Engenharia de Requisitos

Tecnologia de Software

Aula 1: Visão geral

Fábio Levy Siqueira
levy.siqueira@usp.br

Estrutura do curso

Aula	Data	Assunto	Atividade
1	12/05	Visão geral	
2	19/05	Requisitos e inovação	Jornada do usuário
3	26/05	Visão	Documento visão
4	02/06	História do usuário	
5	09/06	Técnicas de elicitação	Workshop de histórias
6	16/06	Priorização e estimativa	Priorização e estimativa de histórias
7	23/06	Caso de Uso I	Redação de casos de uso
8	30/06	Caso de Uso II	Redação de casos de uso
9	07/07	Gerência de requisitos	
10	**14/07**	**Prova**	

Avaliação

- Critério
  - Média = $\frac{\text{Exercícios} + 2*\text{Projeto} + 3*\text{Prova}}{6}$
    - *Exercícios*
      - Seguem o Projeto Integrado
    - *Projeto*
      - Histórias do usuário: cartão e confirmação
      - Caso de uso
      - (Quantidade de Histórias e UCs varia conforme o tamanho do grupo)
    - *Prova*
      - Individual

Bibliografia básica

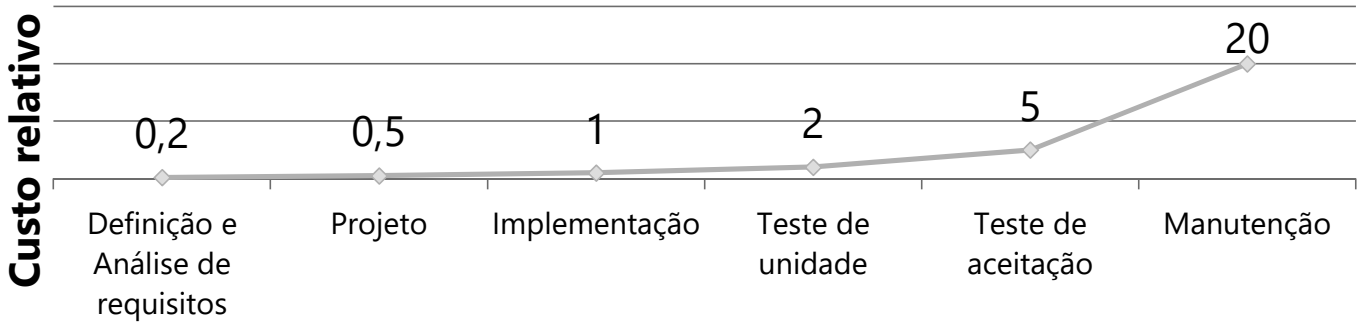
- COHN, M. **User Stories Applied**. Addison-Wesley, 2004.
- LEFFINGWELL, D.; WIDRIG, D. **Managing Software Requirements: A Use Case Approach**. 2ª edição. Addison-Wesley, 2003.
- POHL, K.; RUPP, C. **Requirements Engineering Fundamentals: A Study Guide for the Certified Professional for Requirements Engineering Exam Fondation Level**. 2ª edição. Rocky Nook, 2015.
- WIEGERS, K.; BEATTY, J. **Software Requirements**. 3a edição. Microsoft Press, 2013.

Conceitos

Engenharia de Requisitos

Função interdisciplinar que media entre os domínios do adquirente e fornecedor para estabelecer e manter os requisitos a serem cumpridos pelo sistema, software ou serviço de interesse (ISO, 2018)

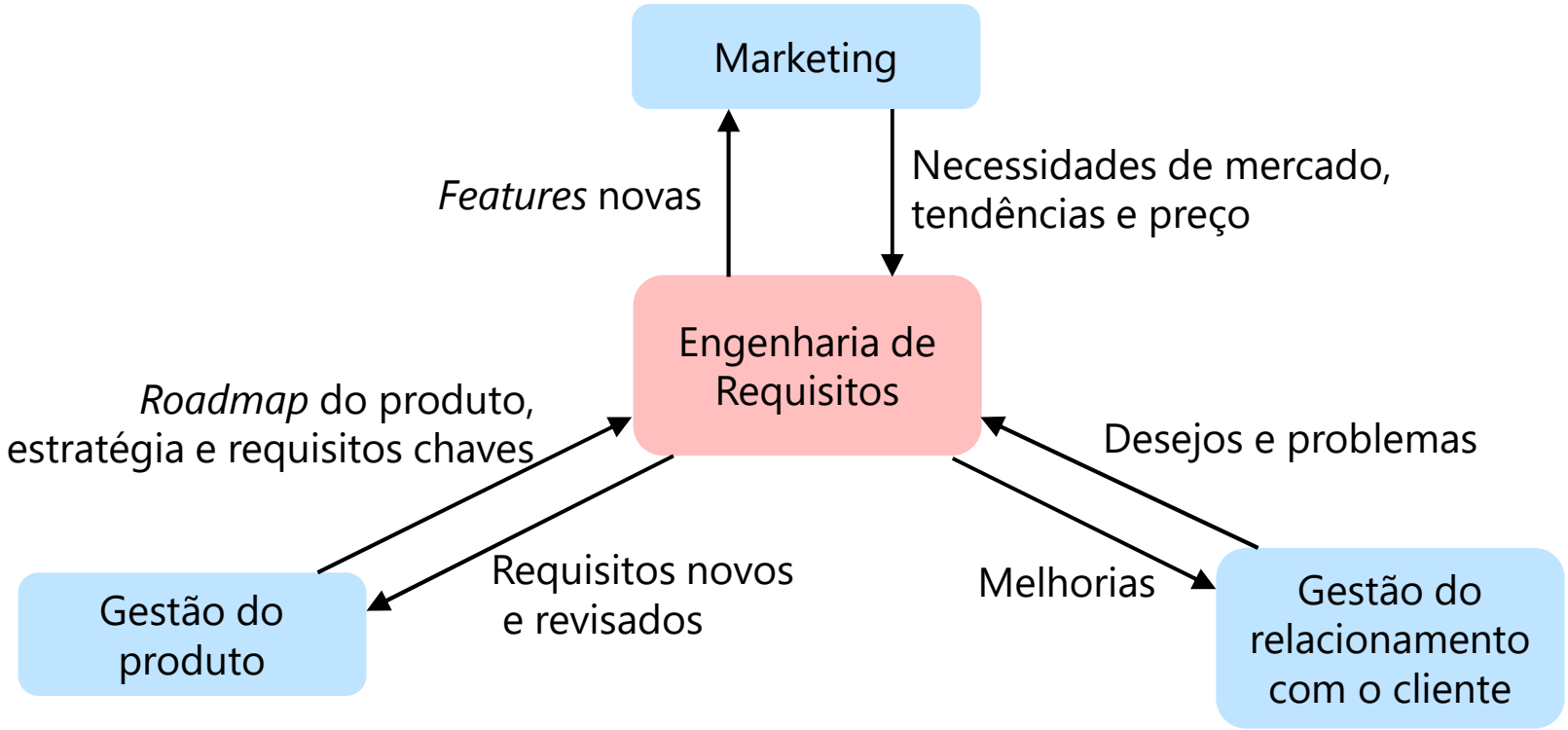
- Importância para o sucesso do projeto
  - Fundamental entender as necessidades dos *stakeholders*
  - Custo alto para corrigir defeitos de requisitos



(Davis, 1993 apud Leffingwell; Widrig, 2003)

Engenharia de Requisitos

- Faz parte de um contexto organizacional
  - (Pohl, 2011)



Problemas

- Alguns problemas para a ER (Wiegers e Beatty, 2013)
  - Envolvimento do usuário insuficiente
  - Planejamento impreciso
    - Planejamento feito com poucas informações
  - Aumento incontrolável dos requisitos
  - Ambiguidade dos requisitos
  - Requisitos desnecessários
    - Funcionalidade que o "usuário vai adorar"
  - Stakeholders menosprezados
  - Dificuldade de o stakeholder expressar o que quer
  - Dificuldade de o ER entender o stakeholder

Stakeholder

Pessoas ou organizações que tem um interesse no sistema a ser desenvolvido (Pohl, 2011)

- Tipicamente tem seus próprios requisitos
- Uma pessoa pode representar vários *stakeholders*

- Tipos (Leffingwell, 2011)
  - *Stakeholders* do sistema
    - Usa o sistema diretamente
    - Trabalha com os resultados de quem usa o sistema
    - Será impactado com a implantação e operação do sistema
  - *Stakeholders* do projeto
    - Tem investimento no orçamento ou no cronograma
    - Tem interesse em entender como a solução será desenvolvida
    - Estará envolvido em propagandear, vender, instalar ou manter

Stakeholder

- Envolvimento (Leffingwell, 2011)
  - Precisam ser informados
  - Precisam ser consultados
  - São parceiros no desenvolvimento
  - Tem controle dos resultados (tem a decisão final)

- *Exemplos*
  - Usuários
  - Cliente
  - Agências regulatórias
  - Desenvolvedores

Requisito

Uma afirmação que traduz ou expressa uma **necessidade** e suas **restrições** e **condições** associadas. (ISO, 2018, p.4)

- Não devem (*em teoria*) ter detalhes de implementação

- Uma outra definição clássica

a) Uma **condição** ou **capacidade** necessária por um ***usuário*** para resolver um problema ou atingir um objetivo.

b) Uma **condição** ou **capacidade** que deve ser cumprida ou possuída por um sistema ou componente do sistema para satisfazer um contrato, padrão, especificação, ou outros documentos formalmente impostos.

(IEEE, 1990, p.172)

Requisito

- Tipos de requisitos
  - Funcionais
  - Não funcionais (ou de qualidade)
  - Restrições

Requisitos funcionais

- Especificam a funcionalidade que o sistema deve prover aos usuários

  Ações fundamentais que devem acontecer no software ao aceitar e processar as entradas e ao processar e gerar as saídas (IEEE, 1998, p.16)

- *Exemplos* (IEEE, 1998)
  - Sequência exatas de operações
  - Validações nas entradas
  - Respostas para situações anormais (ex.: tratamento de erros e recuperação)
  - Relações entre saídas e entradas (ex.: sequências e fórmulas)

- *Exemplo*: loja virtual
  - O software deve permitir a busca de produtos por palavra chave
  - O usuário só pode adicionar ao carrinho de compras os produtos disponíveis

Requisitos não funcionais

- Definem propriedades de qualidade do sistema a serem desenvolvidas
  - Afetam todo o sistema ou uma parte dele
  - Influenciam a arquitetura
  - Também chamados de **requisitos de qualidade**
  - Relacionadas a desempenho, segurança, usabilidade etc.

- *Exemplo*: loja virtual
  - A verificação do cadastro do usuário deve demorar no máximo 5s
  - Apenas um técnico com privilégios de Administrador deve ter acesso aos logs do sistema

Restrições

- Restrição organizacional ou tecnológica que afeta como o software deve ser desenvolvido
  - Orçamento, plataformas, linguagens, leis, soluções tecnológicas etc.
  - Muitas vezes considerada como requisitos não funcionais

- *Exemplo*: loja virtual
  - A interface web deve usar o framework React
  - O software deve ser entregue até dia 02/06
  - O projeto deve usar Scrum

Requisitos

- Quais são os tipos de requisitos dessas afirmações?
  1. Ao cadastrar uma senha, o sistema deve verificar se a senha de acesso tem no mínimo 8 caracteres.
     - O DDD do telefone deve ter 2 dígitos.
  2. As senhas armazenadas no sistema devem ser criptografadas, sendo protegidas contra roubo.
  3. As senhas devem ser criptografadas usando o algoritmo AES e usando sal aleatório de 8 bytes.

Requisitos não funcionais

Requisitos não funcionais

- Existem várias taxonomias
  - Wiegers e Beaty (2013)
  - Chung et al. (2000)
  - van Lamsweerde (2009)
  - **ISO 25010 (2011)**

ISO 25000

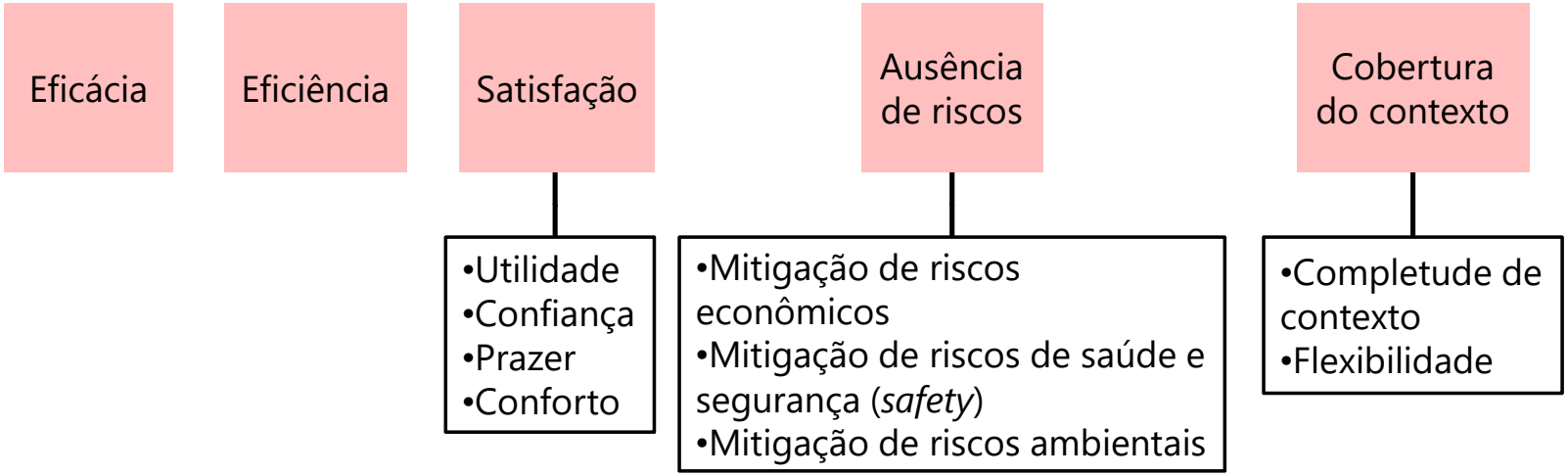
- SQuaRE
  - *Software Product Quality Requirements and Evaluation*
- Família de normas
  - Reorganização e revisão de outras normas
    - ISO 9126 e ISO 14598
  - ISO 25010 (2011) é a parte que define o modelo de qualidade

- Define características que podem ser usadas para especificar, medir e avaliar a qualidade
  - Características são divididas em subcaracterísticas
  - Característica/subcaracterística consideram **graus**
    - O grau em que algo é atendido

ISO 25010

- Define dois modelos
  - Qualidade em uso
    - Representa o quanto o uso do software cumpre as necessidades e atinge as metas de um usuário
      - Resultado do uso do software
  - Qualidade do produto
    - Trata de propriedades de qualidade do produto
    - Útil para especificação ou para a avaliação de um produto
      - Pode ser usada como **taxonomia de requisitos não funcionais**

Qualidade em Uso



- Dependem do **contexto de uso**
  - Outros fatores além do software
    - Características do usuário, características da tarefa, hardware, ambiente de operação e características do ambiente social

Qualidade em Uso

- Eficácia
  - Acurácia ou completude com que o usuário atinge os objetivos
- Eficiência
  - Recursos despendidos em relação à acurácia e completude com o quais o usuário atinge os objetivos
- Satisfação
  - O grau que as necessidades do usuário estão satisfeitas ao usar o produto

Qualidade em Uso

- Ausência de riscos
  - Grau em que o produto mitiga os potenciais riscos
- Cobertura do contexto
  - Grau que o produto pode ser usado com eficiência, eficácia, ausência de risco e satisfação nos contextos definidos ou em outros contextos

ISO 25010 – Qualidade do Produto

Adequação funcional	Eficiência de Execução	Compatibilidade	Usabilidade	Confiabilidade	Segurança (*security*)	Manutenibilidade	Portabilidade
Completeza funcional	Comporta-mento no tempo	Coexistência	Reconhecibi-lidade da adequação	Maturidade	Confidenci-alidade	Modularidade	Adaptabili-dade
Correção funcional	Utilização de recursos	Interoperabi-lidade	Apreensibili-dade	Disponibili-dade	Integridade	Reusabilidade	Capacidade para ser instalado
Apropriabili-dade funcional	Capacidade		Operacionali-dade	Tolerância a falhas	Não repúdio	Analisabilidade	Capacidade para substituir
			Proteção a erro do usuário	Recuperabili-dade	Responsa-bilidade	Modificabilidade	
			Estética da interface do usuário		Autentici-dade	Testabilidade	
			Acessibili-dade				

Qualidade do Produto

- Considera um ambiente diferente do contexto de uso real
  - É uma "estimativa" da qualidade em uso
- Características
  - Adequação funcional
    - O quanto o produto provê funções que cumprem as necessidades especificadas
    - **Em geral não se considera como RNF**
  - Eficiência de execução
    - O desempenho relativo a quantidade de recursos usados
  - Compatibilidade
    - O quanto o produto pode compartilhar informações e recursos
  - Usabilidade
    - O quanto é fácil usar o produto

Qualidade do Produto

- Características (continuação)
  - Confiabilidade
    - O quanto o produto executa suas funções nas condições especificadas e no período de tempo especificado
  - Segurança
    - O quanto as informações são protegidas
  - Manutenibilidade
    - O quanto é fácil modificar o produto
  - Portabilidade
    - O quanto é fácil colocar o produto em outro ambiente

Taxonomias de RNF

- Como usar uma taxonomia?

- Quais RNFs devemos definir em um projeto?
  - Quanto mais melhor?

Escopo do sistema

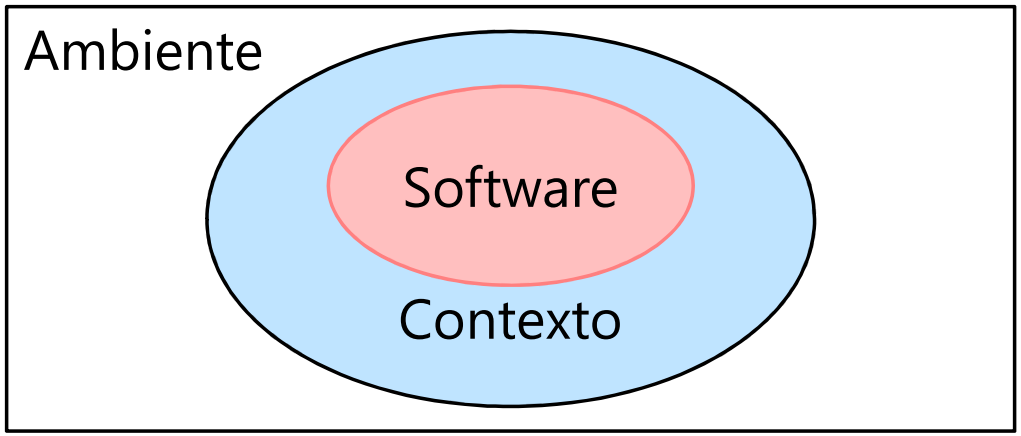
Contexto

- Um software está **sempre** em um ambiente
  - **Ambiente *as-is***
    - Ambiente que existe **antes** do software
  - **Ambiente *to-be***
    - Ambiente que se **espera** ter com o software (ideal)

- Requisitos tratam do ambiente *to-be*
  - *(Mas o ambiente as-is é importante para entender os requisitos)*

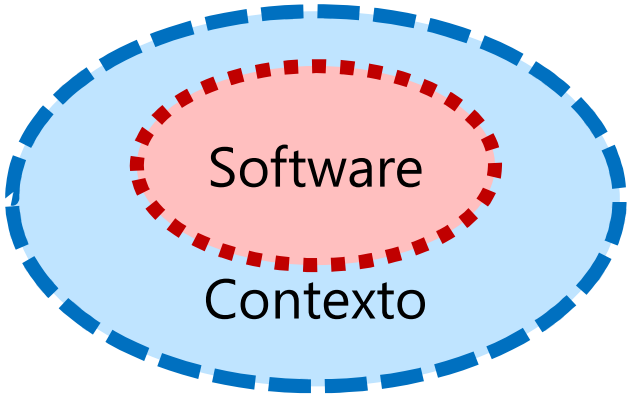
Contexto

- Nem toda informação do ambiente é relevante
  - **Contexto**: parte relevante do ambiente para definir, entender e interpretar os requisitos do software



Contexto

- Existem duas fronteiras
  - Fronteira do software
    - **Escopo** do software
      - (O que o software tratará)
  - Fronteira do contexto
    - Informações relevantes ao sistema



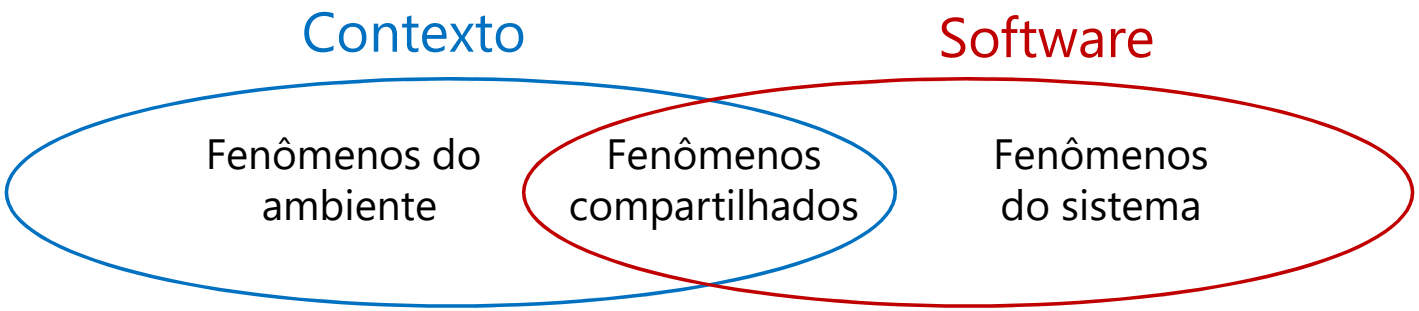
- Essas fronteiras são estáticas?
  - Se não, quando mudam?

Escopo

- O que define o escopo do software?

WRSPM

- Onde estão os requisitos?
  - Modelo WRSPM (Gunter et a., 2000)

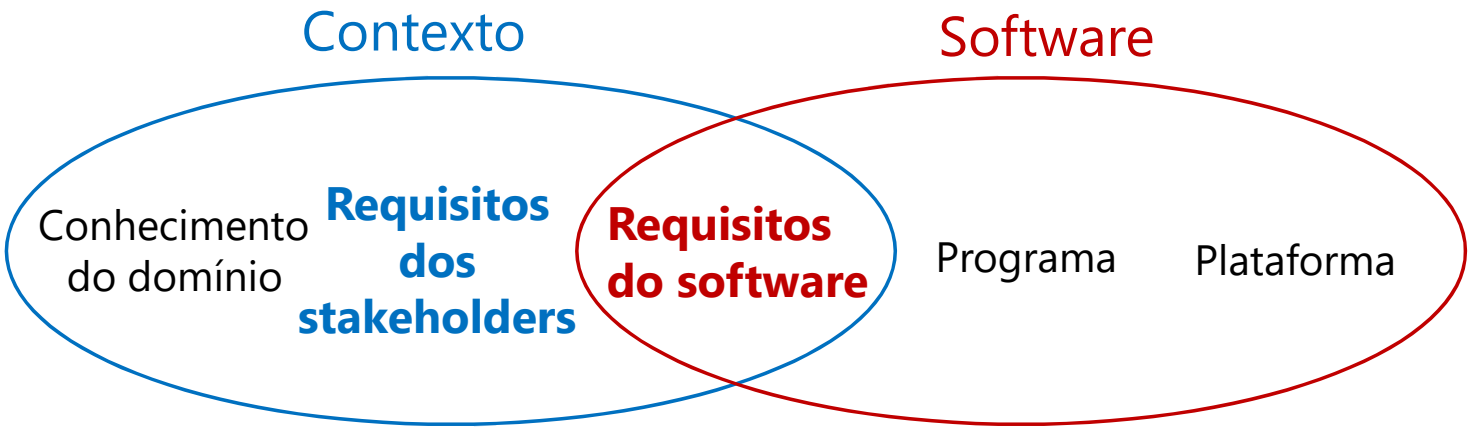


  - *Fenômenos*: eventos, objetos, estados e variáveis

WRSPM

- Artefatos importantes
  - Domínio
    - Afirmações que descrevem propriedades que **sempre são verdadeiras**, independentes do software
  - Requisitos dos *stakeholders* (**requisitos**)
    - Afirmações que descrevem propriedades que os usuários **querem que sejam verdadeiras** no ambiente com a presença do software
  - Requisitos do software (**especificação**)
    - Afirmações que descrevem **o que o software precisa fazer** para atender os requisitos
    - **Informação suficiente para o desenvolvedor construir o software**

WRSPM



- *Exemplo*
  - **Domínio**: o cliente precisa pagar a compra para receber o produto.
  - **Req. Stak.**: o cliente deve fazer o pagamento online.
  - **Req. Soft.**: o sistema deve solicitar os dados de pagamento ao cliente, usando uma conexão segura.

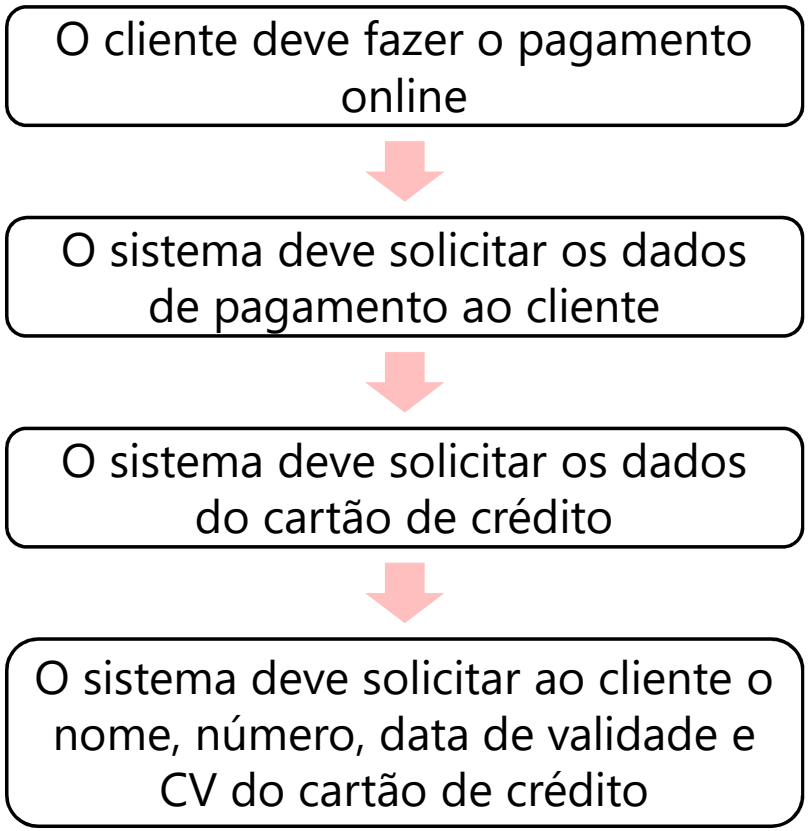
Refinamento

- Mas o requisito de sistema está detalhado **o suficiente** para ser implementado?

> O sistema deve solicitar os dados de pagamento ao cliente, usando uma conexão segura.

Refinamento

- Requisitos são **refinados** em requisitos mais detalhados



Requisito dos stakeholders

Existem outras alternativas de refinamento!
O software pode atender a várias delas!

Último nível:
especificação

Requisito do software

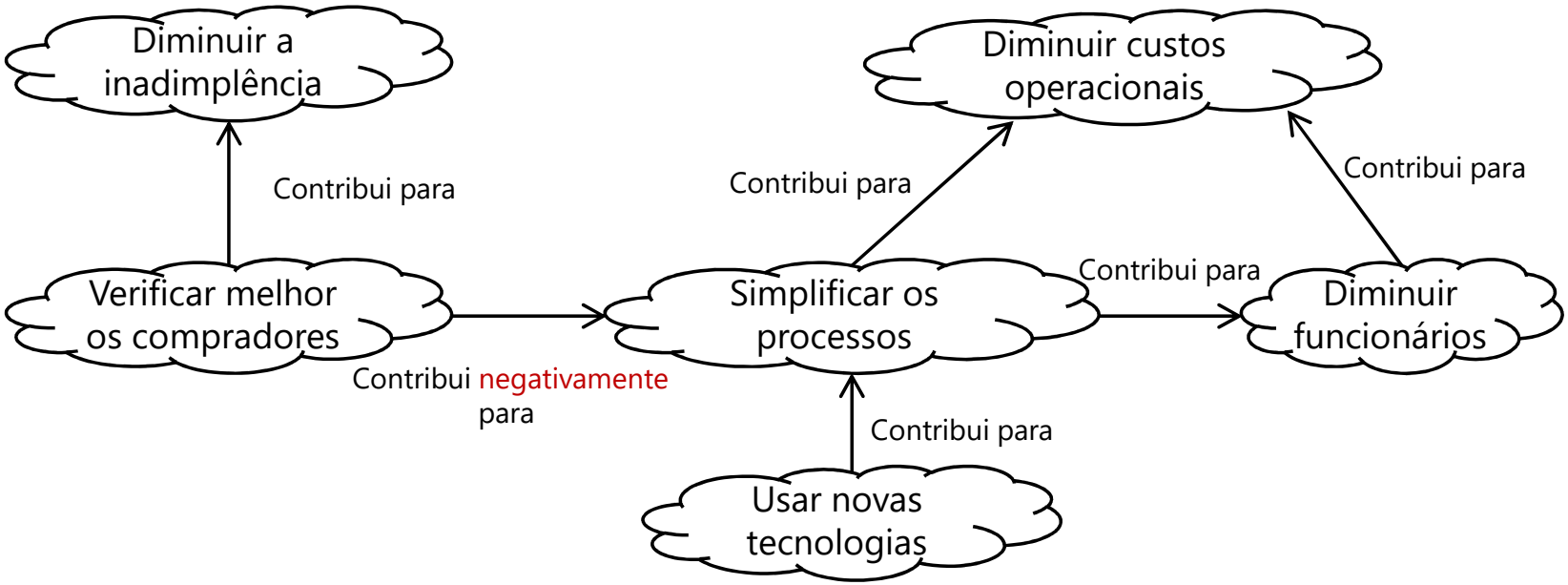
Metas

- Os requisitos são originários de **metas**
  - Objetivos que o software deve atingir (**porquê**)
  - Diferentes *stakeholders* têm diferentes metas
  - Em geral são apresentados de forma vaga
    - *Exemplo*
      - Diminuir custos
      - Diminuir o número de funcionários
      - Usar tecnologias modernas

- Metas mudam lentamente
  - Requisitos e o escopo mudam com mais frequência

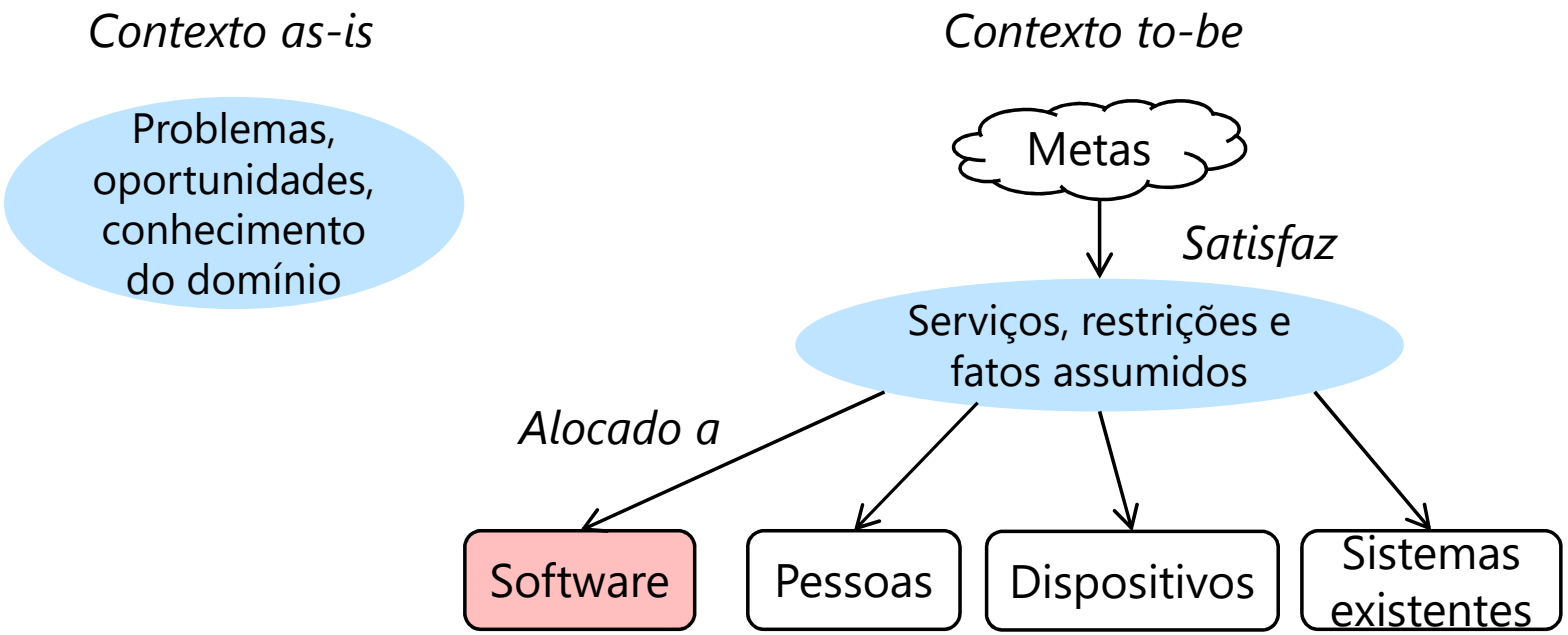
Níveis de refinamento

- Metas são refinadas em outras metas
  - Metas contribuem para outras metas
  - *Exemplo*

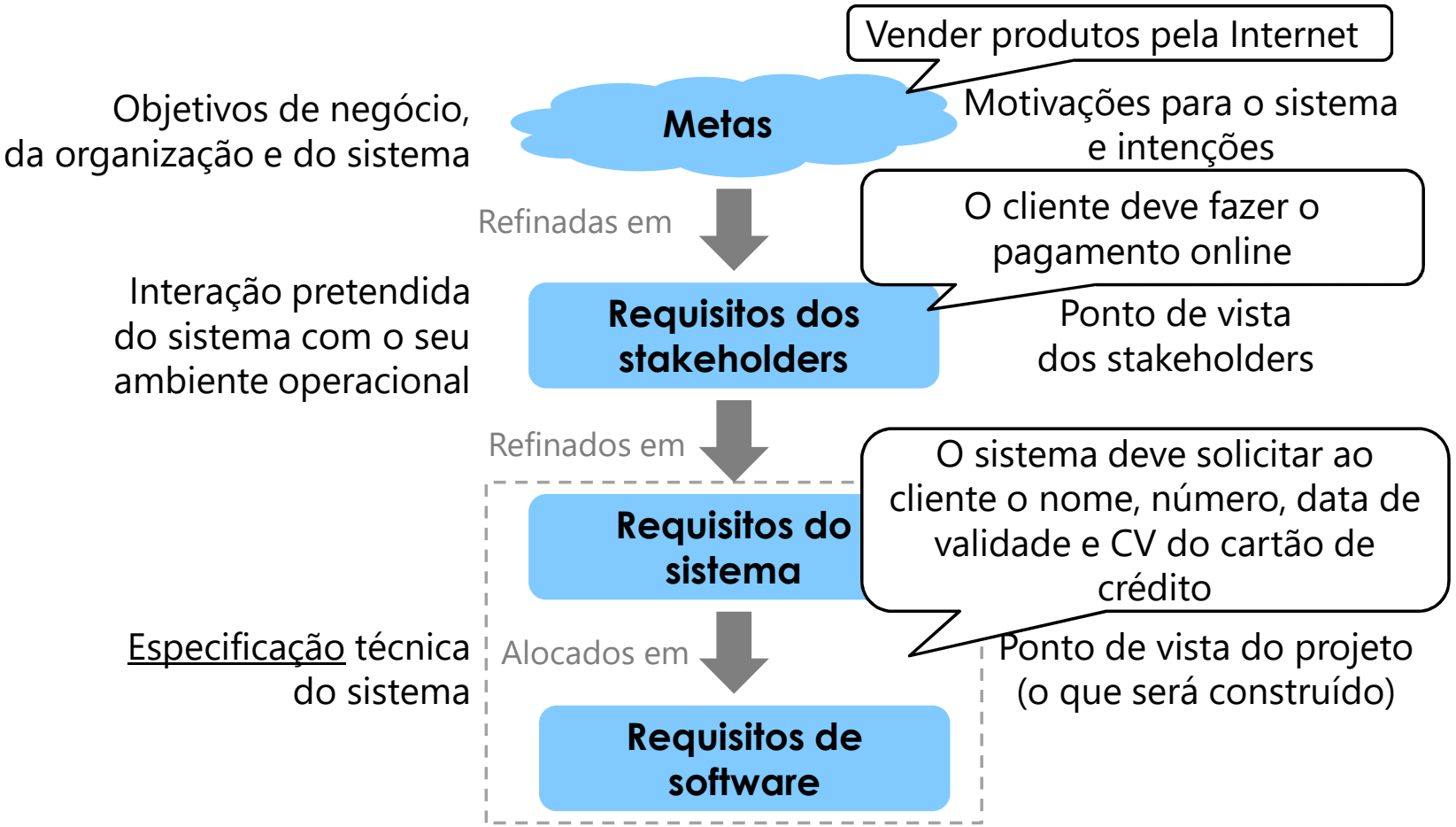


Metas

- Relação das metas e o contexto



Níveis de abstração



Requisitos

- Importante perguntar o "porquê" → Metas
  - Provêm um racional para o requisito
  - Ajudam a identificar regras de negócio
  - Facilitam encontrar requisitos implícitos
  - Ajudam a entender a importância do requisito
  - Facilitam o entendimento do requisito

Requisitos e arquitetura

- Requisitos **não devem** ter detalhes de implementação
  - Mas **decisões arquiteturais** afetam os requisitos
    - Alguns **refinamentos** só são possíveis em uma arquitetura
      - *Exemplo*
        - Se for aplicativo Android, é possível usar GPS ou giroscópio
        - Se usar a autenticação do Google, já confirmamos que a pessoa é válida
    - ...não é possível separar arquitetura dos requisitos...
      - O arquiteto deve participar das atividades de ER

Outros termos

- *Feature* (características)

- Regra de negócio

Referências

- GUNTER, C.; GUNTER, E.; JACKSON, M.; ZAVE, P. **A Reference Model for Requirements and Specifications**. IEEE Software, v.17, n.3, pp.37-43, May/June 2000.
- IEEE. **IEEE Recommended Practice for Software Requirements Specifications**. IEEE Std 830-1998, 1998.
- ISO. **Systems and software engineering – System and software Quality Requirrements and Evaluation (SquaRE) – System and software quality models** ISO/IEC/IEEE 25010. 2011.
- ISO. **Systems and software engineering - Life cycle processes – Requirements engineering**. ISO/IEC/IEEE 29148. 2018.
- LEFFINGWELL, D. **Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise**. Addison-Wesley, 2011.
- POHL, K. **Requirements Engineering: Fundamentals, Principles and Techniques**. Springer, 2010.
- VAN LAMSWEERDE, A. **Requirements Engineering: from System Goals to UML Models to Software Specifications**. Wiley, 2009.
- WIEGERS, K; BEATTY, J. **Software Requirements**. 3ª edição. Microsoft Press, 2013.