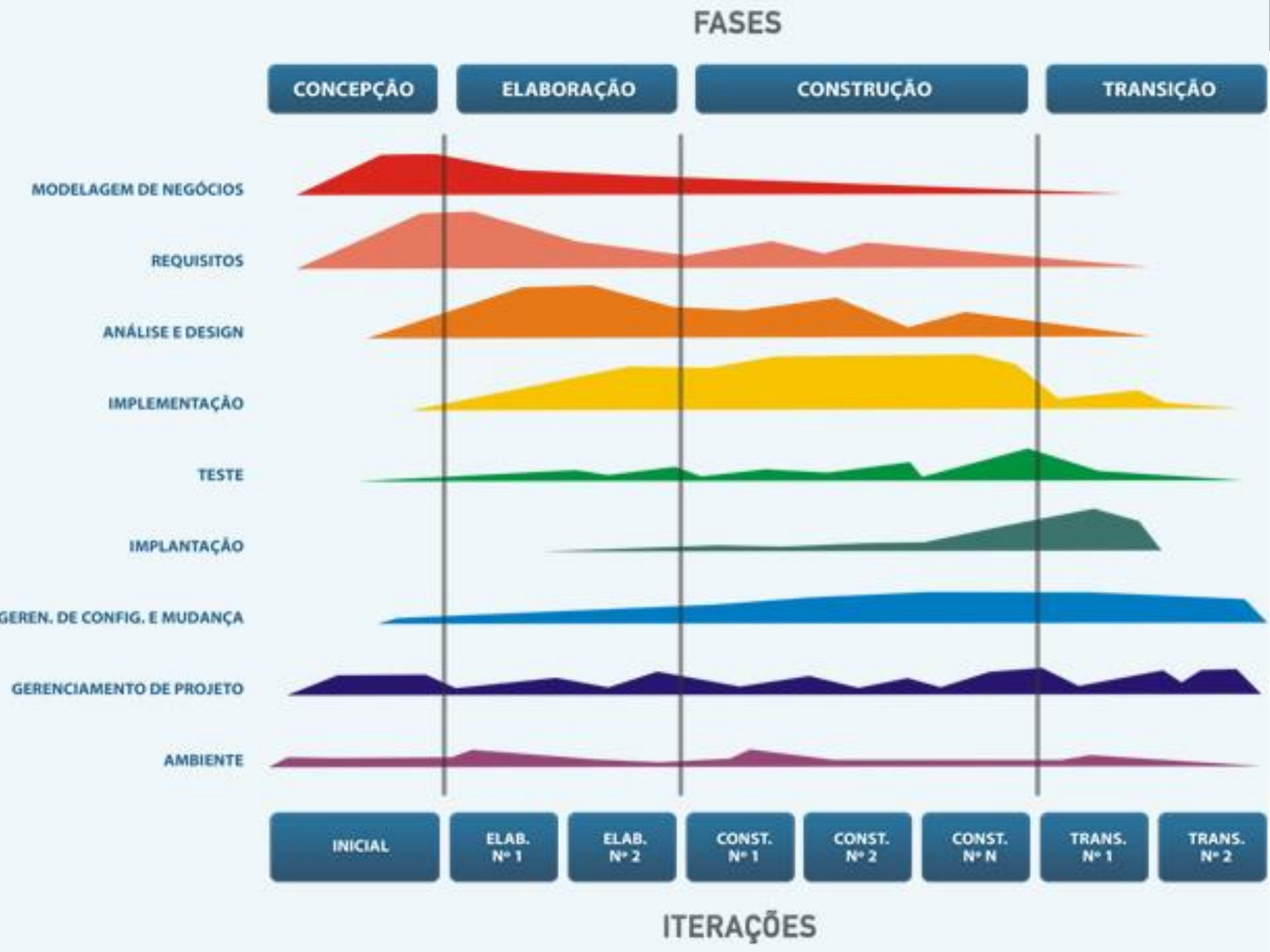
- **Modelo Espiral**



# Metodologia de Desenvolvimento

- Chaos Report (2017): relação entre metodologia e resultado

| SIZE | METHOD | SUCCESSFUL | CHALLENGED | FAILED |
|---|---|---|---|---|
| All Size Projects | Agile | 39% | 52% | 9% |
| | Waterfall | 11% | 60% | 29% |
| | | | | |
| Large Size Projects | Agile | 18% | 59% | 23% |
| | Waterfall | 3% | 55% | 42% |
| Medium Size Projects | Agile | 27% | 62% | 11% |
| | Waterfall | 7% | 68% | 25% |
| Small Size Projects | Agile | 58% | 38% | 4% |
| | Waterfall | 44% | 45% | 11% |


FASES
CONCEPÇÃO
ELABORAÇÃO
CONSTRUÇÃO
TRANSIÇÃO
MODELAGEM DE NEGÓCIOS
REQUISITOS
ANÁLISE E DESIGN
IMPLEMENTAÇÃO
TESTE
IMPLANTAÇÃO
GEREN. DE CONFIG. E MUDANÇA
GERENCIAMENTO DE PROJETO
AMBIENTE
INICIAL
ELAB. Nº 1
ELAB. Nº 2
CONST. Nº 1
CONST. Nº 2
CONST. Nº N
TRANS. Nº 1
TRANS. Nº 2
ITERAÇÕES

# Desenvolvimento de Software

**Levantamento de Requisitos**

- Compreensão do problema
- Ideia do usuário = Ideia dos desenvolvedores (ver próximo slide)
- Desenvolvedores e usuários discutem as necessidades dos futuros usuários do sistema a ser desenvolvido
- Necessidades = Requisitos

- “Requisito é uma condição ou capacidade que deve ser alcançada ou possuída por um sistema ou componente deste, para satisfazer um contrato, padrão, especificação ou outros documentos formalmente impostos.” (Maciaszek, 2000)

# Desenvolvimento de Software


Como o cliente explicou


Como o lider de projeto entendeu

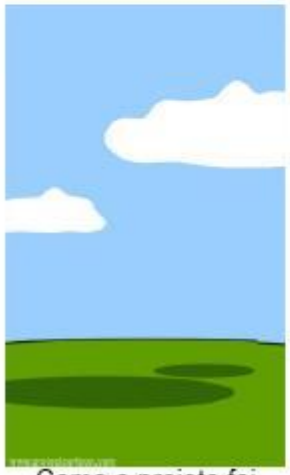

Como o analista planejou


Como o programador codificou

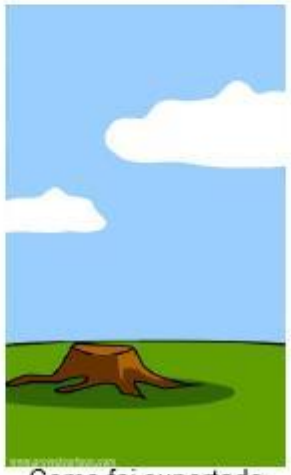

O que os beta testers receberam


Como o consultor de negocios descreveu


Valor que o cliente pagou


Como o projeto foi documentado


O que a assistencia tecnica instalou


Como foi suportado


Quando foi entregue


O que o cliente realmente necessitava

# Desenvolvimento de Software

**Levantamento de Requisitos**

- Domínio do problema ou Domínio do negócio:
  - Parte do mundo real que é relevante para o sistema
  - Quais informações e processos precisam estar no sistema?

- Tipos de Requisitos:
  - Requisitos Funcionais
  - Requisitos Não-Funcionais
  - Requisitos Normativos

# Desenvolvimento de Software

**Levantamento de Requisitos**

- Requisitos Funcionais
  - Definem funcionalidade do sistema
  - E.g. “O sistema deve permitir que cada professor realize o lançamento de notas das turmas nas quais lecionou.”
  - E.g. “Os coordenadores de escola devem poder obter o número de aprovações, reprovações e trancamentos em cada disciplina oferecida em um determinado período.”

# Desenvolvimento de Software

**Levantamento de Requisitos**

- Requisitos Não-Funcionais
  - Definem características de qualidade que o sistema deve possuir
  - Podem estar associadas a funcionalidades
  - Subtipos de requisitos não-funcionais são:
    - Confiabilidade: medidas quantitativas de confiabilidade; e.g. tempo médio entre falhas, recuperação de falhas
    - Desempenho: definem tempos de resposta esperados para determinadas funcionalidades
    - Portabilidade: restrições ou exigências sobre hardware e/ou software
    - Segurança: propriedades de segurança (restrições de senha ou acesso)
    - Usabilidade: relacionados ao uso do software

# Desenvolvimento de Software

**Levantamento de Requisitos**

- Requisitos Normativos
  - Declaração de restrições impostas sobre o desenvolvimento do sistema, que não se enquadram como requisitos não-funcionais
  - E.g. Adequações a custos e prazos;
    componentes de hardware e software a serem adquiridos;
    comunicação com outro sistema;
    restrições de funcionamento da instituição;
    restrições de valores;

# Desenvolvimento de Software

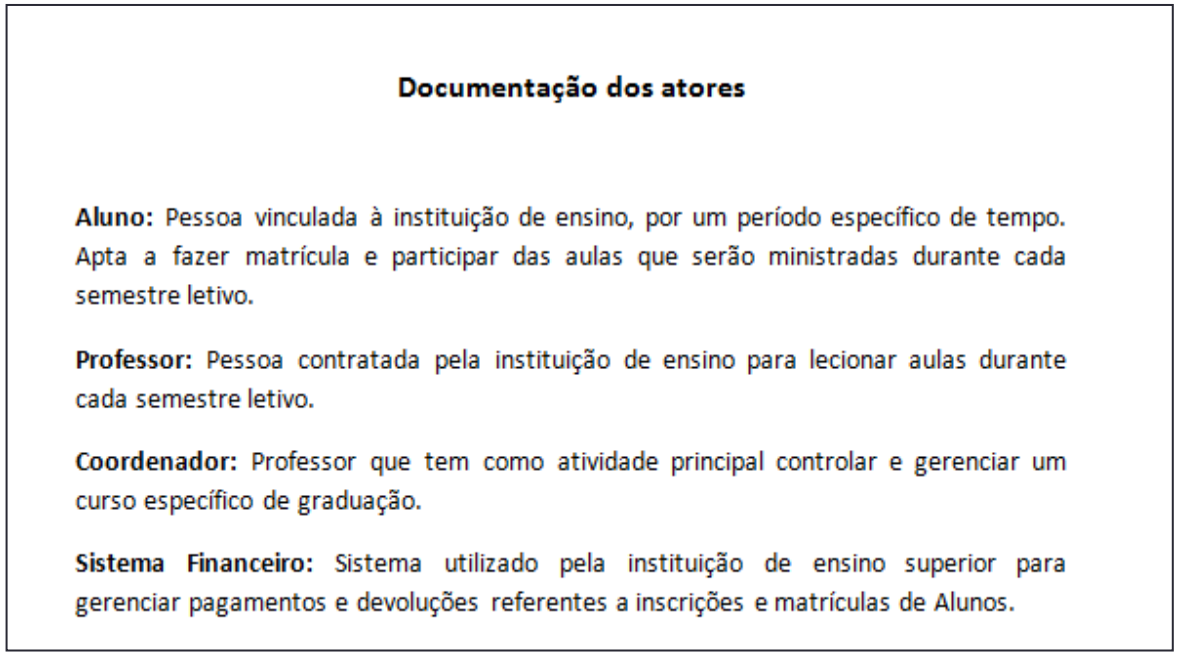
## Levantamento de Requisitos

- Produto resultante: Documento de Requisitos
  - Escrita simples; leitura fácil para técnicos e não-técnicos
  - **Não** contém informações sobre as soluções técnicas
  - É um termo de **consenso** entre equipe técnica e cliente(s)
  - É comum que os requisitos não sejam estáticos:
    - Surgimento de novas necessidades
    - Expectativas dos usuários - ao usar o software, os cliente descobrem requisitos que não tinham pensado.
  - Muitos sistemas são abandonados ou tem um custo maior devido a utilização de pouco tempo para esta etapa

# Desenvolvimento de Software

## Análise de Requisitos

- Uma das primeiras etapas da análise de requisitos é identificar os “atores” => pessoas, dispositivos ou software que irá interagir com o sistema



**Documentação dos atores**

**Aluno:** Pessoa vinculada à instituição de ensino, por um período específico de tempo. Apta a fazer matrícula e participar das aulas que serão ministradas durante cada semestre letivo.

**Professor:** Pessoa contratada pela instituição de ensino para lecionar aulas durante cada semestre letivo.

**Coordenador:** Professor que tem como atividade principal controlar e gerenciar um curso específico de graduação.

**Sistema Financeiro:** Sistema utilizado pela instituição de ensino superior para gerenciar pagamentos e devoluções referentes a inscrições e matrículas de Alunos.

# Desenvolvimento de Software

## Análise de Requisitos

- Deve-se também registrar as “regras de negócio”, com:
  - **Nome e identificador:** um nome e identificador para facilitar a referência e a busca por regras de negócio
  - **Descrição:** a descrição textual da regra de negócio

**Documentação das Regras de Negócio**

| | **Quantidade de inscrições possíveis** | **[RN-01]** |
|---|---|---|
| Descrição: | Um aluno não pode se inscrever em mais de seis disciplinas por semestre letivo. | |
| Fonte: | Coordenador da escola de informática | |
| Histórico: | Data de identificação: | 12/07/2002 |
| | Data de atualização: | - |

# Exemplo

Levantar os requisitos necessários para um sistema acadêmico que permite o controle e gerenciamento de matricula, frequência e desempenho dos discentes e a organização das disciplinas ofertadas. O sistema acadêmico deverá permitir que os acadêmicos realizem suas matrículas nas turmas de disciplinas disponíveis, considerando restrições de pré-requisitos, número máximo de créditos (9) e limite de alunos por turma. Deverá permitir que chefes de departamento incluam novas disciplinas e novos professores, abram novas turmas para as disciplinas existentes com sala, horário, lotação máxima e professor definidos. As disciplinas só poderão ser ofertadas entre 7:30 e 12:00, e, 13:30 e 21:40, em blocos de 50 minutos por aula (hora-aula). Também deverá ser possível que professores acessem suas turmas e registrem frequência e notas para seus alunos.

# Exemplo

O sistema deverá ter uma opção para finalizar o semestre, possibilitando a inclusão das notas de exame. Um aluno deverá ter frequência superior a 75% e deverá ter uma média superior a 3 para realizar exame. Caso sua nota seja maior ou igual a 7 está aprovado (desde que tenha a frequência necessária). Após a digitação das notas de exame o professor deverá finalizar a turma e o sistema mostrará o resultado final. O sistema deverá funcionar nos sistemas operacionais Windows e Linux e deverá ter seu acesso controlado por login e senha.

# Atividade

Agora é a sua vez!

Levante os requisitos necessários para o sistema descrito no documento que está disponível na página da disciplina.

Documente também os atores e as regras de negócio do sistema descrito.

# Bibliografia


- **Básica:**

  BEZERRA, E. Princípios de Análise e Projetos de Sistemas com UML. Rio de Janeiro: Campus, 2003.

  PRESSMAN, R.S. Engenharia de Software. São Paulo: Makron Books, 2002.

  SOMMERVILLE, I. Engenharia de Software. São Paulo: Addison Wesley, 2003.

- **Complementar:**

  WARNIER, J. Lógica de Construção de Programas. Rio de Janeiro: Campus, 1984.

  JACKSON, M. Princípios de Projeto de Programas. Rio de Janeiro: Campus, 1988.

  PAGE-JONES, M. Projeto Estruturado de Sistemas. São Paulo: McGraw-Hill, 1988.