

Universidade Federal da Paraíba – Campus I  
Centro de Informática  
Departamento de Sistemas e Computação

# Métodos e Projeto de Software

## Material 2: Revisão:

### Processo de Desenvolvimento

### Engenharia de Requisitos

Prof. Raoni Kulesza  
[raoni@ci.ufpb.br](mailto:raoni@ci.ufpb.br)



# Motivação

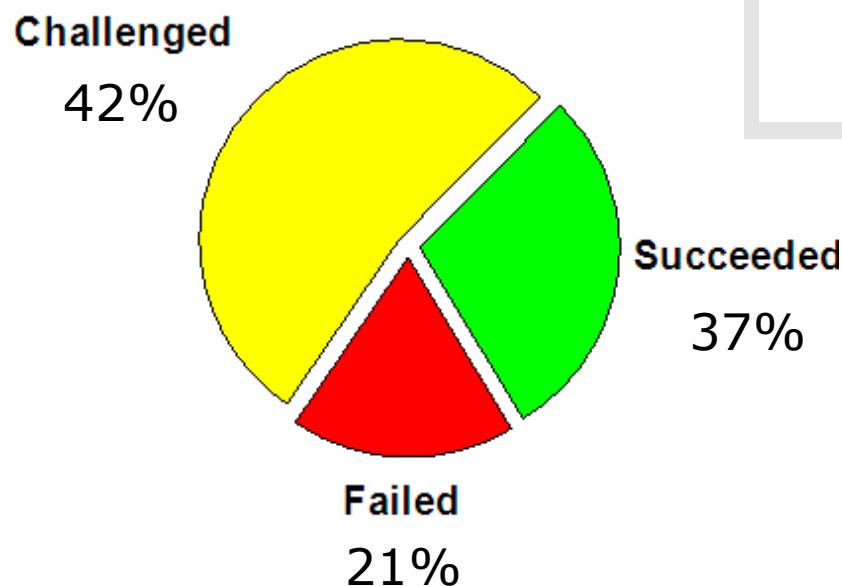
- Cenário Atual de desenvolvimento de software:
  - Dificuldade de entendimento dos requisitos
  - Aprendizado constante de novas tecnologias
  - Tempo curto disponível para desenvolvimento
  - Equipe médias / grandes
  - Necessidade de sistemas mais confiáveis e de qualidade
  - Sistemas mais complexos
  - Recursos limitados



# CHAOS

- Pesquisa do Standish Group (2011) em empresas de software dos EUA

## CHAOS 2011



The project is **completed** on time and on budget, with all features and functions originally specified.

The project is **completed** and **operational**, but over budget, late, and with fewer features and functions than initially specified.

The project is **canceled** before completion, or never implemented.



# CHAOS

- Redefinição de Successful:
- "*Standish admits "on target" was never a good measure because it lacked any measure of customer outcome. They even note there are many projects that met the triple constraints, but left the customer unsatisfied. So they replaced that measure with a measure of customer perceived value. This resulted in a 7% decrease in the rate of successful projects"*
- "*You read that right. If we start measuring ourselves the way our customers measure us, we're doing at least 7% worse than we think we are. This should be the final nail in the coffin of traditional scope/schedule/budget project management*"



# CHAOS

## MODERN RESOLUTION FOR ALL PROJECTS

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

The Modern Resolution (OnTime, OnBudget, with a satisfactory result) of all software projects from FY2011–2015 within the new CHAOS database. Please note that for the rest of this report CHAOS Resolution will refer to the Modern Resolution definition not the Traditional Resolution definition.



# CHAOS

## CHAOS RESOLUTION BY PROJECT SIZE

	SUCCESSFUL	CHALLENGED	FAILED
Grand	2%	7%	17%
Large	6%	17%	24%
Medium	9%	26%	31%
Moderate	21%	32%	17%
Small	62%	16%	11%
<b>TOTAL</b>	<b>100%</b>	<b>100%</b>	<b>100%</b>

*The resolution of all software projects by size from FY2011–2015 within the new CHAOS database.*



# *Engenharia de Software: Áreas*

- Segundo o SWEBOK:
  - Requisitos de software
  - Projeto de software
  - Construção de software
  - Teste de software
  - Manutenção de software
  - Gerência de configuração de software
  - Gerência de engenharia de software
  - Processos de Engenharia de Software
  - Ferramentas e Métodos de Engenharia de Software
  - Qualidade de software



# *Engenharia de Software: Definição*

“*Engenharia de Software (ES) é uma tecnologia em 3 camadas: **processos, métodos e ferramentas**. E a base de todas essas camadas é o foco na **qualidade do software** desenvolvido*”  
[PRESSMAN, 96].



# *Engenharia de Software: Definição*

*Relaciona-se com métodos, ferramentas e técnicas para **desenvolver** e **gerenciar** o processo de **criar** e **evoluir** produtos de software” [Sommerville-95]*

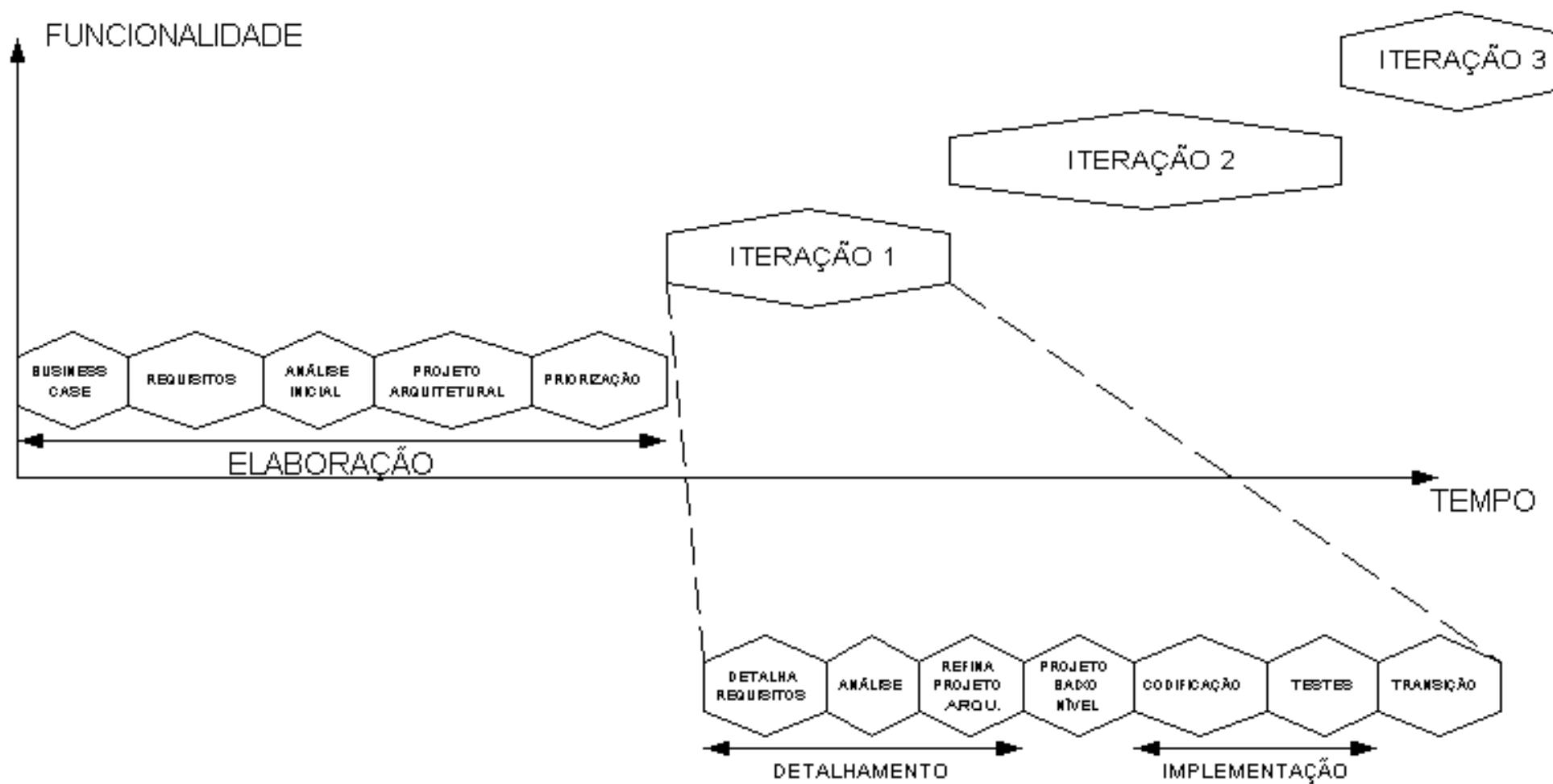


# *Engenharia de Software - Objetivos*

- Objetivos Gerais
  - Desenvolver produtos de software de *qualidade*
    - Produto que atende às necessidades dos clientes
    - Produto entregue dentro do custo estimado e no prazo previsto
  - Organizar o processo de desenvolvimento fazendo com que o mesmo seja mais produtivo e o software gerado tenha mais *qualidade*
    - Produto fácil de manter e evoluir
- Processo de Desenvolvimento  
Determina o quê é feito, como é feito, quando e por quem?



# Engenharia de Software



# *Processo de Desenvolvimento*

- Compreende as atividades necessárias para definir, desenvolver, testar e manter um produto (sistema de software).
- Alguns objetivos de um PDSW
  - Definir quais as atividades a serem executadas ao longo do projeto;
  - Quando, como e por quem tais atividades serão executadas;
  - Prover pontos de controle para verificar o andamento do projeto;
  - Padronizar a forma de desenvolver software em uma organização.



# *Atividades Típicas de PDSW*



Idéias

Levantamento  
de Requisitos

Análise de  
Requisitos

Projeto de  
Software

Implementação  
e Testes

Implantação

Produto

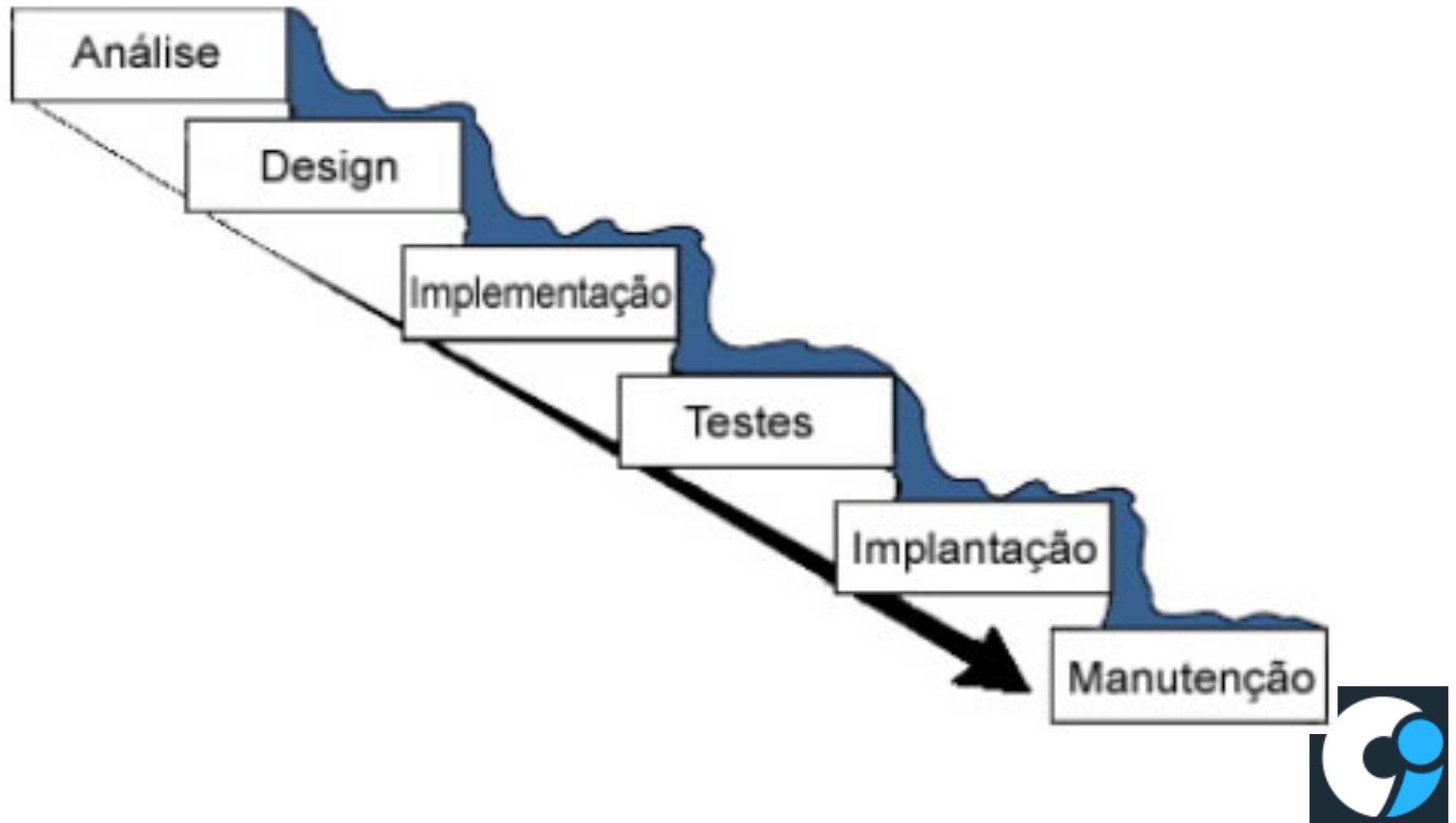


# *Metodologias Tradicionais (Passado)*

- Modelo Cascata(*Waterfall*)
- Modelo antigo e mais utilizado
- Foi desenvolvido com base no ciclo da engenharia tradicional
- É caracterizado por uma abordagem seqüencial para o desenvolvimento do software
- Cada atividade é uma fase distinta. Só após o seu total término é que a próxima atividade começa.



# *Desenvolvimento em Cascata*

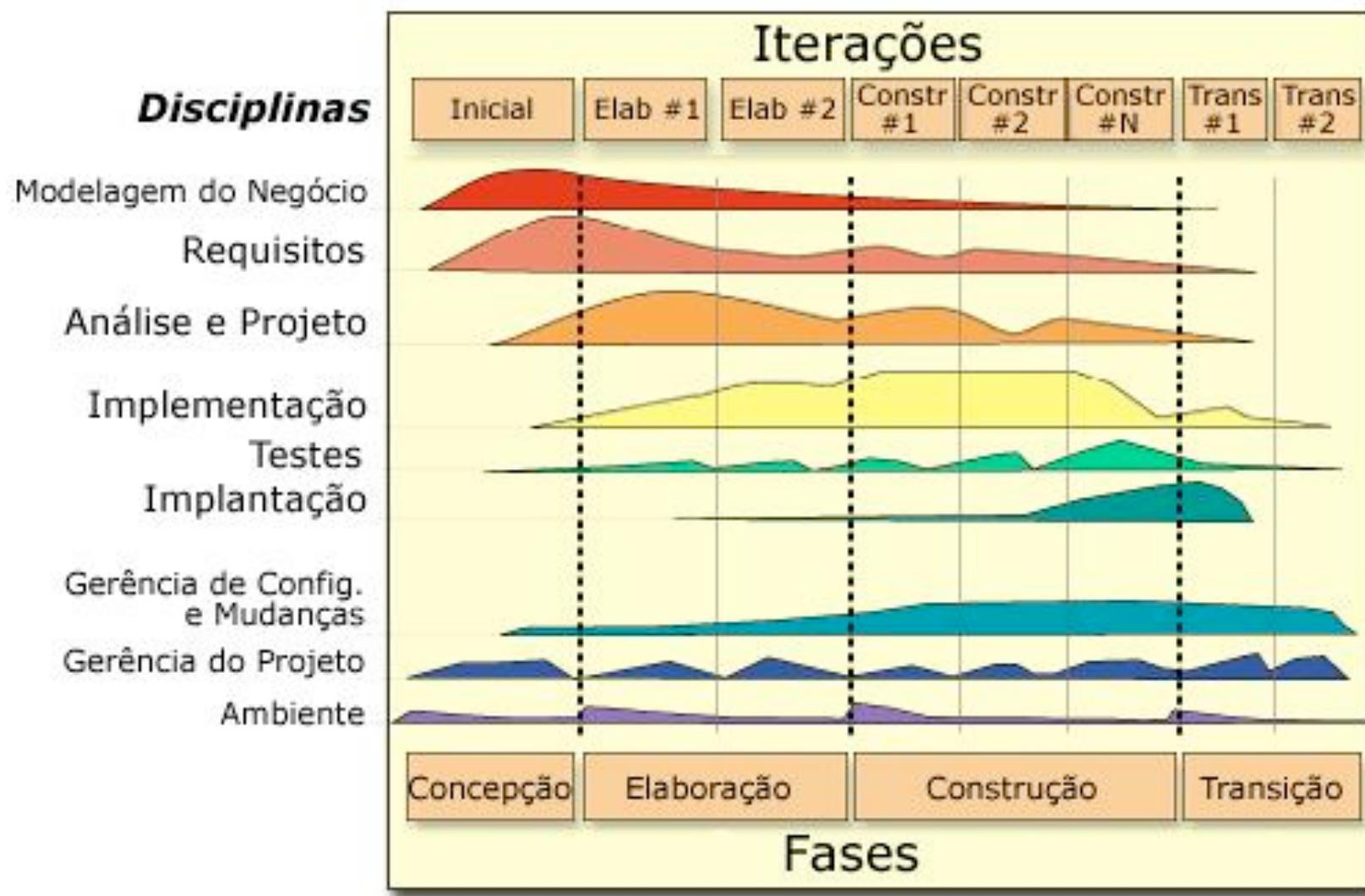


## *Metodologias Tradicionais – Até um tempo atrás*

- Demandam um certo grau de formalismo
- São em sua maioria processos **prescritivos**
- Focam em Planejamento
- Construção através de refinamentos sucessivos
- A mais conhecida: RUP (Rational Unified Process)



# RUP



# RUP



# *Pontos Chave*

- Quantidade de Formalismo:  
“O Peso” da Metodologia
- Foco no Planejamento
- Muitos modelos de processo que não são em cascata (por exemplo: RUP), são adotados de forma semelhante ao modelo em cascata.



# *Metodologia Ágil*

- Metodologia de Desenvolvimento Ágil de software (*Agile software development*) é um **conjunto de métodos de desenvolvimento de software**.
- O desenvolvimento ágil, tal como qualquer metodologia de software, providencia uma **estrutura conceitual para reger projetos de engenharia de software**.



# *Metodologia Ágil*

- A maioria dos métodos ágeis tenta minimizar o risco pelo desenvolvimento do software em curtos períodos (iteração)
- Entre 1 e até 4 semanas (no máximo 4 meses)
- Iteração = processo de software em miniatura
- Etapas: Planejamento, Análise de Requisitos, Projeto, Codificação, Teste e Documentação.



# *Metodologia Ágil*

- Processo convencional, cada iteração **não está necessariamente focada em adicionar um novo conjunto significativo de funcionalidades**
- Projeto de software ágil busca a **capacidade de implantar uma nova versão do software ao fim de cada iteração.**
- Desenvolvimento Ágil enfatiza a comunicação face-a-face, por isso **produzem pouca documentação** em relação a outros métodos, sendo este um de seus pontos diferenciais

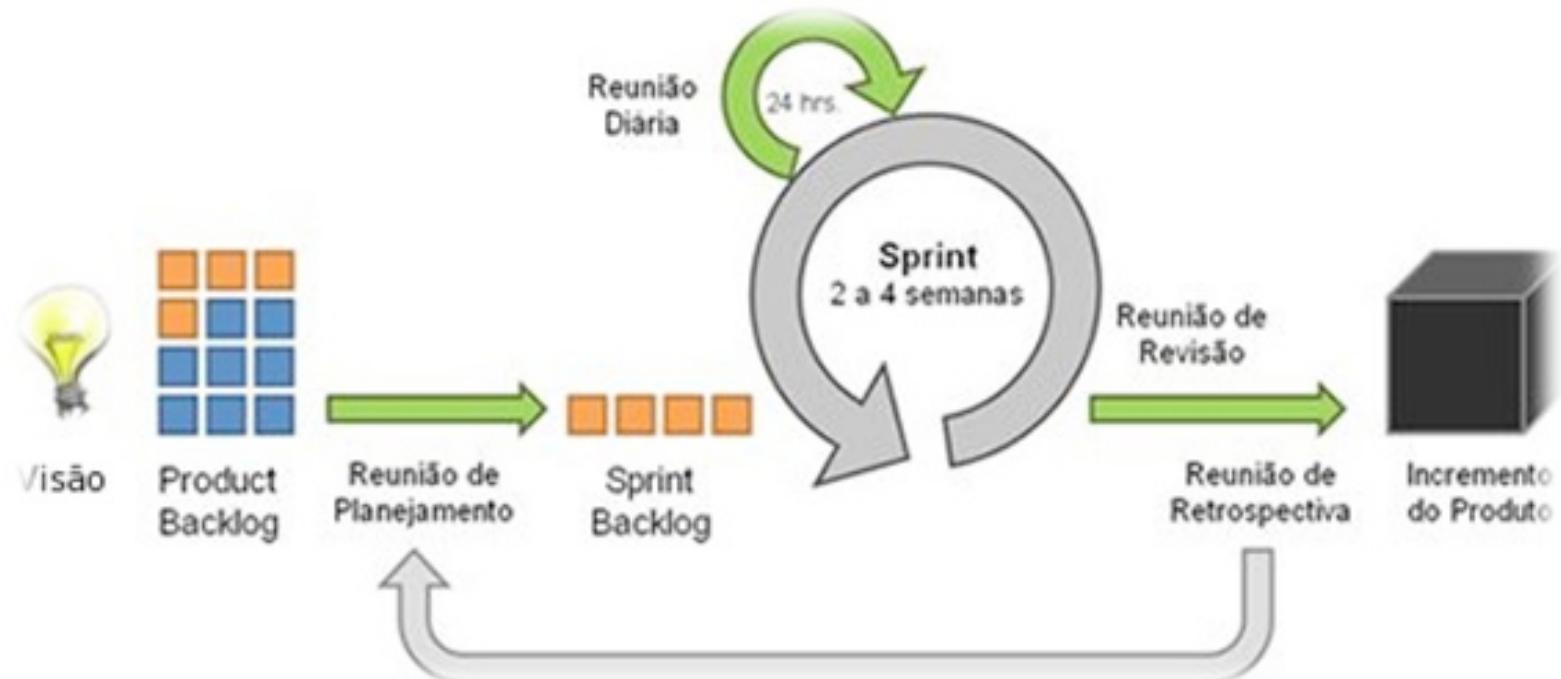


# *Metodologia Ágil*

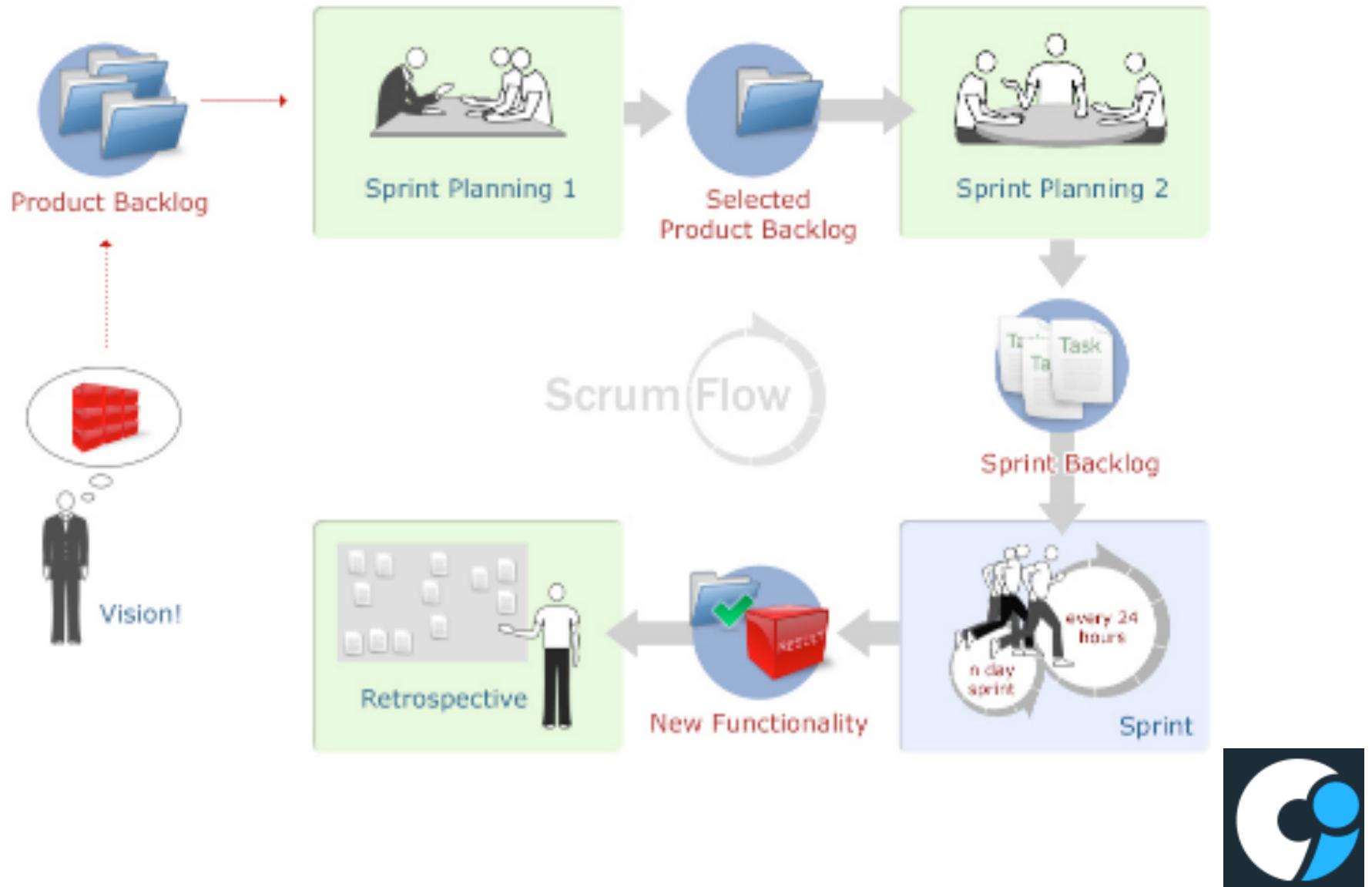
- Eliminam grande parte do excesso de modelos e de documentação e o tempo gasto nestas tarefas;
- Enfatizam um desenvolvimento de aplicação simples e iterativo;
- Exemplos: *Extreme Programming* (1996), DSDM (Método de Desenvolvimento de Sistemas Dinâmicos) (1995), *Crystal* (2004) e o *Scrum* (1986).



# Scrum



# Scrum



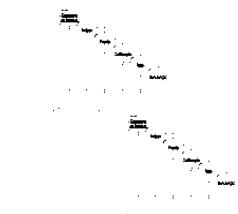
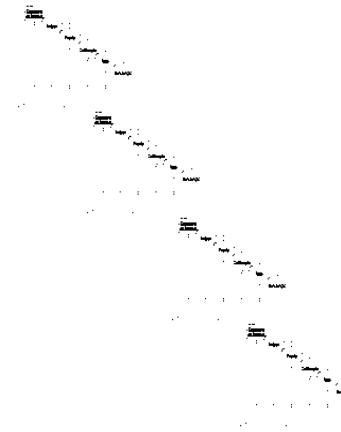
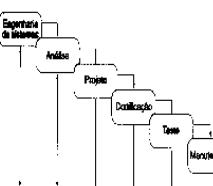
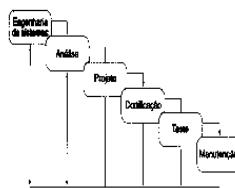
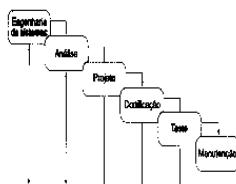
# *Diferentes Modelos (Cascata, RUP, Ágil)*

Tempo

Clássico

Iterativo

Ágil



# *Modelos de Processo - Discussão*

*“Processo de software são **complexos** e, como todos os processos intelectuais e criativos, dependem de **julgamento humano**”*

*“**Não existe um processo ideal**, e várias organizações desenvolveram abordagens inteiramente **diferentes**”*

*“Os processos **evoluíram** para explorar as capacidades das **pessoas** em uma organização e as características **específicas** dos sistemas que estão sendo desenvolvidos”*

*(Sommerville, capítulo 4 – p.42-43 )*



# *Modelos de Processo - Conclusão*

- O CHAOS 2015 tem uma visão diferente:
- “*Across all sized projects, agile projects are 350% more likely to be successful. This difference is minimal when running small projects - 32%. But at the huge project end of the spectrum, agile projects are 600% more likely to be successful. This is really strange, considering the intense focus on "scaling" that exists in the agile industry. Indeed, it's waterfall that sucks at scaling".*
- “The overall results clearly show that waterfall projects do not scale well, while agile projects scale much better.”

**Run small and agile projects. Train your agile team**

[2015 CHAOS Report, Standish Group]



*Dúvidas? Obrigado.*  
*raoni@ci.ufpb.br*



# *Engenharia de Requisitos: Introdução*



# *Atividades Típicas*



Idéias

Levantamento  
de Requisitos

Análise de  
Requisitos

Projeto de  
Software

Implementação  
e Testes

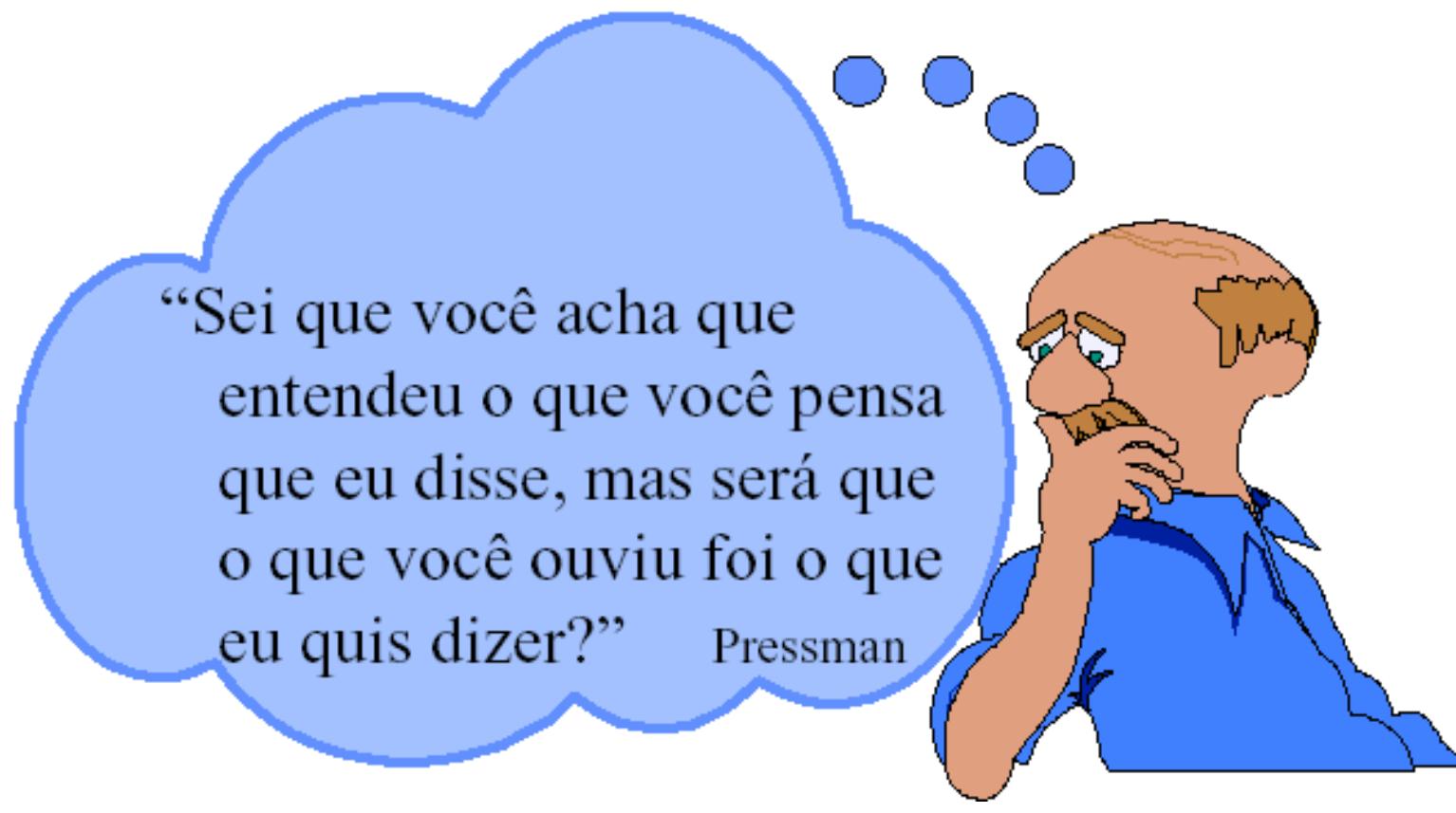
Implantação

Produto



# *Requisitos: Motivação*

- Levantamento de Requisitos
  - Um cenário comum em desenvolvimento de SW



# *Importancia de Engenharia de Requisitos*



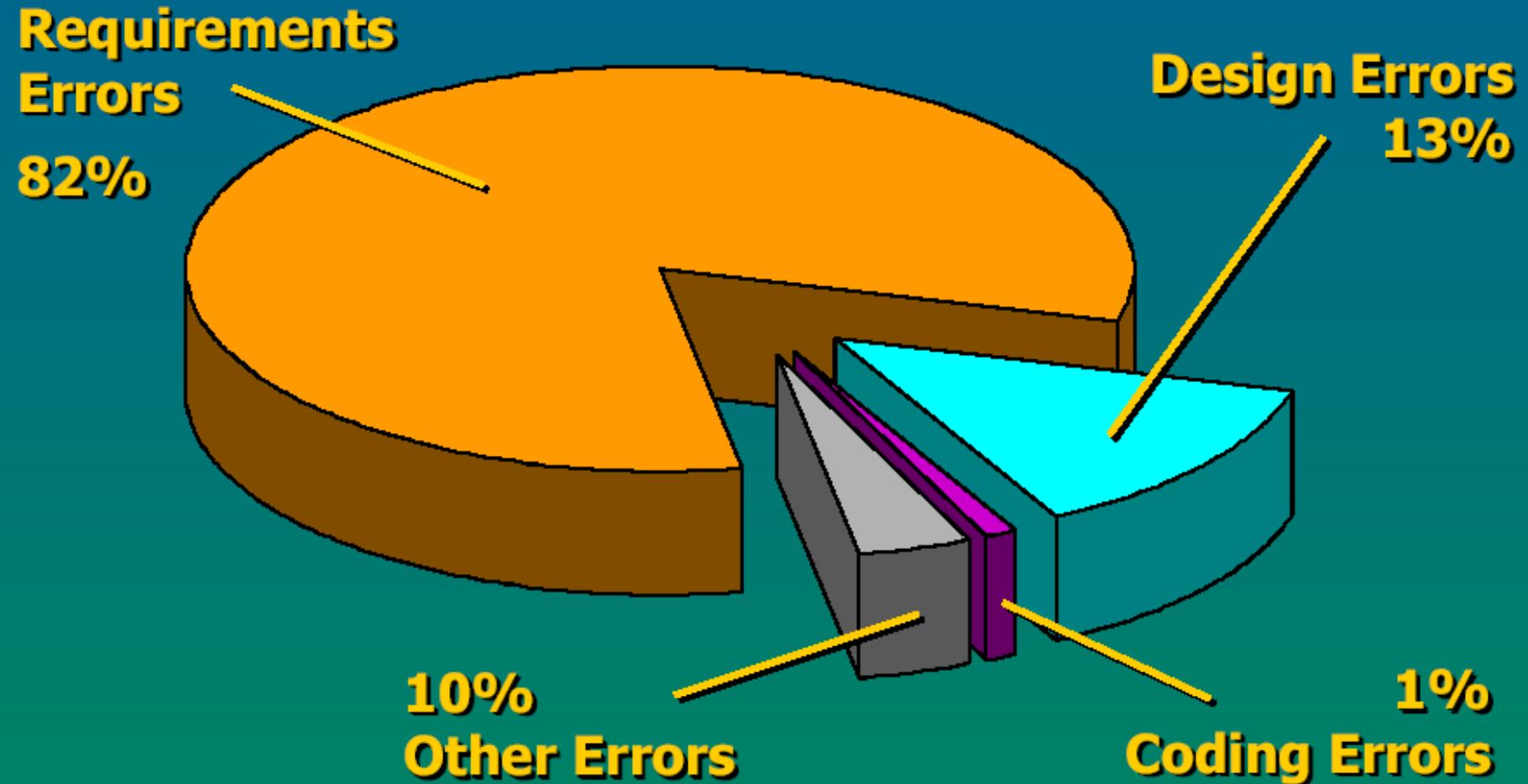
Mesmo um pequeno erro na fase de requisitos  
pode ocasionar um grande problema na solução



# Typical Development Process



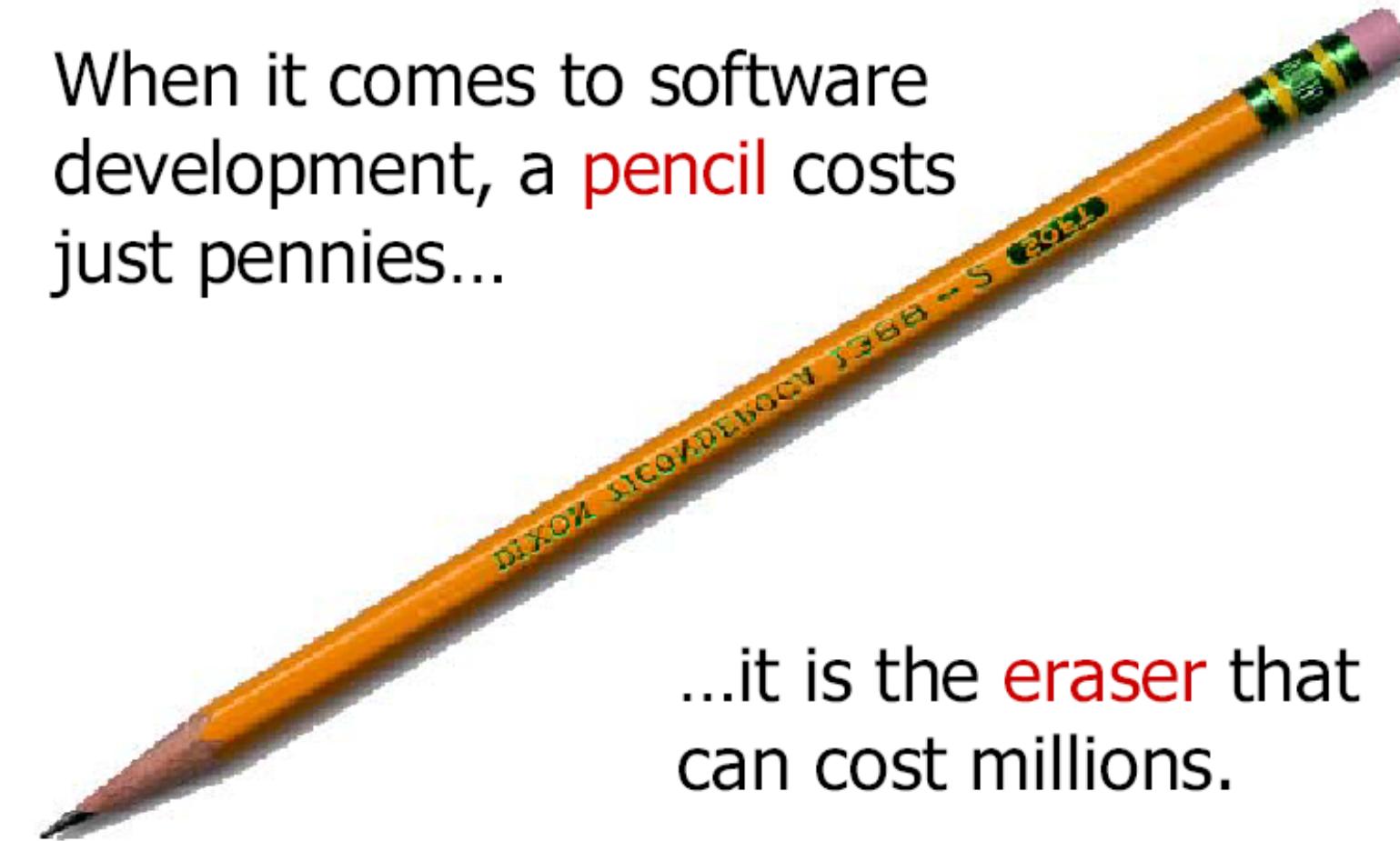
# The Cost to Fix Software Defects



Source: "An Information Systems Manifesto"  
James Martin

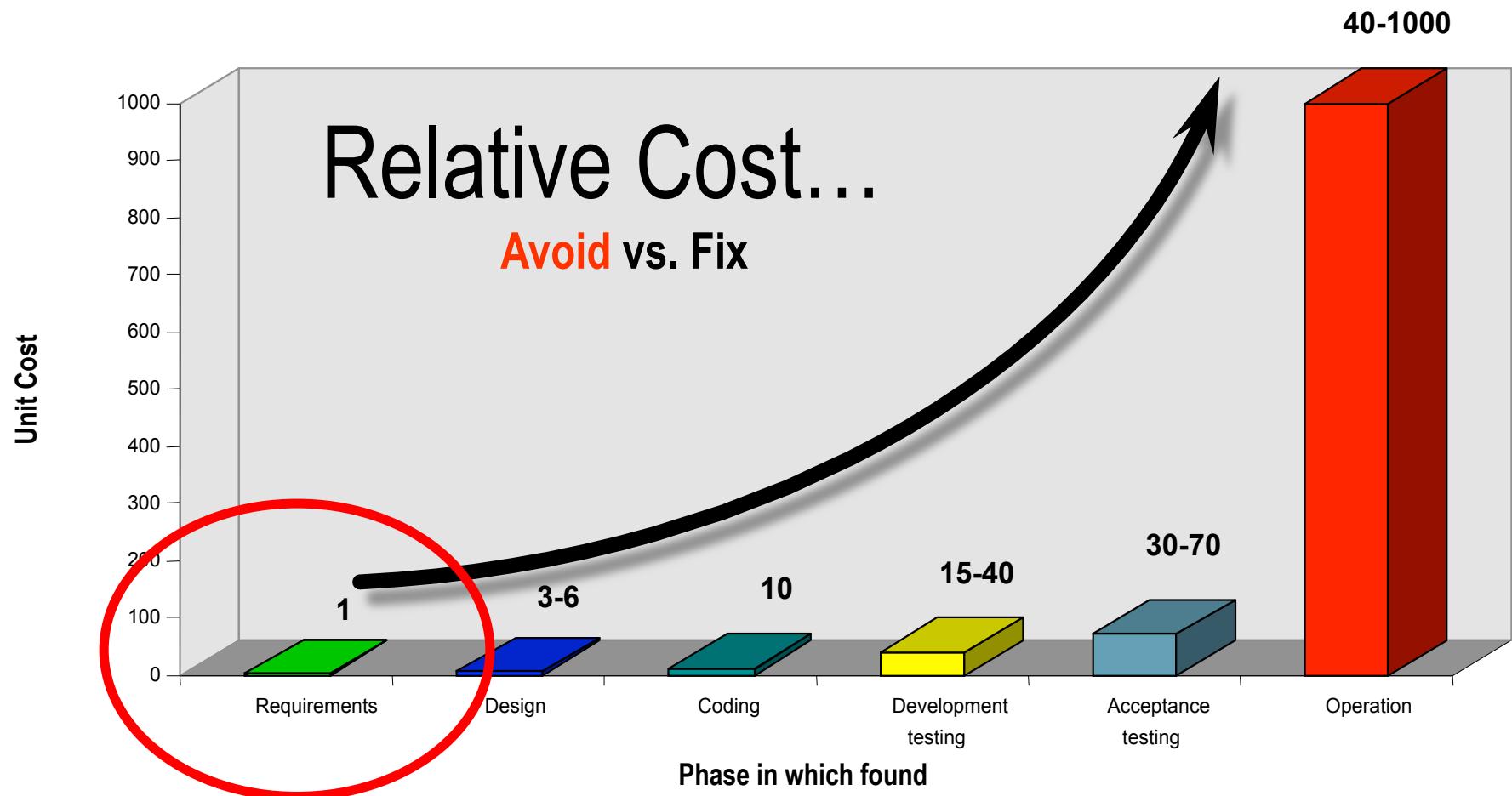
# Recognizing Where the \$\$\$ Goes

When it comes to software development, a **pencil** costs just pennies...



...it is the **eraser** that can cost millions.

# Correção x \$\$\$



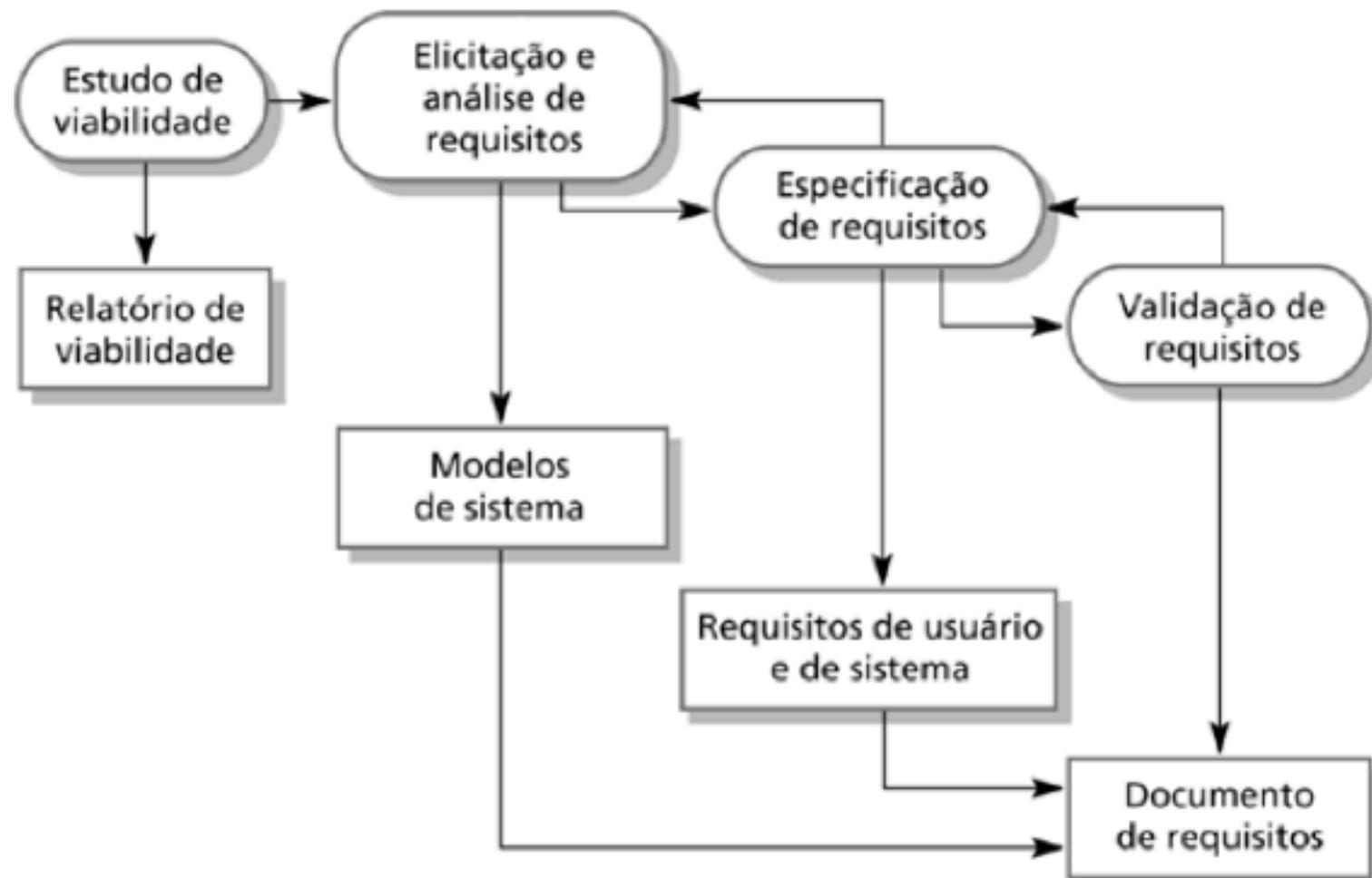
- Gause, Donald and Gerald Weinberg. Exploring Requirements: Quality Before Design

# *Engenharia de Requisitos*

- Processo que engloba todas as atividades que contribuem para a produção de um documento de requisitos e sua **manutenção** ao longo do tempo.
- O processo pode ser bem **diferente** dependendo do domínio da aplicação, das pessoas envolvidas e organização envolvida
- Porém, normalmente contém 4 atividades genéricas:
  - **Elicitação**
  - **Análise**
  - **Validação**
  - **Gerenciamento**
- Normal essas atividade são precedidas por um **estudo de viabilidade**



# *O Processo de Engenharia de Requisitos*



Modelagem de Processos de Negócio (BPMN)



*Dúvidas? Obrigado.*  
*raoni@ci.ufpb.br*

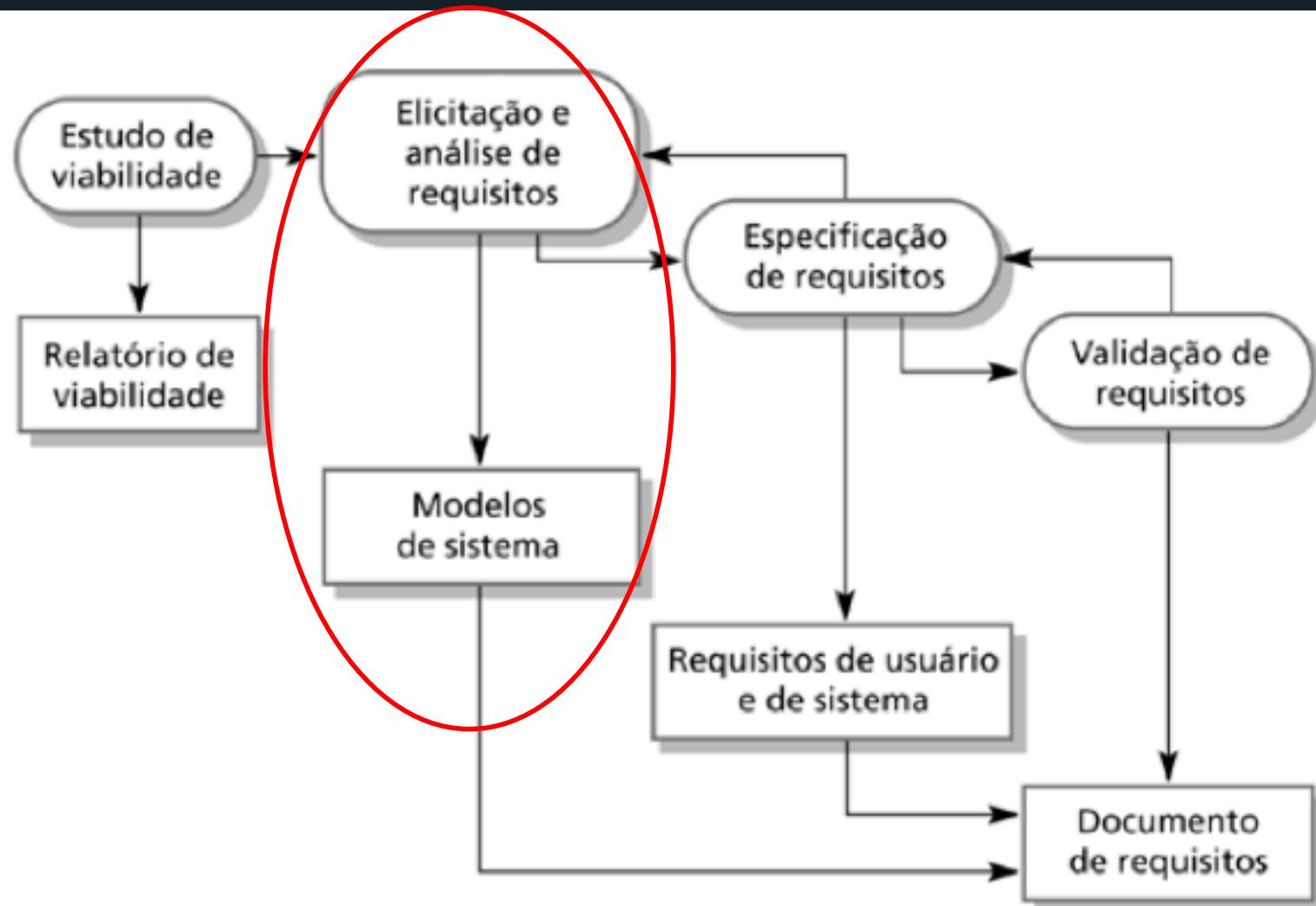


# *Estudo de viabilidade*

- Permite verificar se um projeto é viável a partir de restrições definidas pelo contexto em que está ou será inserido do ponto de vista **tecnológico e organizacional**
- Exemplos de questões a tratar:
  - Sistema vai contribuir para as metas da empresa
  - Sistema é compatível com as restrições organizacionais (econômicas, políticas, ambientais, recursos disponíveis) , tecnológicas e temporais
  - Sistema pode ser integrado com outros sistemas atuais em uso
- A identificação de requisitos só inicia após aprovação do estudo de viabilidade



# *O Processo de Engenharia de Requisitos*



# *Elicitação dos requisitos*

Princípios  
Atividades  
Técnicas  
Artefato Produzido



# *Princípios*

- O desenvolvimento de um sistema é motivado por um problema.
- O objetivo da elicição de requisitos é entender o problema claramente.
- O problema pode ser entendido através da descoberta de quem ou do que é afetado pelo problema.
- Várias atividades precisam ser realizadas:
  - Analisar o problema
  - Identificar as fontes de requisitos
  - Elicitar os requisitos a partir destas fontes



# *Atividade N.o 1: Analisar o problema*

- Identificar metas (motivação do projeto) :
  - e.x. Aumentar as vendas da empresa
- Identificar restrições na solução:
  - e.x. restrições nos recursos (tempo, pessoas, orçamento)
- Definir o escopo do problema (os limites do sistema)
- O que é externo ao problema (o ambiente)
- O que é interno ao problema (o núcleo do sistema)
- Avaliar os riscos (e.x. financeiro, técnico, etc).
- Estimar um custo aproximado do projeto.



# *Atividade N.o 2:Identificar fontes de requisitos*

- Stakeholders:
  - Qualquer um que seja afetado pelo sistema ou que influencie o sistema
- Representa papéis tais como um trabalho que alguém faz, um tipo de usuário, ou uma área operacional tal como um departamento
- Elicitar requisitos a partir de papéis representativos implica em obter vários pontos de vista diferentes que ajudam na completude e consistência
- O domínio: ambiente operacional, ambiente organizacional, domínio da aplicação



# *Ambiente Operacional*



# *Ambiente Organizacional*

- Informações típicas que você deve descobrir:
  - Quem usará o sistema e por que?
  - Quão organizado é este ambiente? Nossa solução pode mudá-lo?
  - A organização tem requisitos sobre a qualidade do sistema ou do processo?



# *Atividade N.o 3: Elicitar os requisitos*

- Dois objetivos principais:
  - Encontrar o que os stakeholders "precisam"
  - Coletar informação sobre o que é viável para os stakeholders possuírem.
    - Nem todos os requisitos dos stakeholders serão realísticos.
    - Nem toda informação será útil.
      - Mas neste estágio, nós apenas as coletamos!



# *Técnicas para Elicitação de Requisitos*

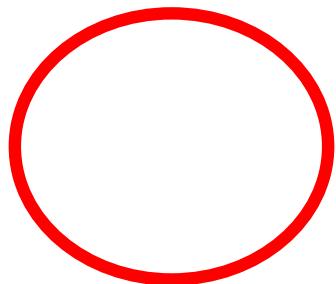
- Algumas técnicas incluem
  - Entrevistas
  - Questionários
  - Reunião Facilitada
  - Observações
    - Análise de Artefatos Existentes
    - Análise de Protocolos
    - Etnografia
  - Cenários
  - Prototipagem



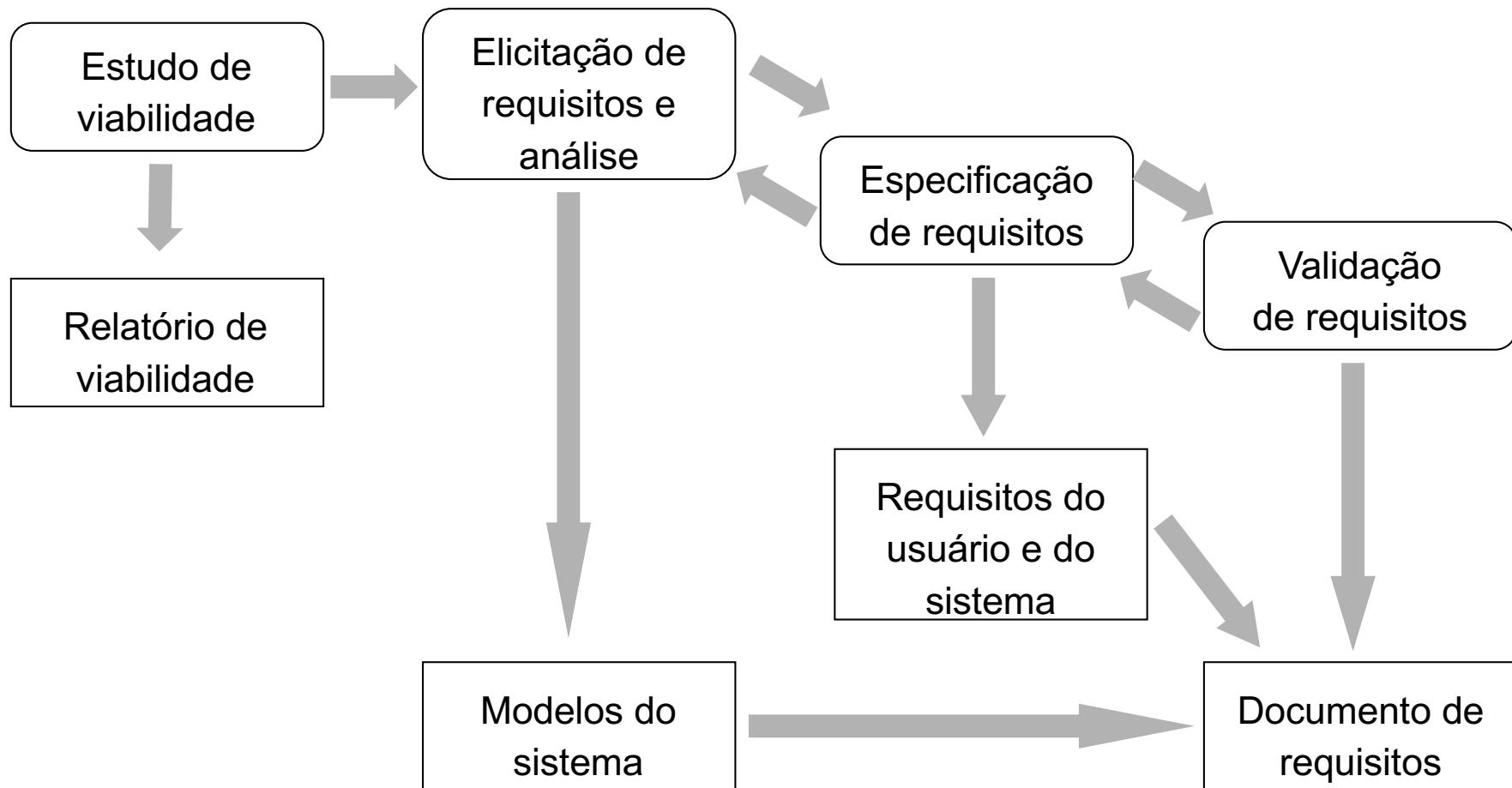
# *Papéis dos Atores e Artefatos*



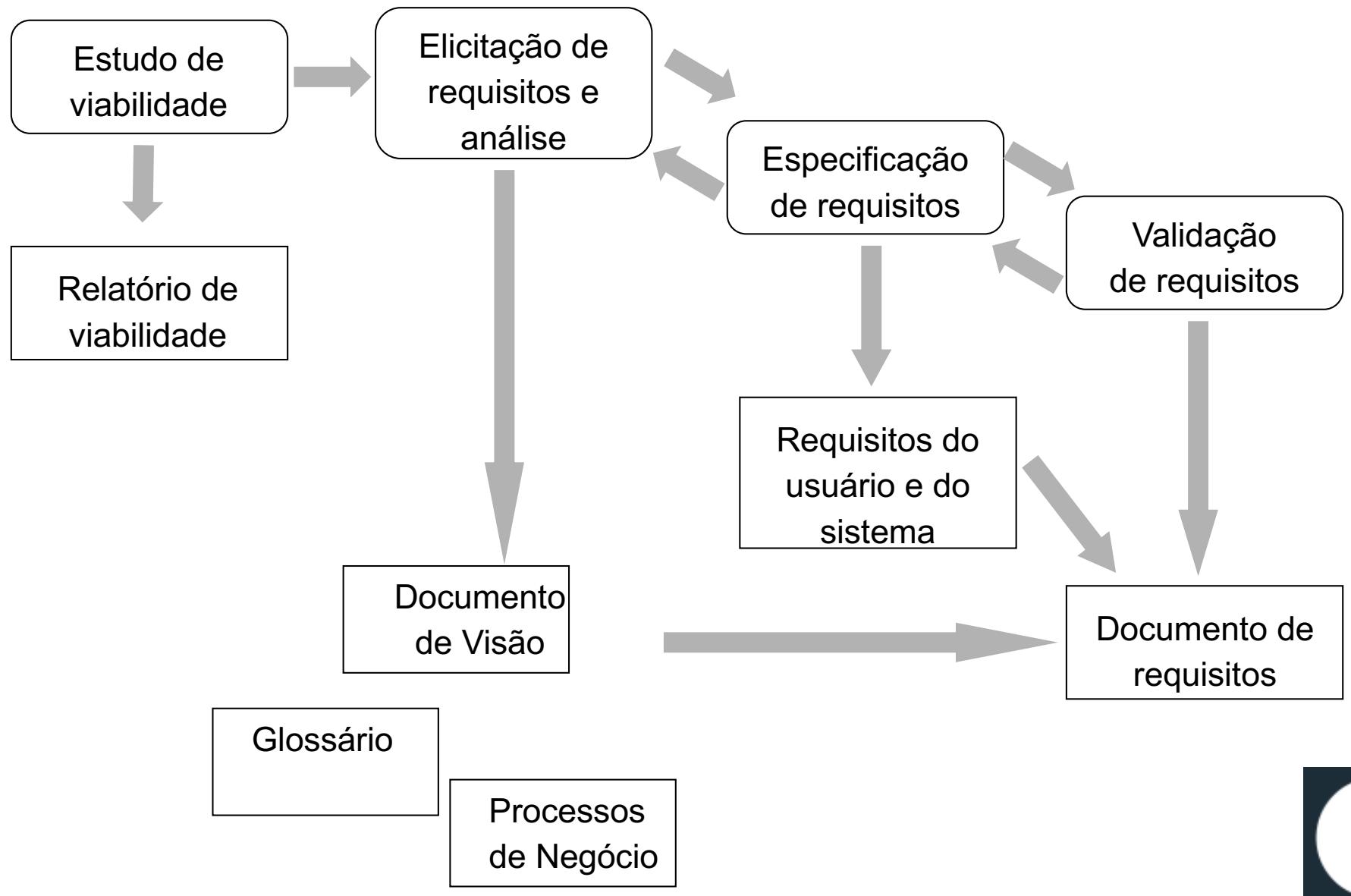
This image cannot currently be displayed.



# *O Processo da Engenharia de Requisitos*



# O Processo da Engenharia de Requisitos

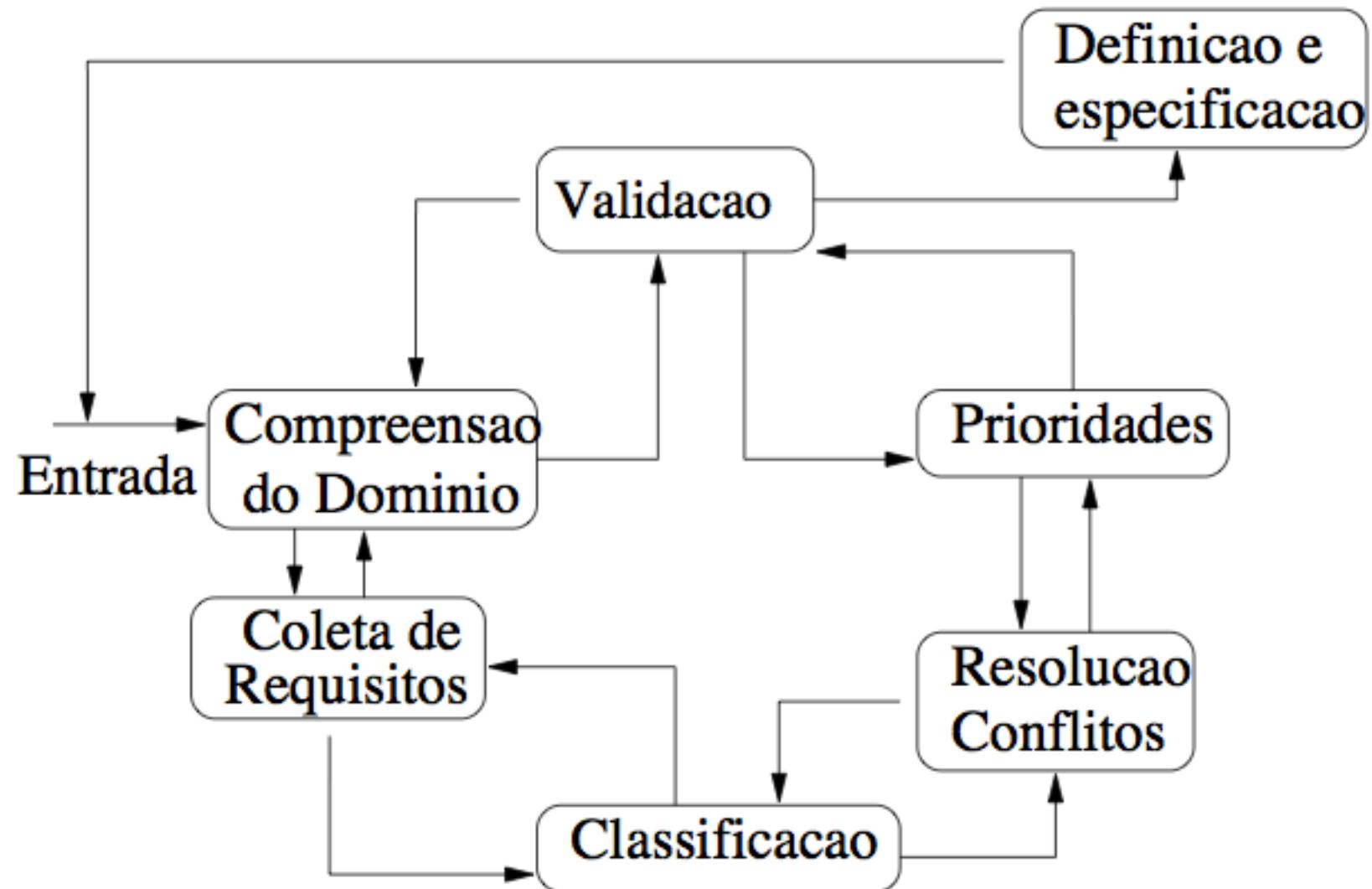


# *Análise de Requisitos*

- Após a identificação dos requisitos, a próxima etapa é a análise dos requisitos e negociação.
- Atividades envolvidas:
  - Classificação: classificação para ajudar na visão global do funcionamento da solução
  - Resolução de conflitos: natureza organizacional e política
  - Priorização: classificação de acordo com importância (baixa/média/alta, escala numérica)
  - Confirmação: completude, consistência e validade
- Não são independentes
  - Tanto a elicitação como a análise são iterativas e tem como objetivo aumentar o grau de entendimento do sistema a cada ciclo

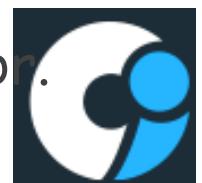


# Análise de Requisitos



# *Análise de Requisitos*

- Projeto prematuro
  - Os requisitos incluem informação prematura de projeto ou implementação?
- Requisitos combinados
  - A descrição dos requisitos descreve um requisito único ou pode ser descritos em vários requisitos diferentes?
- Requisitos desnecessários
  - O requisito é realmente necessária, ou será que é uma mera adição cosmética ao sistema?
- Uso de hardware não padronizado
  - Os requisitos implicam no uso de uma plataforma de hardware não padronizada? Para tomar esta decisão, você precisa conhecer os requisitos de plataforma do computador.



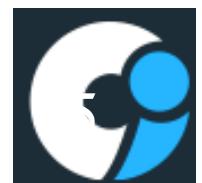
# *Artefato da Elicitação e Análise*

- Um documento de requisitos incluindo uma definição dos requisitos.
  - A maioria das organizações usam um modelo de documento de requisitos pré-definido desenvolvido por uma organização de padrão (e.x. IEEE/ANSI 830-1998).
  - O modelo define a estrutura (tabela de conteúdos) e o estilo do documento.



# *O que é um requisito?*

- Pode variar de uma: (1) declaração abstrata de alto nível de um serviço ou (2) de uma restrição de sistema para (3) uma especificação matemática funcional.
- Isto é inevitável quando os requisitos podem servir uma função dual
  - Pode ser a base para uma proposta de um contrato – portanto deve ser aberta para interpretação;
  - Pode ser a base para o contrato em si – portanto deve ser definido em detalhe;
  - Ambas as declarações podem ser chamadas requisitos.



# *Níveis de requisitos*

- Requisitos de usuário
  - Declarações em linguagem natural mais diagramas de serviços que o sistema fornece e suas restrições operacionais. Escritos para os usuários.
- Requisitos de sistema
  - Um documento estruturado estabelecendo descrições detalhadas das funções, serviços e restrições operacionais do sistema. Define o que deve ser implementado e assim, pode ser parte de um contrato entre o cliente e o desenvolvedor.



# *O sistema LIBSYS*

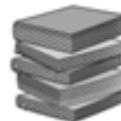
- Um sistema de biblioteca que fornece uma interface única para uma série de banco de dados de artigos em bibliotecas diferentes.
- Os usuários podem pesquisar, baixar e imprimir estes artigos para estudo pessoal.



# *Definições e especificações*

## **Quadro 6.1**

Requisitos de usuário  
do sistema.



### **Definição de requisitos de usuário**

1. LIBSYS deve manter o acompanhamento de todos os dados exigidos pelas agências de licenciamento de direitos autorais no Reino Unido e em outros lugares.

### **Especificação dos requisitos de sistema**

- 1.1 Ao solicitar um documento ao LIBSYS, deve ser apresentado ao solicitante um formulário que regista os detalhes do usuário e da solicitação feita.
- 1.2 Os formulários de solicitação do LIBSYS devem ser armazenados no sistema durante cinco anos, a partir da data da solicitação.
- 1.3 Todos os formulários do LIBSYS devem ser indexados por usuário, nome do material solicitado e fornecedor da solicitação.
- 1.4 O LIBSYS deve manter um registro de todas as solicitações feitas a sistema.
- 1.5 Para materiais aos quais se aplicam os direitos de empréstimo dos autores, os detalhes do empréstimo devem ser enviados mensalmente às agências de licenciamento de direitos autorais que se registraram no LIBSYS.

# *Tipos de Requisitos*

- Requisitos funcionais
  - Declarações de serviços que o sistema deve fornecer, como o sistema deve reagir a entradas específicas e como o sistema deve se comportar em determinadas situações.
- Requisitos não funcionais
  - Restrições sobre serviços ou funções oferecidos pelo sistema tais como restrições de *timing*, restrições sobre o processo de desenvolvimento, padrões, etc.
- Requisitos de domínio
  - Requisitos que vêm do domínio de aplicação do sistema e que refletem as características desse domínio.



# *Requisitos funcionais*

- Descrevem a funcionalidade ou serviços de sistema.
- Dependem do tipo de software, dos usuários esperados e o tipo de sistema onde o software é usado.
- Requisitos funcionais de usuário podem ser declarações de alto nível do que o sistema deve fazer, mas os requisitos funcionais de sistema devem descrever os serviços de sistema em detalhe.



# *Exemplos de requisitos funcionais*

- O usuário deve ser capaz de pesquisar em todo o conjunto inicial de banco de dados ou selecionar um subconjunto a partir dele.
- O sistema deve fornecer telas apropriadas para o usuário ler os documentos no repositório de documentos.
- Para todo pedido deve ser alocado um identificador único (ORDER\_ID) no qual o usuário deve ser capaz de copiar para a área de armazenamento permanente da sua conta.



# *Imprecisão de requisitos*

- Problemas surgem quando os requisitos não são precisamente definidos.
- Requisitos ambíguos podem ser interpretados de maneiras diferentes pelos desenvolvedores e usuários.
- Considere o termo ‘telas apropriadas’
  - Intenção do usuário – tela de propósito especial para cada tipo diferente de documento;
  - Interpretação do desenvolvedor – fornece uma tela de texto que mostra o conteúdo do documento.



# *Requisitos completos e consistentes*

- Em princípio, requisitos devem ser ambos, completos e consistentes.
- Completude
  - Eles devem incluir descrições de todos os recursos requeridos.
- Consistência
  - Não deve haver conflitos ou contradições nas descrições dos recursos de sistema.
- Na prática, é impossível produzir um documento de requisitos completo e consistente.



## *Mais Exemplos de Requisitos Funcionais*

- 1.O sistema deve ser capaz de armazenar todas as informações sobre seus clientes(RG, CPF, Nome, data de nascimento e endereço) no banco de dados.
- 2.O sistema deverá atribuir um identificador único (código) para cada pedido de produtos.
- 3.O sistema deverá cancelar automaticamente um orçamento que tenha sido feito há mais de 30 dias e não tenha sido transformado em venda.



# *Requisitos não funcionais*

- Estes definem propriedades e restrições de sistema, por exemplo, confiabilidade, tempo de resposta e requisitos de armazenamento. Restrições são capacidade de dispositivos de E/S, representações de sistema, etc.
- Podem ainda estar relacionados a portabilidade, de SO, de BD, etc.
- Requisitos de processo podem também ser especificados impondo uma ferramenta CASE particular, linguagem de programação ou método de desenvolvimento.
- Requisitos não funcionais podem ser mais críticos do que os requisitos funcionais. Se estes não forem atendidos, o sistema é inútil.



# *Classificações de requisitos não funcionais*

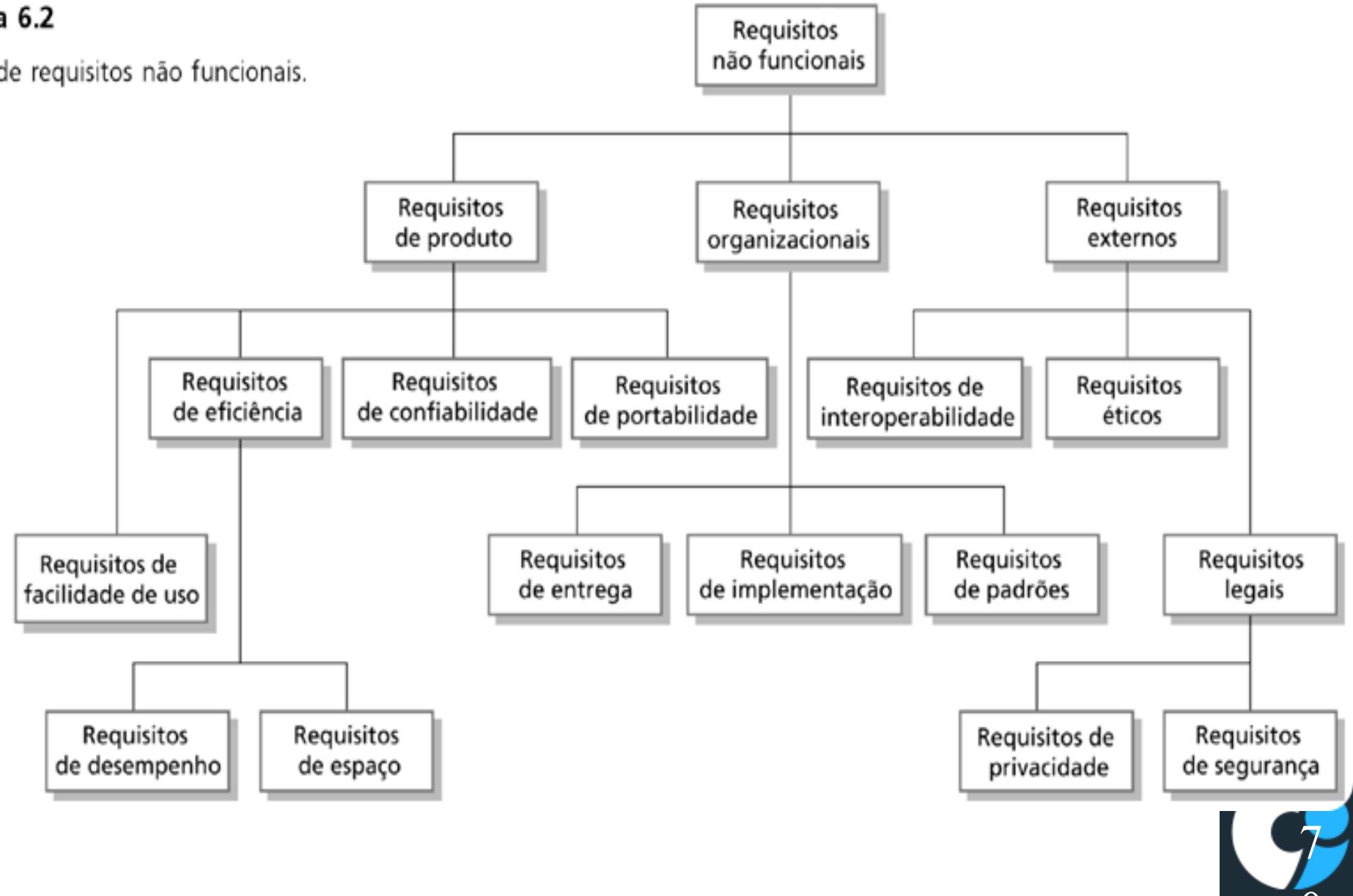
- Requisitos de produto
  - Requisitos que especificam que o produto entregue deve se comportar de uma maneira particular, por exemplo, velocidade de execução, confiabilidade, etc.
- Requisitos organizacionais
  - Requisitos que são uma consequência de políticas e procedimentos da organização, por exemplo, padrões de processo usados, requisitos de implementação, etc.
- Requisitos externos
  - Requisitos que surgem a partir de fatores externos ao sistema e seu processo de desenvolvimento, por exemplo, requisitos de interoperabilidade, requisitos legais, etc.



# *Tipos de requisitos não funcionais*

**Figura 6.2**

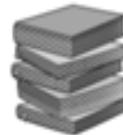
Tipos de requisitos não funcionais.



# *Exemplos de requisitos não funcionais*

## Quadro 6.2

Exemplos de requisitos não funcionais.



### **Requisito de produto**

8.1 A interface de usuário para o LIBSYS deve ser implementada como simples HTML, sem frames ou applets de Java.

### **Requisito organizacional**

9.3.2 O processo de desenvolvimento do sistema e os documentos a serem entregues devem estar em conformidade com o processo e produtos a serem entregues definidos em XYZCo-SP-STAN-95.

### **Requisito externo**

10.6 O sistema não deve revelar quaisquer informações pessoais sobre os usuários do sistema ao pessoal da biblioteca que usa o sistema, com exceção do nome e número de referência da biblioteca.



# *Exemplos de Métricas para requisitos não funcionais*

<b>Propriedade</b>	<b>Medida</b>
Velocidade	Transações processadas/ <u>seg</u> Tempo de resposta do usuário/evento
Tamanho	<u>K</u> bytes Nº de chips de RAM
Facilidade de uso	Tempo de treinamento Nº de quadros de ajuda
Confiabilidade	Tempo médio de falhas Probabilidade de indisponibilidade Taxa de ocorrência de falhas
Robustez	Tempo de reinício após falha Percentual de eventos causando falhas Probabilidade de corrupção de dados após falha
Portabilidade	Percentual de declarações dependentes do destino Nº de sistemas destino



## *Requisitos de domínio*

- Derivados do domínio de aplicação e descrevem características de sistema que refletem o domínio.
- Podem restringir os requisitos funcionais existentes ou estabelecer como cálculos específicos devem ser realizados.
- Se os requisitos de domínio não forem satisfeitos, o sistema pode não funcionar.

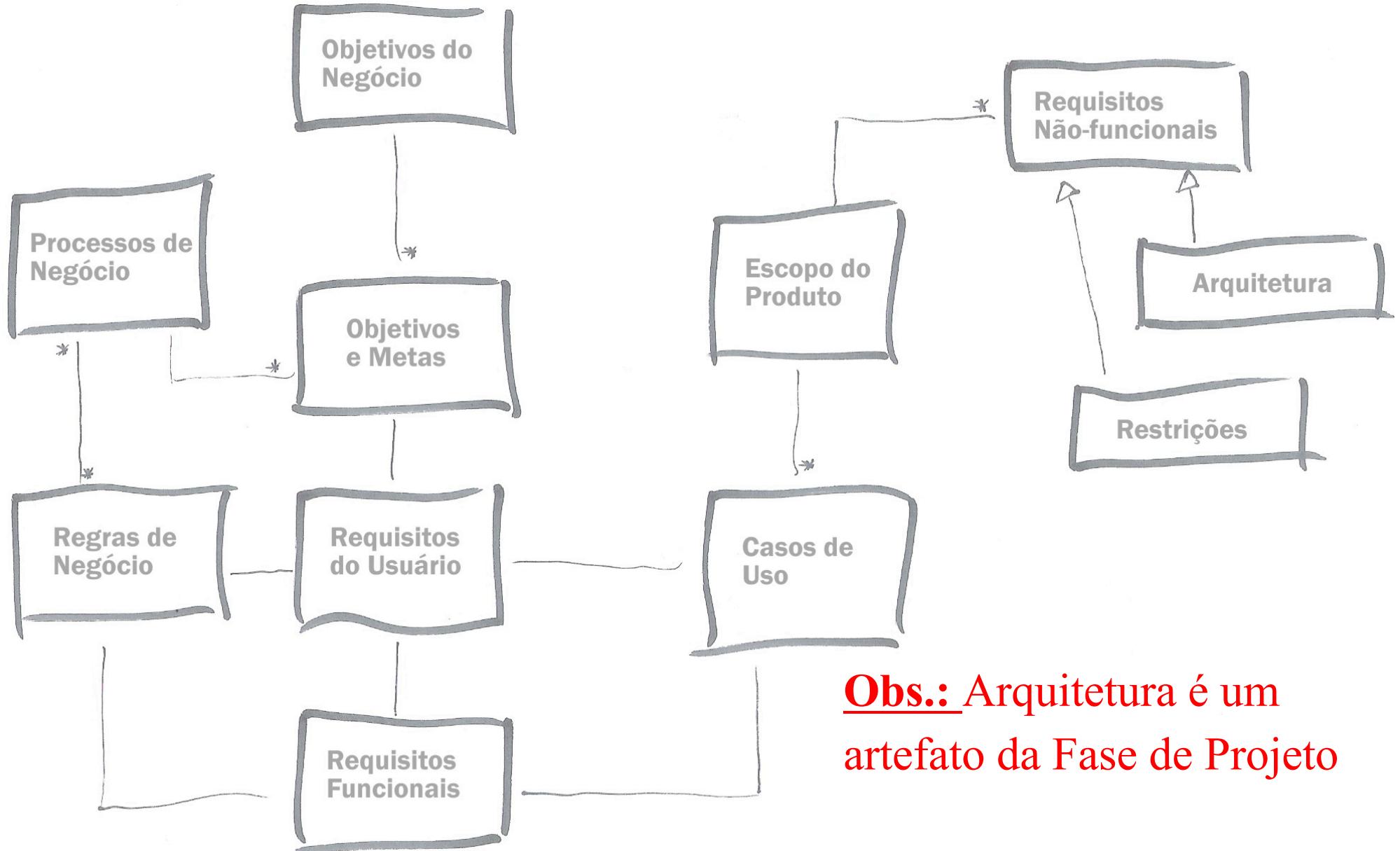


# *Requisitos de domínio do sistema de bibliotecas*

- Deve existir uma interface padrão com o usuário para todos os bancos de dados baseada no padrão ABC-98
  - Devido a restrições de direitos autorais, alguns documentos devem ser apagados assim que chegam. Dependendo dos requisitos do usuário, tais documentos devem ser impressos em impressora local ou remota
  - A desaceleração do trem deve ser computada através da fórmula:  
$$-D_{trem} = D_{controle} + D_{gradiente}$$
 onde ..



# Artefatos da Engenharia de Requisitos



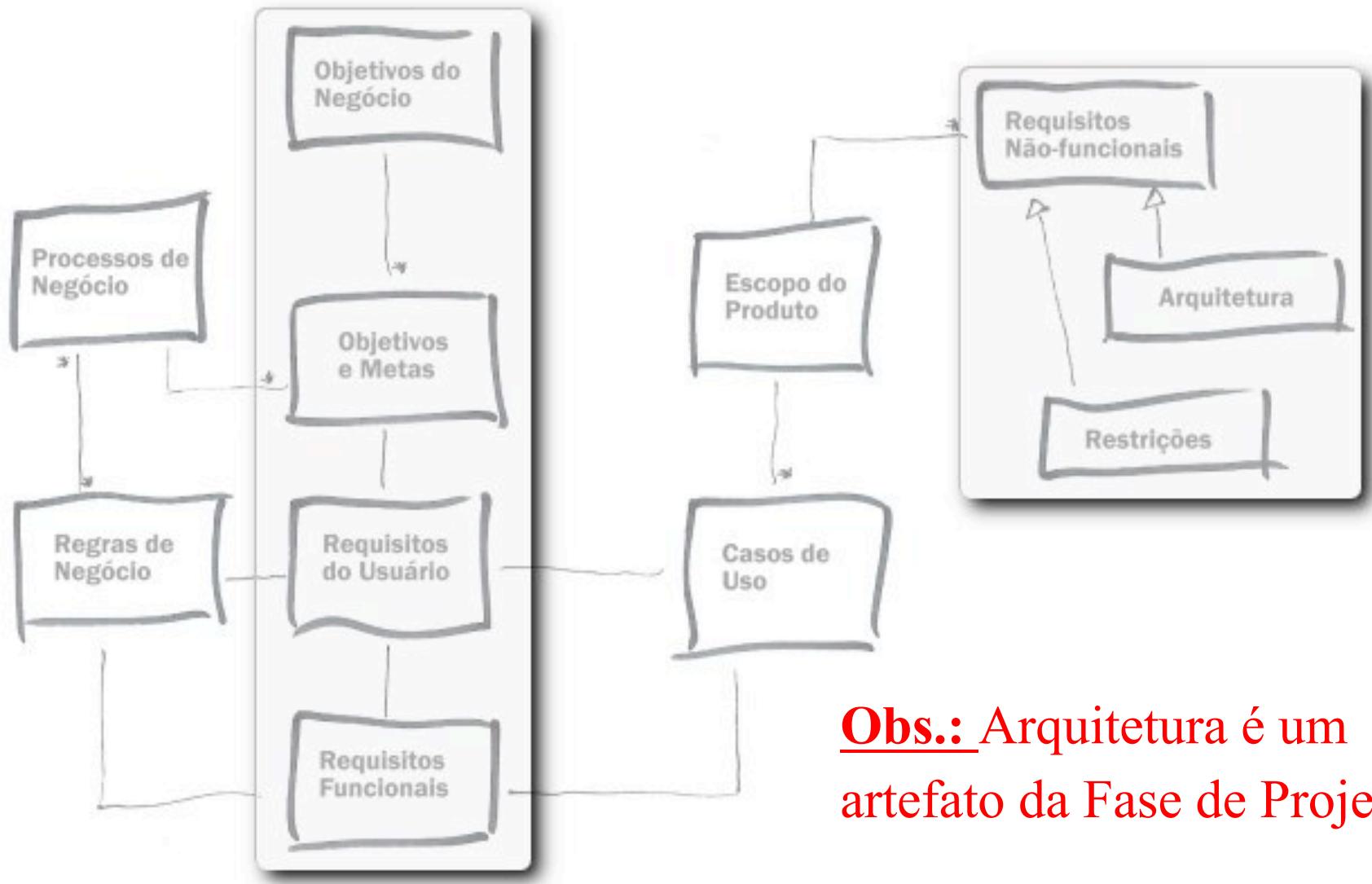
**Obs.:** Arquitetura é um artefato da Fase de Projeto

# *Estruturando Requisitos*

- Objetivos de negócio são quebrados em Objetivos e Metas:
  - “reduzir em 10% os custos de vendas é uma meta colocada para o processo de vendas”
- Regras de Negócio x Requisitos do Usuário
  - As regras são mais específicas e voláteis
  - Requisito: “nas operações de vendas o sistema deve permitir a aplicação de um desconto sobre o valor total da transação”.
  - Regra: “O desconto máximo de 10% só pode ser aplicado em vendas superiores a R\$ 100,00”.



# *Artefatos da Engenharia de Requisitos*



**Obs.:** Arquitetura é um  
artefato da Fase de Projeto



# *Estruturando Requisitos*

- Requisitos do usuário são solicitações que podemos quebrar em vários Requisitos Funcionais.
  - Exemplo: “Emitir uma Nota Fiscal”
- Requisito de usuário pode ser transformar em um Caso de Uso. Isso se o Escopo do Projeto incorporar o requisito.
- Casos de Uso auxiliam o agrupamento lógico dos Requisitos Funcionais, ou seja, estes (RF) são as “menores unidades de solicitações” dos usuários ou clientes.



# *Problemas de requisitos de domínio*

- Facilidade de entendimento
  - Requisitos são expressos na linguagem do domínio de aplicação;
  - Isso não é, freqüentemente, compreendido pelos engenheiros de software que estão desenvolvendo o sistema.
- Implícito
  - Especialistas em domínio compreendem a área tão bem que não pensam em tornar os requisitos de domínio explícitos.



# *Documento de Especificação de Requisitos*

- O documento de requisitos do software deve ser composto por sentenças em linguagem natural, seguindo determinados padrões:
  - 1) Use ‘deve’ para requisitos obrigatórios, e ‘deveria’ ou ‘pode’ para requisitos desejáveis.
    - Exemplo: “**O sistema deve** rodar em microcomputadores da linha IBM PC que possuam microprocessador 486 DX ou superior.”
  - 2) Os requisitos devem estar organizados logicamente, como por exemplo, inicialmente todos os requisitos de entrada, depois os de processamento e por último os requisitos de saída.



## *Documento de Especificação de Requisitos (cont.)*

- O documento de requisitos do software deve ser composto por sentenças em linguagem natural, seguindo determinados padrões:
  - 3) Cada requisito deve ter um identificador único, por exemplo, um identificador numérico, para posterior referência.
  - 4) Os requisitos do software devem estar divididos em *requisitos funcionais* e *não funcionais (de qualidade)*.
  - 5) Evitar o uso de jargões de computação.



# *Formato da Especificação de Requisitos*

- Existem vários padrões de especificações de requisitos.
- Um exemplo:
  - I. Visão Geral do Sistema
  - II. Requisitos Funcionais
  - III. Requisitos de Qualidade
  - IV. Apêndice
- Padrão IEEE/ANSI 830/1998.
- A Especificação pode ser acompanhada de um PROTOTIPO executável (ou em papel).



# *Formato sugerido pelo padrão IEEE*

- 1. Introdução**
  1. Propósito do documento de requisitos
  2. Escopo do produto
  3. Definições, acrônimos e abreviaturas
  4. Referências
  5. Visão geral do restante do documento
- 2. Descrição Geral**
  1. Perspectiva do produto
  2. Funções do produto
  3. Características dos usuários
  4. Restrições gerais
  5. Suposições de dependências
- 3. Requisitos específicos (requisitos funcionais e não-funcionais)**
- 4. Apêndices**
- 5. Índice**

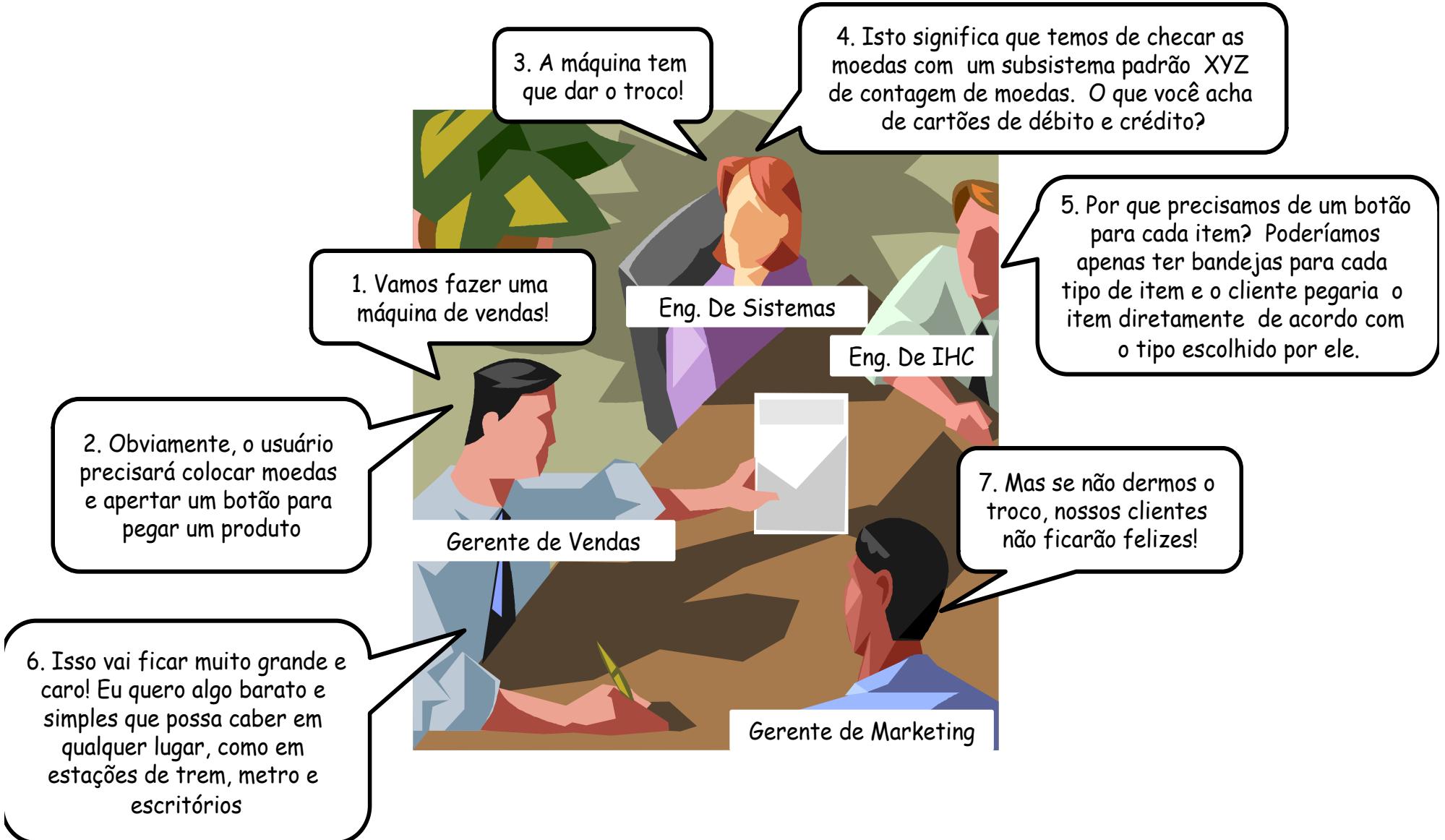


# *Problemas dos requisitos*

- Os requisitos não satisfazem as reais necessidades dos clientes do sistema.
- Os requisitos são **inconsistentes e/ou incompletos.**
- O custo alto para se fazer mudanças de requisitos depois destes terem sido concordados.
- Existirem mal entendidos entre clientes, aqueles que desenvolvem os requisitos do sistema e os engenheiros de software que desenvolvem ou mantêm o sistema.



# Problemas dos requisitos: exemplo



# Análise do exemplo

Os requisitos encontrados pela equipe de desenvolvimento parecem ser simples...

... até você tentar escrevê-los!

E, de repente, um conjunto de problemas surgem!

As metas do projeto não estão claras

As prioridades dos stakeholders diferem

As pessoas tem interpretações diferentes

As pessoas fazem suposições não declaradas para os demais

Testadores reclamam que os requisitos não podem ser verificados



# *Perguntas para o próximo mini-teste?*

1. Quais são as fases da Engenharia de Requisitos?
2. Quais passos você utilizaria para realizar uma Elicitação de Requisitos de Software?
3. Que tipo de problema as técnicas de elicitação tratam? Quais são as principais técnicas de Elicitação de Requisitos Software? Qual técnica de Elicitação de Requisitos de Software você empregaria? Por que?
4. Quais as atividades envolvidas na especificação de requisitos.
5. Cite exemplos de requisitos não-funcionais.
6. Quais os principais artefatos gerados na Elicitação e Análise de Requisitos?



*Dúvidas? Obrigado.*

*raoni@ci.ufpb.br*

