Arquitetura de Software – Processo de Definição

João Pedro Oliveira Batisteli

Um breve comentário sobre arquitetura

- **Arquitetura de Software ≠ Estilo Arquitetural**
  - **Arquitetura de Software**: visão global da estrutura do sistema, definindo seus principais componentes, interações, restrições e decisões críticas.

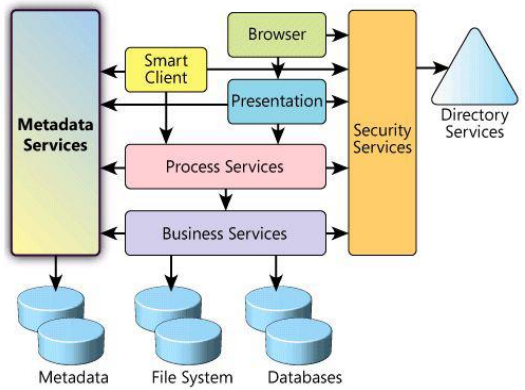
  - **Estilos Arquiteturais** (ex.: MVC, em camadas): são formas padronizadas de organizar sistemas, ou seja, maneiras específicas de implementar a arquitetura.

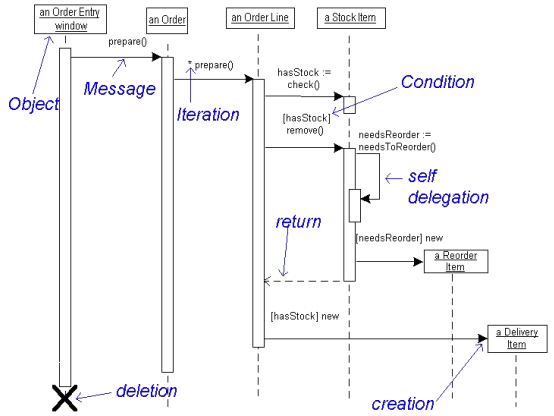
- A arquitetura é mais abstrata e estratégica.
- Os estilos (MVC, camadas, microserviços, etc.) são instâncias ou modelos de arquitetura.

Projeto x arquitetura

- O projeto da arquitetura de software
  - Mais **alto nível** com granularidade **macro** do software, definindo apenas os components e a comunicação/iteração entre eles
  - **Satisfaz os requisitos de qualidade**

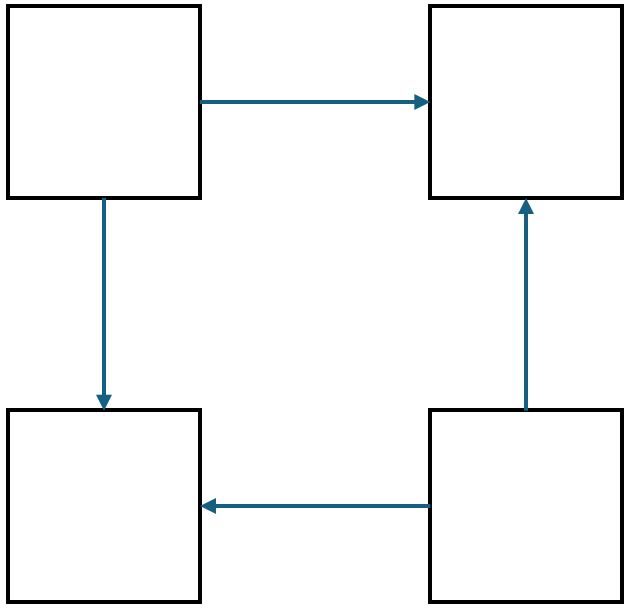
- O projeto detalhado do software
  - Projeto mais **baixo nível**, com granularidade **micro** do software, definindo seus objetivos e a forma de colaboraçao entre eles
  - **Satisfaz os requisitos funcionais.**





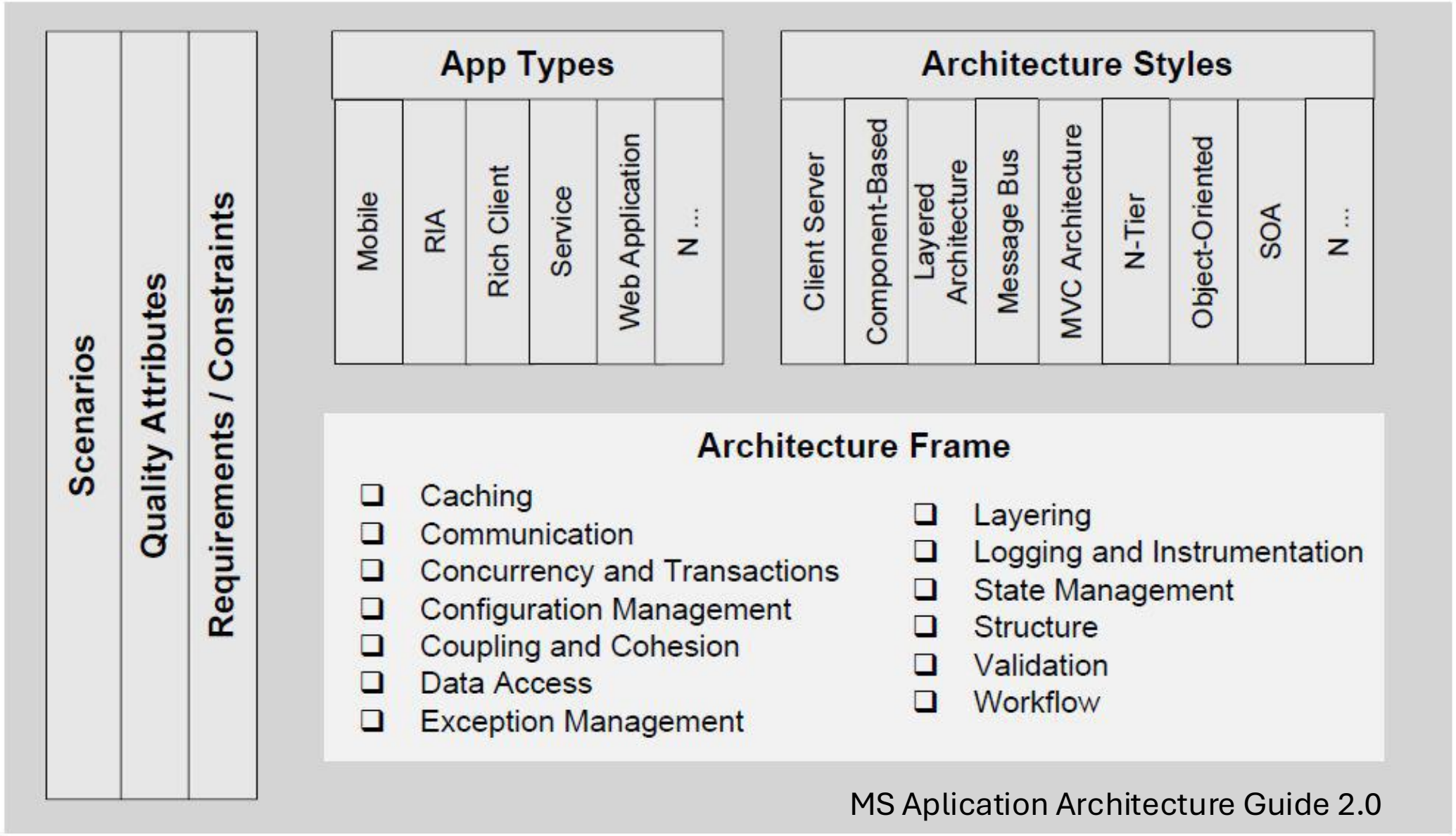
Como você definiria a arquitetura de um sistema?

Como você definiria a arquitetura de um sistema?



- Quais camadas você criaria?
- Onde ficaria a lógica do negócio?
- Como os dados seriam armazenados e acesados?
- Como os usuários interagiriam com o Sistema (UI/UX)?
- Que padrões ou estilos você adotaria (MVC, camadas, etc)?

Framework

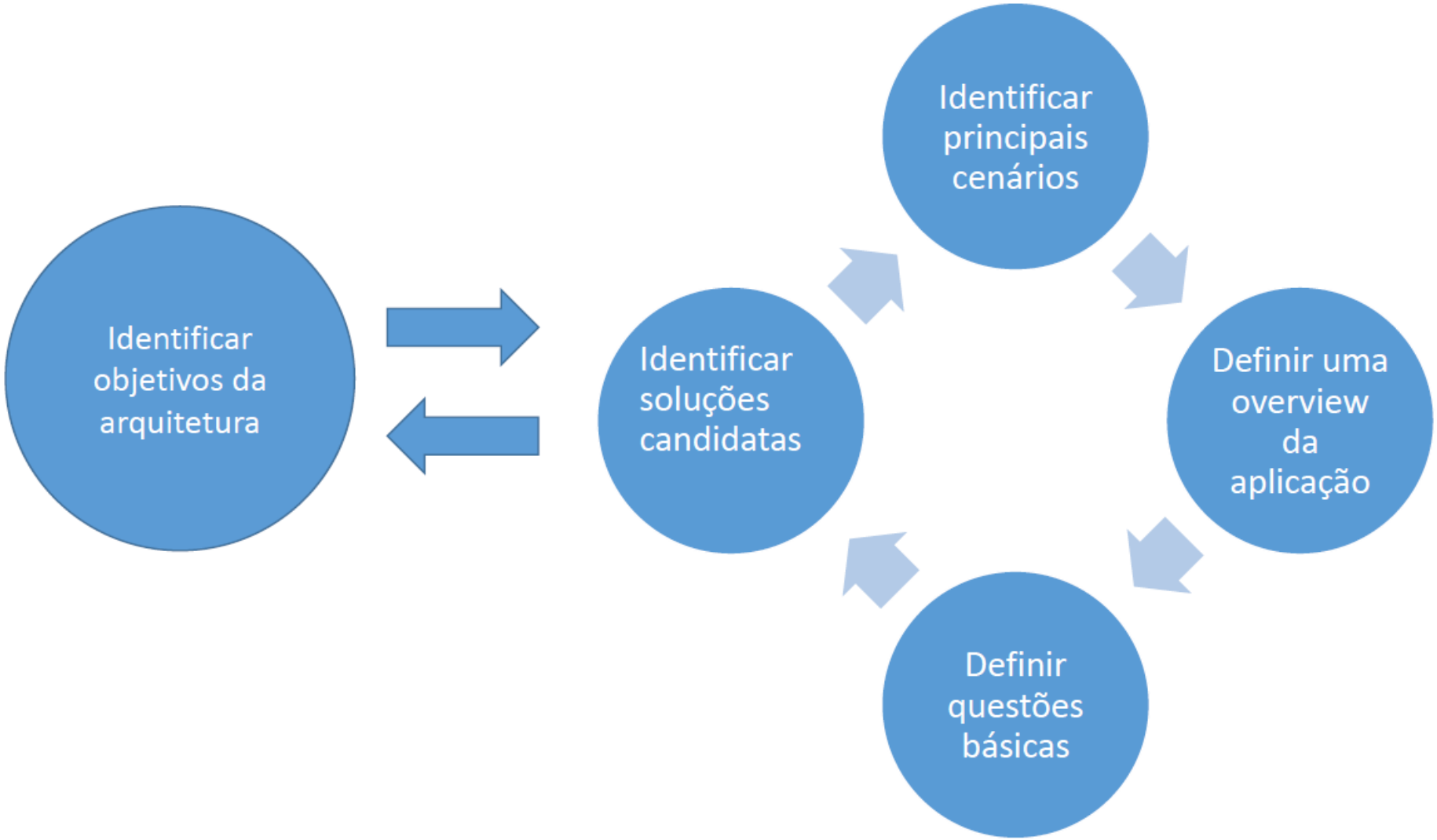


MS Aplication Architecture Guide 2.0

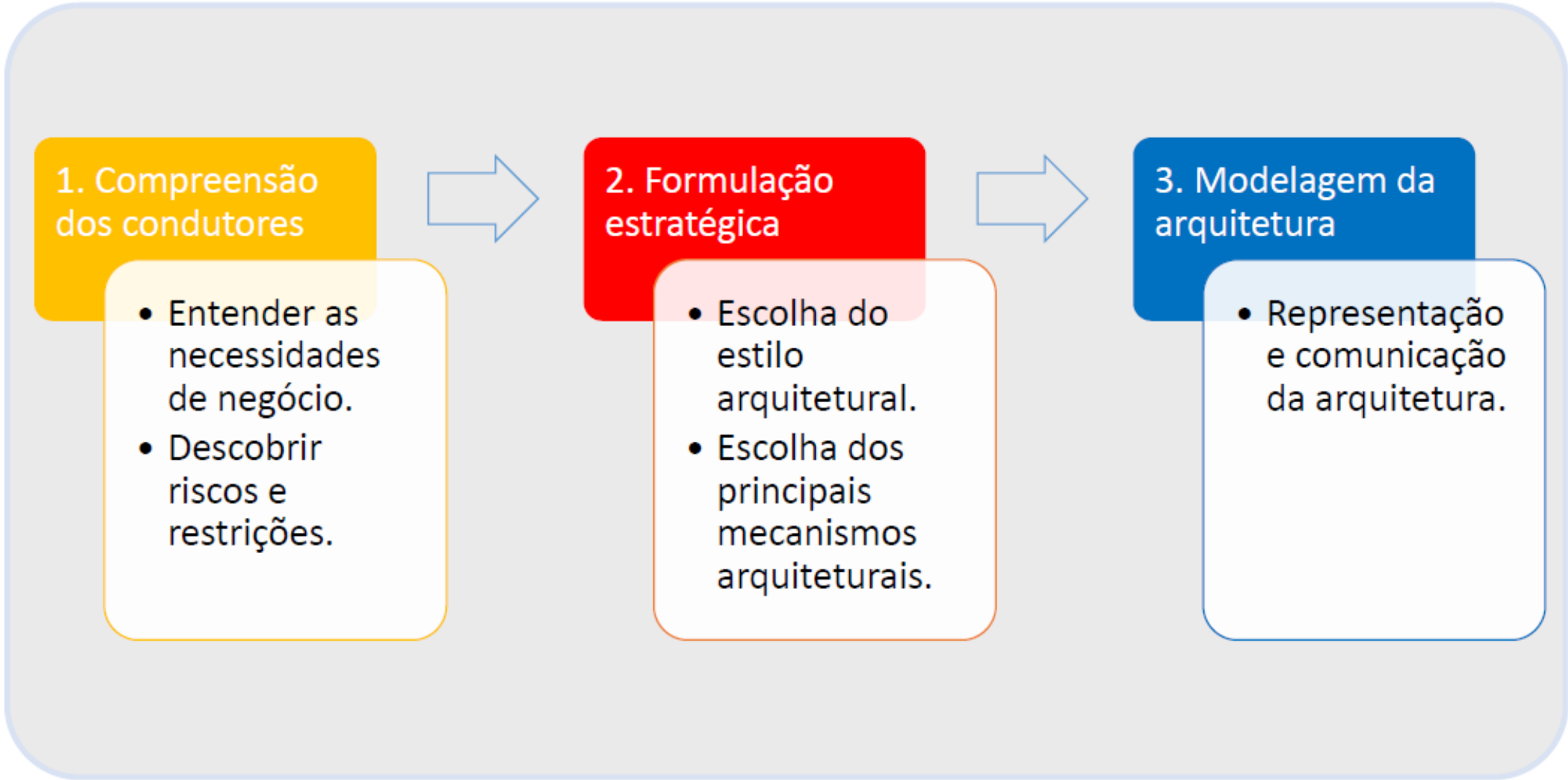
MS Aplication Architecture Guide

- Criado pela Microsoft patterns & practices (2009).
- Apoia arquitetos no processo de definição de software.
- Apresenta boas práticas e padrões (camadas, MVC, SOA, serviços).
- Foca em princípios de qualidade: desempenho, segurança, escalabilidade, testabilidade.
- Propõe um processo iterativo:
  - Identificar requisitos
  - Escolher estilos arquiteturais
  - Definir camadas e componentes
  - Validar trade-offs
  - Documentar e revisar

Ciclo Básico Para Definição da Arquitetura



Método Didático Para Definir uma Arquitetura



Etapa 1: O que é entender as necessidades de negócio?

Vamos ver uma analogia com o processo de compra de um novo automóvel

- Geralmente escolhe-se o carro com base em questões emocionais
- Nossas escolhas na arquitetura não podem ser dessa maneira

Abordagem adequada para comprar um carro









Arquitetura de Software - Definição

- Decisão estratégica: escolha de um carro.
- Antes de escolher um carro, deve-se compreender a real necessidade.
- É importante definir quais critérios irão conduzir a escolha

Arquitetura de Software - Definição

- Decisão estratégica: escolha de um carro.
- Antes de escolher um carro, deve-se compreender a real necessidade.
- É importante definir quais critérios irão conduzir a escolha.

Arquitetura de Software - Definição

Necessidades:

- “Uso carro para fins urbanos.”
- “Sou Casado e tenho 3 filhos.”
- “Faço pequenas viagens e minha esposa carrega muitas bagagens.”
- “Meu orçamento é limitado a 65 mil reais.”
- “Moro em uma cidade quente.”


Ka


FAS999

Uso carro para fins urbanos

Uso carro para fins urbanos


Ka

Sou casado e tenho 3 filhos

Sou casado e tenho 3 filhos

Faço pequenas viagens e minha mulher carrega muitas bagagens

Faço pequenas viagens e minha mulher carrega muitas bagagens

Meu orçamento é limitado a 65 mil reais

Meu orçamento é limitado a 65 mil reais

Decisões baseadas em escolhas racionais

- Podemos não ter gostado da escolha realizada, mas devemos admitir que foi uma escolha racional baseada em decisões lógicas.

Decisões baseadas em escolhas racionais

- Faltou alguma necessidade para analisarmos?

Decisões baseadas em escolhas racionais

- **“Moro em uma cidade quente”**
  - Carro com ar-condicionado

- Apesar de não ter definido a escolha do carro, é também uma **decisão relevante** a ser considerada

Arquitetura de Software - Definição

- Sem uma estratégia baseada em decisões racionais, podemos acabar com uma arquitetura inadequada ...

Necessidades - Software

- Quando vamos comprar um carro, pensamos em coisas como velocidade, espaço, segurança e custo.
- Na arquitetura de software, as **necessidades são análogas**, mas se traduzem em requisitos de desempenho, escalabilidade, segurança, integração e custo operacional.
- Quais necessidades nosso software pode ter?

Necessidades - Software

- **Desempenho:**
  - Quantos usuários simultâneos o sistema deve suportar?
  - Qual o tempo máximo de resposta aceitável para uma operação crítica?
- **Escalabilidade:**
  - O sistema precisa crescer para atender milhares/milhões de usuários?
  - Precisa funcionar em nuvem, em múltiplos servidores ou apenas localmente?
- **Disponibilidade:**
  - O sistema pode ficar indisponível para manutenção?
  - Necessidade de redundância ou alta disponibilidade (HA)?
- **Segurança:**
  - Que nível de proteção de dados é necessário?
  - Precisa de autenticação forte, criptografia, auditoria de acessos?
- **Integração:**
  - Com quais sistemas externos a aplicação deve conversar (APIs, ERPs, bancos de dados legados)?
- **Custo:**
  - Quais são os limites de orçamento para infraestrutura e manutenção?

Necessidades - Software

- Condutores arquiteturais: necessidades do negócio.
- Precisamos ter uma boa compreensão dos condutores.
- Necessidades do negócio:
  - Quais são as necessidades de negócio endereçadas pela arquitetura?
  - As mais importantes são condutores arquiteturais.

Condutores arquiteturais: Identificar riscos

Quais riscos realmente influenciam as decisões de arquitetura?

- Nem todos os riscos impactam a arquitetura.
- Os **mais críticos** se tornam **condutores arquiteturais**.
- São fatores que **obrigam o arquiteto** a tomar decisões específicas (ex: desempenho, segurança, disponibilidade).

Exemplo:

- Se a aplicação precisa rodar para milhões de usuários → desempenho se torna um condutor.
- Se lida com dados sensíveis → segurança se torna um condutor.

.

Condutores arquiteturais: Identificar restrições

Quais restrições realmente moldam a arquitetura?

- **Tecnológicas** → uso obrigatório de certa linguagem, framework ou banco de dados.
- **De projeto** → prazos curtos, equipe limitada, necessidade de integração com sistemas legados.
- **Corporativas** → políticas de compliance, custos de licenciamento, padrões internos da organização.

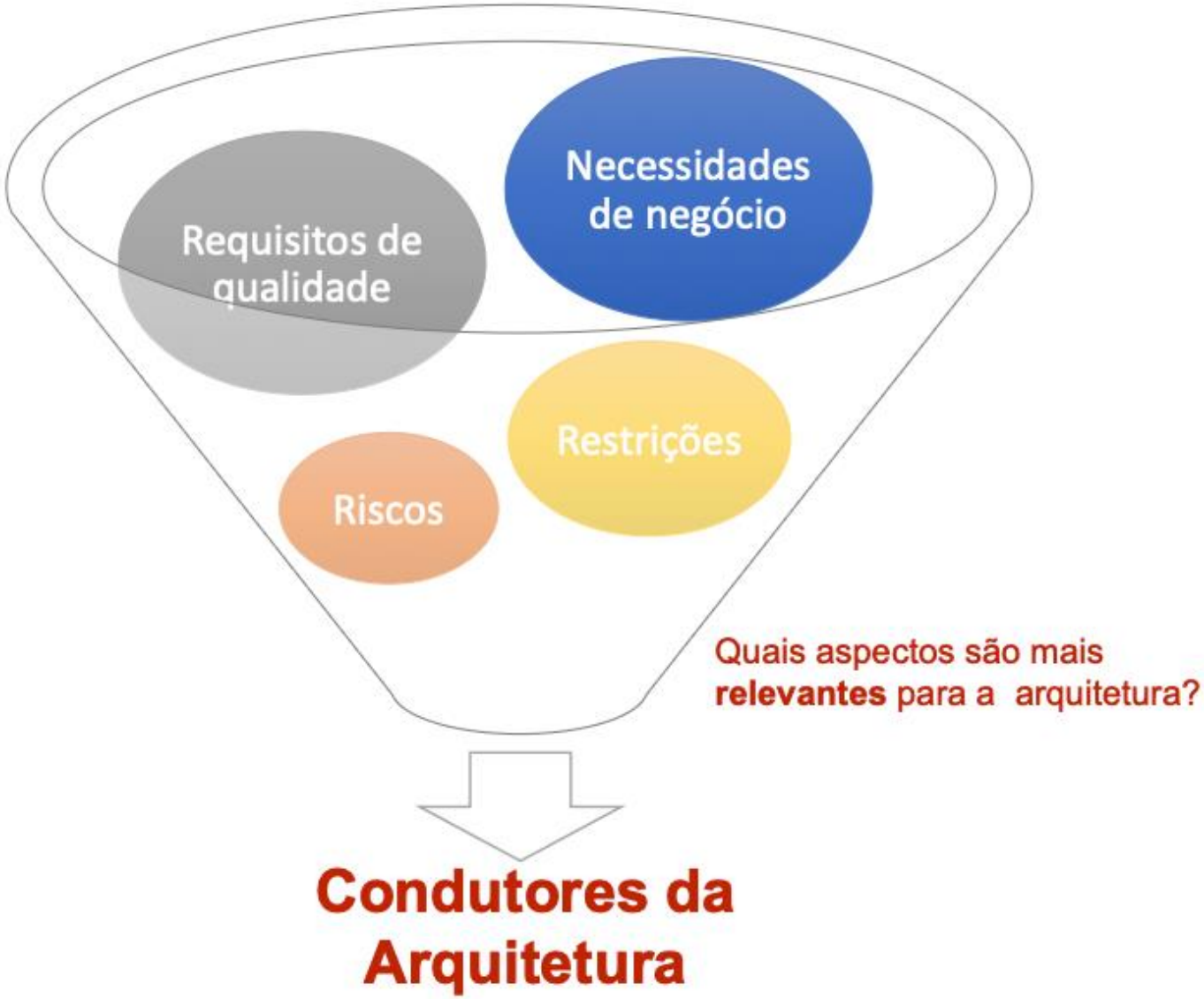
- Nem todas impactam a arquitetura.

- As **restrições mais críticas** tornam-se **condutores arquiteturais** → direcionam as decisões do arquiteto.

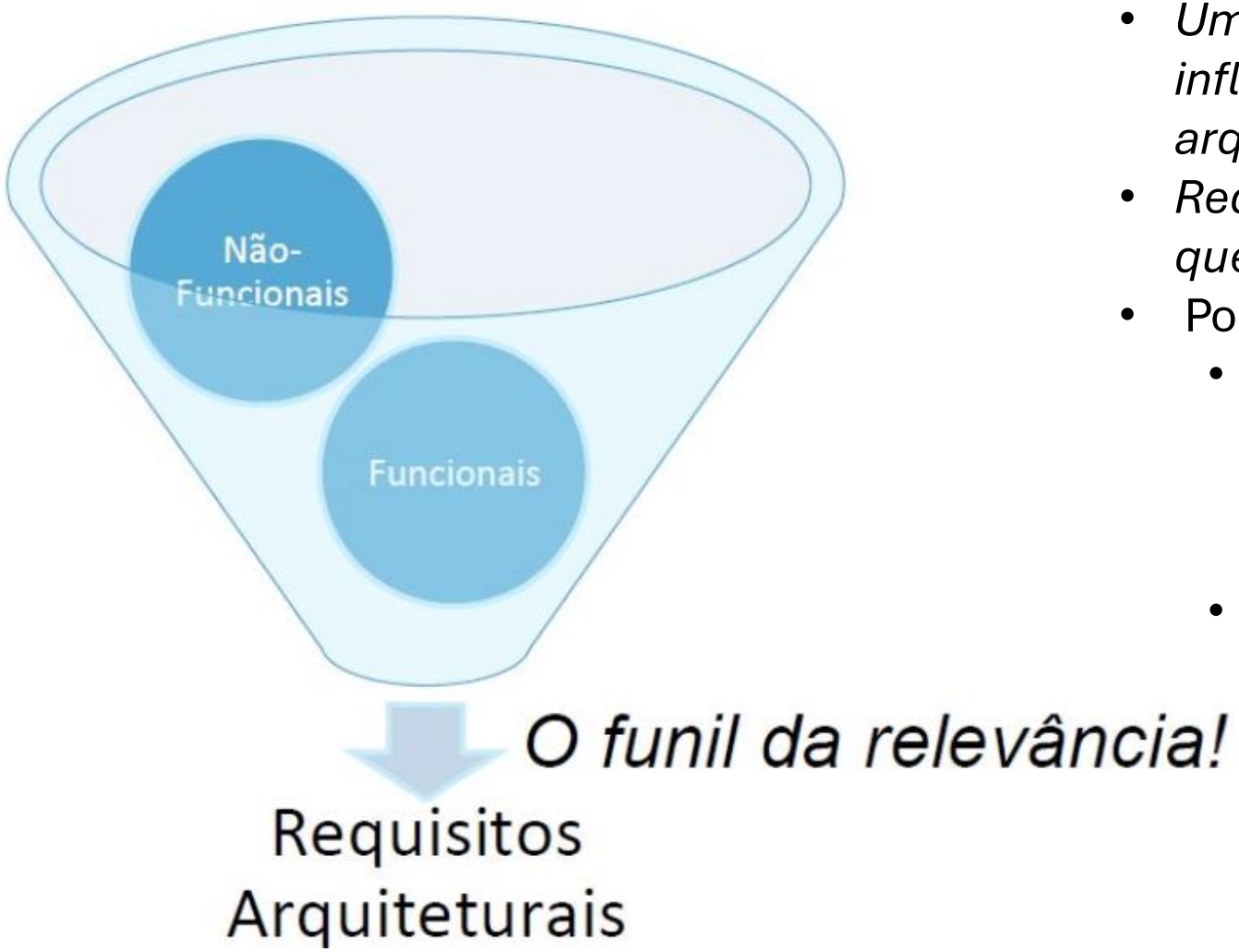
Requisitos de Qualidade

Exemplos de requisitos de qualidade impactados pela distribuição de um sistema.

Funil dos condutores arquiteturais



Funil dos condutores arquiteturais



- *Um requisito se torna **arquitetural** quando influencia diretamente as decisões de arquitetura.*
- *Requisitos são condições ou capacidades que o sistema deve atender.*
- Podem ser:
  - **Funcionais** → descrevem *o que* o sistema deve fazer.
    Ex.: registrar pedidos, autenticar usuários.
  - **Não Funcionais** → descrevem *como* o sistema deve se comportar.
    Ex.: tempo de resposta < 2s, 99,9% de disponibilidade, compatibilidade com dispositivos móveis.

Requisitos não arquiteturais

- Requisitos que não afetam significativamente as decisões de arquitetura
- **Funcionais (não arquiteturais):**
  - O sistema deve permitir cadastro de clientes com nome, CPF e endereço.
  - O usuário poderá atualizar sua senha no perfil.
  - O sistema deve enviar e-mails de confirmação após o registro.
  - O relatório de vendas deve incluir filtros por data e categoria.
- Perceba que esses requisitos descrevem funcionalidades específicas, mas **não mudam a estrutura ou as decisões centrais da arquitetura**. Eles podem ser implementados independentemente da tecnologia base ou da forma como os componentes estão organizados.
- **Não Funcionais (não arquiteturais):**
  - O sistema deve ter a interface em português e inglês.
  - O logotipo da empresa deve aparecer em todas as páginas.
  - A senha deve ter pelo menos 8 caracteres.
  - O sistema deve utilizar cores da identidade visual da empresa.
- Aqui também temos **condições importantes**, mas que **não determinam grandes decisões arquiteturais**.

Quando a arquitetura não atende às necessidades reais ...

Caso 1

- Uma equipe de TI no governo descobriu na semana da entrega que o envio de email não funcionava devido a **topologia física desconhecida e restrições de segurança (Certificados digitais e HTTPS)**

Caso 2

- Uma outra equipe teve que refazer toda a camada visual de Javascript ao perceber em produção que **o sistema não atendia o Firefox**

Caso 3

- Uma terceira equipe de uma grande empresa de minas descobriu em produção que o tempo médio dos **casos de uso era de 40 segundos**

Caso 4

- Uma equipe descobriu que a aplicação **tratava o banco de dados em produção periodicamente (deadlocks) e parava toda o lançamento de notas de uma universidade.**

Quando a arquitetura não atende às necessidades reais ...

Caso 1

- Uma equipe de TI no governo descobriu na semana da entrega que o envio de email não funcionava devido a **topologia física desconhecida e restrições de segurança (Certificados digitais e HTTPS)**

- Uma outra equipe teve que refazer toda a camada visual de

Ignorar condutores arquiteturais cedo custa caro mais tarde!

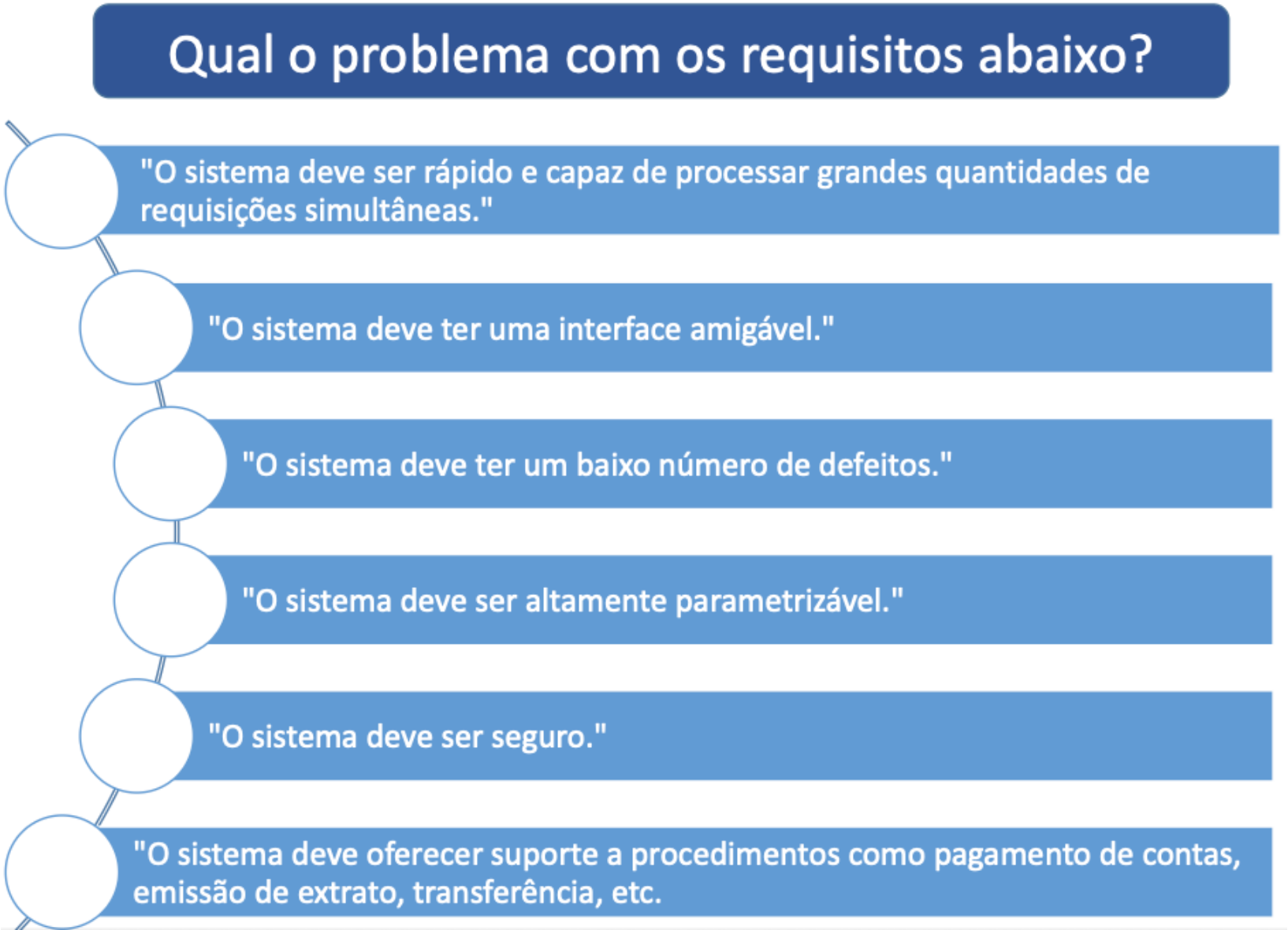
Caso 3

- Uma terceira equipe de uma grande empresa de minas descobriu em produção que o tempo médio dos **casos de uso era de 40 segundos**

Caso 4

- Uma equipe descobriu que a aplicação **tratava o banco de dados em produção periodicamente (deadlocks) e parava toda o lançamento de notas de uma universidade.**

Qualidade dos Requisitos



Qualidade dos Requisitos

Requisitos tolos!

- **Utilização de termos genéricos:**
  - Geralmente, usualmente, frequentemente, normalmente, tipicamente, aproximadamente.
- **Termos relativos:**
  - Rápido, flexível, intuitivo, amigável, grande número de requisições, baixo tempo de resposta, etc.
- **Termos furtivos:**
  - Poderia, deveria, talvez, pode, provavelmente, etc.

Requisitos “Espertos” – Princípio SMART

- Princípio SMART
  - **Specific (Específicos), Mensurable (Mensurável), Attainable (Atingível), Realizable (Realizável), Traceable (Rastreável).**

- O princípio SMART permite que um requisito seja escrito com menor ambiguidade.
  - Exemplo, considere o requisito “**A interface do sistema deve ser amigável**”.
  - O uso de palavras genéricas como sistema e advérbios como amigável não transmite certeza sobre como devemos resolver este problema e implementar o código.

Requisitos “Espertos” – Princípio SMART

- Requisitos SMART são aqueles que atendem a cinco critérios:

1- ***Specific*** (Específicos).

- Um requisito deve ser específico. Exemplos de requisitos específicos indicam o caso de uso, tela ou módulo a que eles se referem.

- Como exemplo do requisito fornecido anteriormente poderíamos dizer que a **interface da tela de cadastro de estudantes deve ser amigável**. O requisito foi tornado específico.

Requisitos “Espertos” – Princípio SMART

2- ***Mensurable (***Mensurável***).***

- **Todo requisito arquitetural deve ser mensurável.**
- É preciso definir critérios claros que permitam **verificar objetivamente** se o requisito foi atendido.
- Medidas evitam ambiguidades e permitem validação/teste.

- Exemplo: “*Um usuário novato deve conseguir operar a tela de cadastro de aluno após um treinamento de no máximo* ***30 minutos****, cometendo no máximo* ***1 erro ou advertência*** *fornecido pelo sistema*.”

Requisitos “Espertos” – Princípio SMART

2- ***Mensurable (***Mensurável***).***

- **Todo requisito arquitetural deve ser mensurável.**
- É preciso definir critérios claros que permitam **verificar objetivamente** se o requisito foi atendido.
- Medidas evitam ambiguidades e permitem validação/teste.

- Exemplo: “*Um usuário novato deve conseguir operar a tela de cadastro de aluno após um treinamento de no máximo* ***30 minutos****, cometendo no máximo* ***1 erro ou advertência*** *fornecido pelo sistema*.”

Requisitos “Espertos” – Princípio SMART

3- *Attainable* **(Atingível)**

- **Um requisito arquitetural deve ser tecnicamente viável.**
- Nem tudo que parece desejável é realmente possível.
- É preciso considerar **limitações tecnológicas, físicas e humanas**.

Exemplo de contraexemplo:

- *Exigir que a tela de cadastro de alunos tenha **100% de disponibilidade*** → impossível, já que **todo sistema depende de máquinas sujeitas a falhas** (MTBF – Mean Time Between Failures).
- Disponibilidade de **99,9%** no período de 1 ano. (BOM REQUISITO).

Requisitos “Espertos” – Princípio SMART

4 - *Realizable* **(Realizável)**

- **Nem todo requisito atingível é realizável no contexto do projeto**
- Um requisito é **realista** apenas se considerar:
  - **Recursos do time** (tamanho e competências)
  - **Tempo disponível**
  - **Orçamento e infraestrutura**

Exemplo:

  - Exigir que a tela de cadastro opere em ambiente tolerante a falhas — Se não houver **orçamento para máquinas redundantes, esse requisito não é realizável**.
- **Se não for realizável** → deve ser **relaxado ou ajustado** para se adequar às condições reais do projeto.

Requisitos “Espertos” – Princípio SMART

5 - *Traceable* **(Rastreável)**

- Estudos do Standish Group mostram que quase 50% dos requisitos de softwares em produção nunca são usados ou são raramente utilizados.
- Uma das causas é a inclusão de requisitos de forma desorganizada, às vezes pelos próprios desenvolvedores, sem aprovação dos usuários.
- Por isso, é essencial conhecer a **origem de cada requisito**.
- Todo requisito deve ser **rastreável**, permitindo identificar quem solicitou, por que foi incluído e como será validado.

Requisitos “Espertos” – Princípio SMART

Exemplo:

- Requisito ambíguo: “O sistema deve ser **rápido** e capaz de processar **grandes quantidades** de requisições simultâneas.”

- Requisito SMART: “A **tela de cadastro** de usuários deve possuir um tempo de resposta **menor que 8 segundos** e suportar 20 usuários simultâneos em horários de pico (15:00 às 19:00). [Requisito elicitado e aprovado com usuário Paulo Silva]”

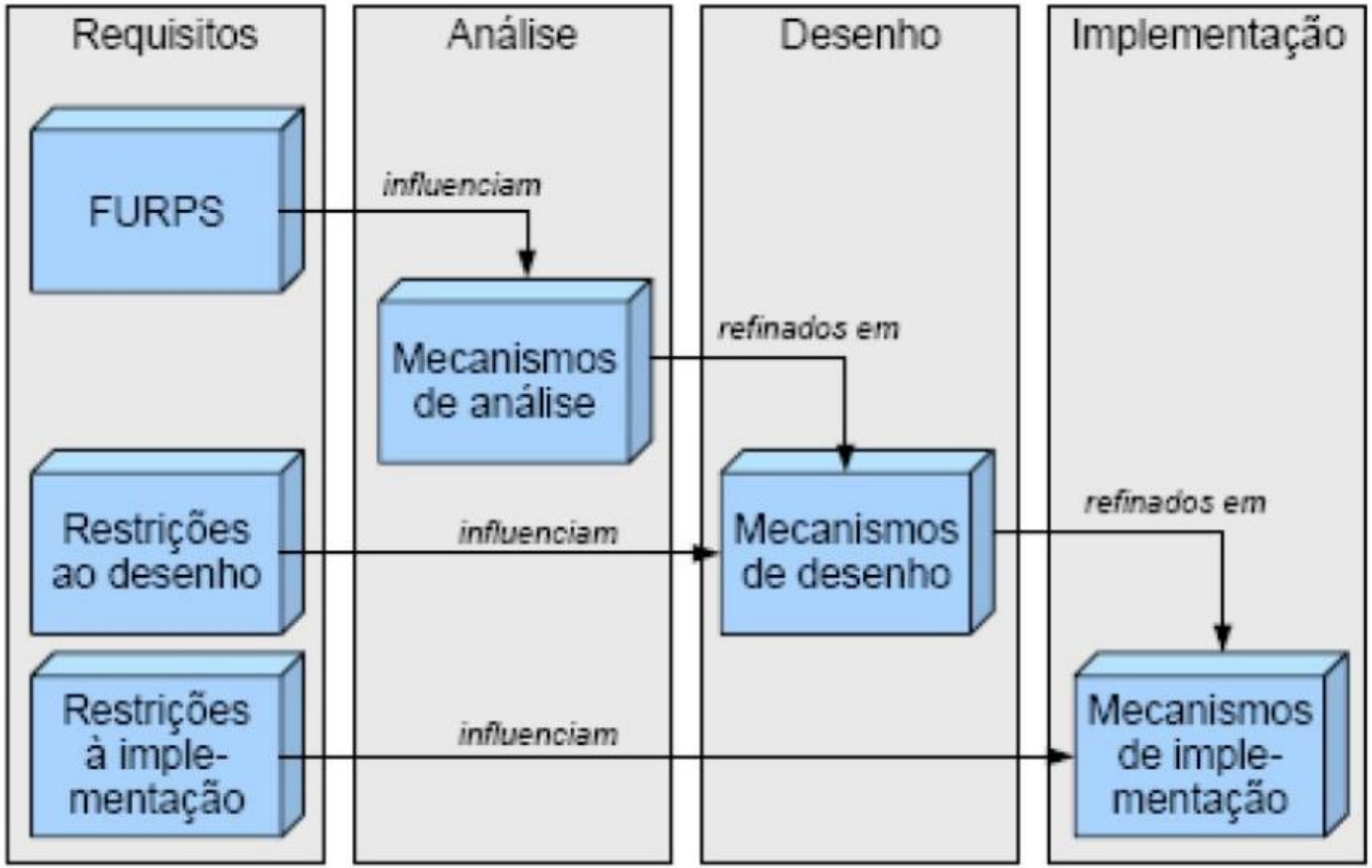
Formulação Estratégica: Escolha de Tecnologias

- Como escolher tecnologias de uma arquitetura?

Formulação Estratégica: Escolha de Tecnologias

- Um **mecanismo arquitetural** representa uma solução recorrente para problemas comuns.
- **Benefícios na definição de mecanismos de arquitetura:**
  - Destacam explicitamente aspectos da solução que se repetem em todo o sistema.
  - Facilitam o planejamento do desenvolvimento.
  - Permitem que desenvolvedores construam componentes uma vez e **os reutilizem**, reduzindo esforço e trabalho duplicado.
- Um mecanismo arquitetural pode existir em três estados: **análise**, **design** e **implementação**.

Formulação Estratégica: Escolha de Tecnologias



Um exemplo complexo - segurança

- **Requisitos Arquiteturais**
  - Os usuários devem ser **corretamente identificados**.
- **Restrições**
  - A solução de segurança deve adotar **padrões abertos**.
  - **Senhas não podem trafegar em texto claro** pela rede.
  - A empresa já utiliza **soluções Microsoft** (compatibilidade necessária).

Um exemplo complexo - segurança

Mecanismo de Análise

- **Como os projetistas analisam?**
  - Estudam **materiais técnicos** sobre mecanismos de **autenticação**.
  - Avaliam **opções técnicas**: senha, autenticação de dois fatores (2FA), biometria etc.
  - **Autenticação** é a solução técnica que garante que apenas **usuários autorizados** tenham acesso ao sistema.

Um exemplo complexo - segurança

Opções técnicas avaliadas:

- **Kerberos** e **LDAP** elencados como soluções possíveis, ambos **padrões abertos**.

- **Decisão arquitetural**
  - **Kerberos** é escolhido, pois **não trafega senhas em texto pela rede**, aumentando a segurança contra ataques de interceptação.

Um exemplo complexo - segurança

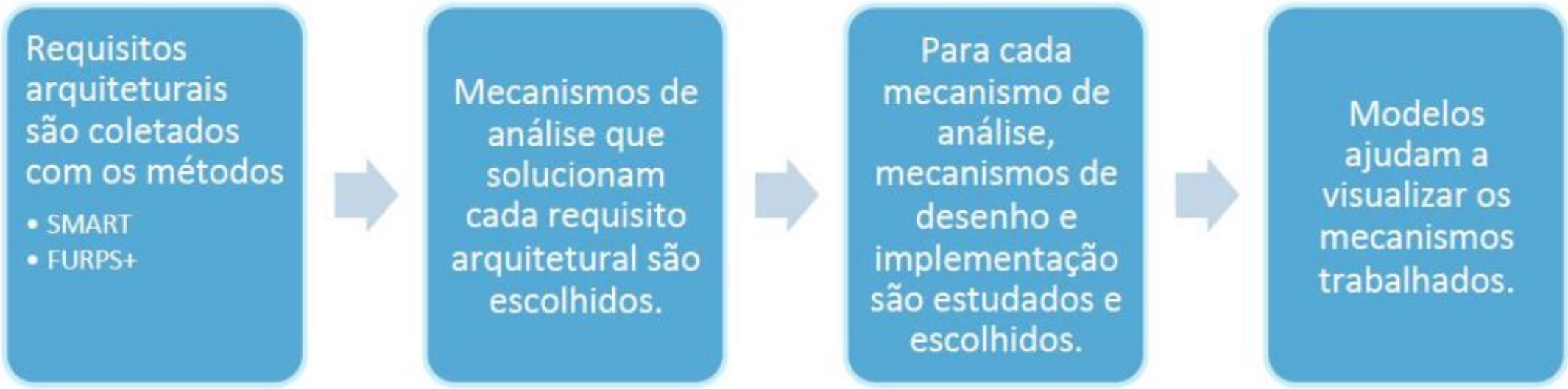
Mecanismo de Implementação

- **Contexto organizacional**
  - A empresa já utiliza **produtos Microsoft** no seu ecossistema.

- **Decisão de implementação**
  - Foi escolhido o **Active Directory**, que **suporta o protocolo Kerberos** de forma nativa.

Mecanismo de Análise	Mecanismo de Design	Mecanismo de Implementação
Front-End	Interface com Usuário	Bootstrap, Javascript, HTML5 , CSS
ESB	Integração entre serviços	MuleESB
Message Broker	Troca de mensagens	RabbitMQ
Comunicação entre processos	Framework SOA	WCF
Comunicação entre processos	Framework SOA	WebAPI
Persistência	Banco de dados relacional	Microsoft SQL Server
Persistência	Framework ORM	Entity Framework
Versionamento	Versionamento dos fontes da aplicação.	Visual Studio Team Foundation Services
Alta Disponibilidade	Load Balance de Serviços	Network Load Balancing
Alta Disponibilidade	Load Balance de ESB	MuleESB Cluster
Alta Disponibilidade	Load Balance de ESB	Microsoft SQL Server AlwaysOn

Processo de Modelagem Resumidos



Modelagem da Arquitetura

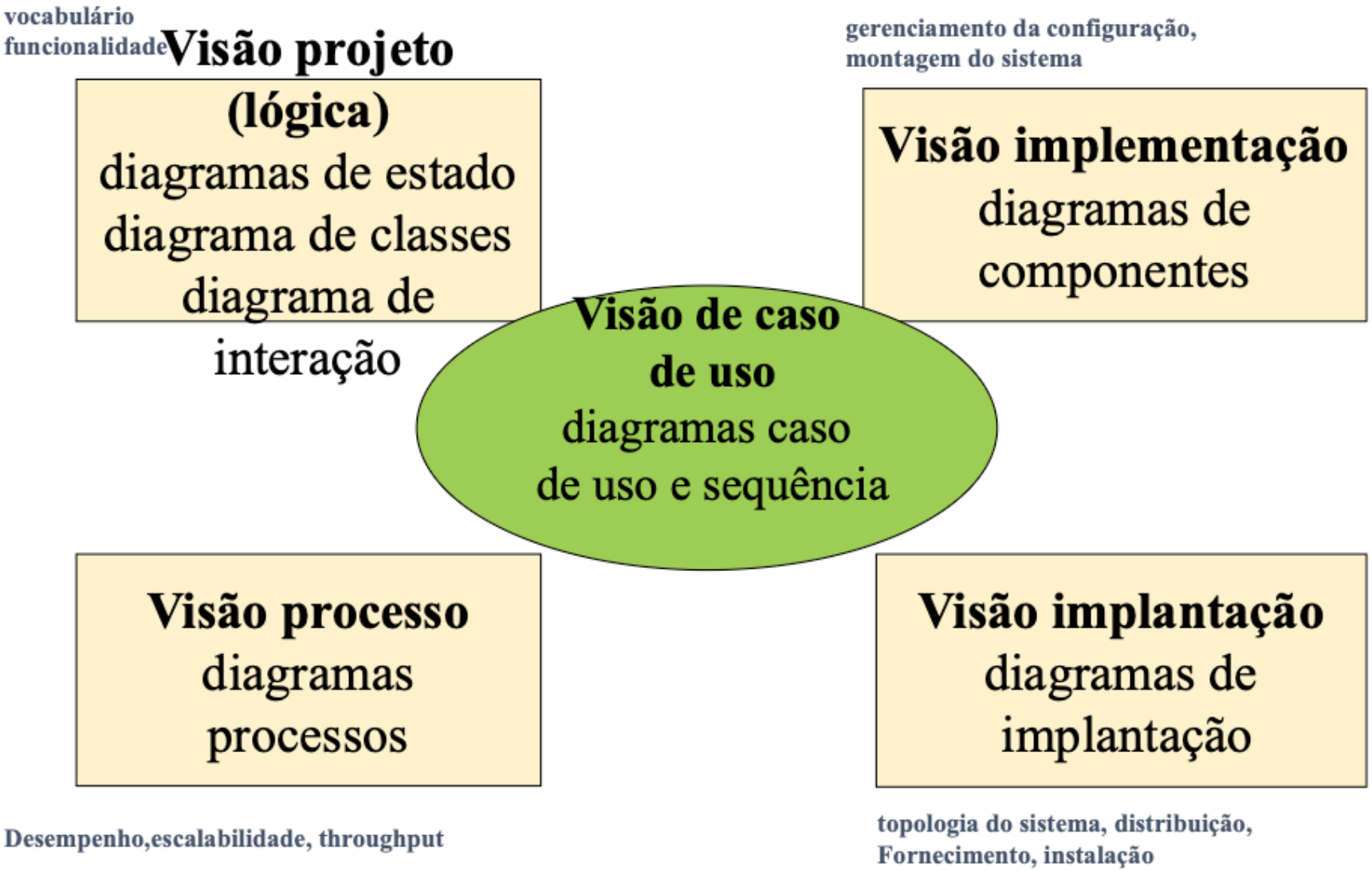
Para que modelar?

- **Comunicar:** Facilitar o entendimento entre times e stakeholders.
- **Validar:** Testar e analisar a arquitetura antes da implementação.
- **Documentar:** Criar um "mapa" do sistema para referência futura.

Como representamos?

- Diagramas de componentes e de fluxo.
- Modelos de dados e tabelas.
- Descrições textuais detalhadas.

Modelagem da Arquitetura



Modelagem da Arquitetura