

---

# Fundamentos da Arquitetura de Software

*Uma abordagem de engenharia*

*Mark Richards e  
Neal Ford*

AMOSTRA



ALTA BOOKS

GRUPO EDITORIAL

Rio de Janeiro, 2024

---

# Sumário

<b>Prefácio: Invalidando Axiomas .....</b>	<b>xiii</b>
<b>1. Introdução .....</b>	<b>1</b>
Definindo Arquitetura de Software	3
Expectativas de um Arquiteto	7
Tomar Decisões de Arquitetura	8
Analisar Continuamente a Arquitetura	8
Manter-se Atualizado com as Últimas Tendências	9
Assegurar a Conformidade com as Decisões	9
Exposição e Experiência Diversificadas	10
Ter Conhecimento sobre o Domínio do Negócio	10
Ter Habilidades Interpessoais	11
Entender e Lidar Bem com Questões Políticas	11
Interseção da Arquitetura e...	13
Práticas da Engenharia	14
Operações/DevOps	18
Processo	18
Dados	19
Leis da Arquitetura de Software	20

---

## Parte I. Fundamentos

<b>2. Pensamento Arquitetônico .....</b>	<b>23</b>
Arquitetura Versus Design	23
Amplitude Técnica	25
Analisando os Trade-offs	30
Entendendo as Motivações Comerciais	34
Equilibrando Arquitetura e Codificação	34

<b>3. Modularidade .....</b>	<b>37</b>
Definição	38
Medindo a Modularidade	40
Coesão	40
Acoplamento	44
Abstração, Instabilidade e Distância da Sequência Principal	45
Distância da Sequência Principal	46
Consciência	49
Unificando as Métricas de Acoplamento e Consciência	53
De Módulos a Componentes	54
<b>4. Definição das Características da Arquitetura .....</b>	<b>55</b>
Características da Arquitetura Listadas (em Parte)	58
Características Operacionais da Arquitetura	58
Características Estruturais da Arquitetura	59
Características Transversais da Arquitetura	59
Trade-offs e Arquitetura Menos Pior	64
<b>5. Identificando as Características da Arquitetura .....</b>	<b>65</b>
Extraindo Características da Arquitetura das Preocupações do Domínio	65
Extraindo Características da Arquitetura dos Requisitos	68
Estudo de Caso: Silicon Sandwiches	69
Características Explícitas	70
Características Implícitas	74
<b>6. Medindo e Controlando as Características da Arquitetura .....</b>	<b>77</b>
Medindo as Características da Arquitetura	77
Medidas Operacionais	78
Medidas Estruturais	79
Medidas do Processo	82
Governança e Funções de Aptidão	82
Governando as Características da Arquitetura	83
Funções de Aptidão	83
<b>7. Escopo das Características da Arquitetura .....</b>	<b>91</b>
Acoplamento e Consciência	92
Quanta Arquitetural e Granularidade	92
Estudo de Caso: Going, Going, Gone	95

<b>8. Pensamento Baseado em Componentes .....</b>	<b>101</b>
Escopo do Componente	101
Função do Arquiteto	103
Particionamento da Arquitetura	104
Estudo de Caso: Silicon Sandwiches: Particionamento	108
Função do Desenvolvedor	110
Fluxo de Identificação dos Componentes	111
Identificando os Componentes Iniciais	111
Atribuir Requisitos aos Componentes	112
Analisar Funções e Responsabilidades	112
Analisar as Características da Arquitetura	112
Reestruturar os Componentes	112
Granularidade do Componente	113
Design do Componente	113
Descobrimdo os Componentes	113
Estudo de Caso: Going, Going, Gone: Descobrimdo os Componentes	116
Quantum da Arquitetura Revisitado: Escolhendo Entre Arquiteturas Monolíticas Versus Distribuídas	119

---

## **Parte II. Estilos de Arquitetura**

<b>9. Fundamentos.....</b>	<b>123</b>
Padrões Fundamentais	123
A Grande Bola de Lama	124
Arquitetura Unitária	125
Cliente/Servidor	126
Arquiteturas Monolíticas Versus Distribuídas	128
Falácia 1: A Rede É Confiável	129
Falácia 2: A Latência É Zero	129
Falácia 3: A Largura de Banda É Infinita	130
Falácia 4: A Rede É Segura	132
Falácia 5: A Topologia Nunca Muda	132
Falácia 6: Existe Apenas Um Administrador	133
Falácia 7: O Custo do Transporte É Zero	134
Falácia 8: A Rede É Homogênea	134
Outras Considerações Distribuídas	135

<b>10. Estilo de Arquitetura em Camadas .....</b>	<b>137</b>
Topologia	137
Camadas de Isolamento	139
Adicionando Camadas	141
Outras Considerações	142
Por que Usar Esse Estilo de Arquitetura	143
Classificações das Características da Arquitetura	144
<b>11. Estilo de Arquitetura Pipeline .....</b>	<b>147</b>
Topologia	147
Canais	148
Filtros	148
Exemplo	149
Classificações das Características da Arquitetura	151
<b>12. Estilo de Arquitetura Microkernel .....</b>	<b>153</b>
Topologia	153
Sistema Central	154
Componentes de Plug-in	157
Registro	161
Contratos	162
Exemplos e Casos de Uso	163
Classificações das Características da Arquitetura	164
<b>13. Estilo de Arquitetura Baseada em Serviços .....</b>	<b>167</b>
Topologia	167
Variantes da Topologia	169
Design do Serviço e Granularidade	171
Particionamento do Banco de Dados	173
Arquitetura de Exemplo	176
Classificações das Características da Arquitetura	178
Quando Usar Esse Estilo de Arquitetura	181
<b>14. Estilo de Arquitetura Orientada a Eventos .....</b>	<b>183</b>
Topologia	184
Topologia do broker	184
Topologia do Mediador	190
Capacidades Assíncronas	200

Tratamento de Erro	201
Evitando a Perda de Dados	205
Capacidades de Transmissão	207
Requisição-Resposta	208
Escolhendo Entre Modelo Baseado em Requisição e Orientado a eventos	211
Arquiteturas Híbridas Orientadas a Eventos	212
Classificações das Características da Arquitetura	213
<b>15. Estilo de Arquitetura Baseada em Espaço .....</b>	<b>217</b>
Topologia Geral	218
Unidade de Processamento	219
Middleware Virtualizado	220
Data Pumps	225
Gravações de Dados	227
Leituras de Dados	228
Colisões de Dados	230
Implementações na Nuvem Versus Locais	233
Cache Replicado Versus Distribuído	234
Observações sobre o Near-Cache	236
Exemplos de Implementação	237
Sistema de Ingressos para Concertos	238
Sistema de Leilão Online	238
Classificações das Características da Arquitetura	239
<b>16. Arquitetura Orientada a Serviços e Baseada em Orquestração .....</b>	<b>241</b>
História e Filosofia	241
Topologia	242
Taxonomia	243
Serviços do Negócio	243
Serviços Corporativos	243
Serviços do Aplicativo	244
Serviços da Infraestrutura	244
Mecanismo de Orquestração	244
Fluxo das Mensagens	245
Reutilização... e Acoplamento	245
Classificações das Características da Arquitetura	248

<b>17. Arquitetura de Microserviços .....</b>	<b>251</b>
História	251
Topologia	252
Distribuída	253
Contexto Delimitado	254
Granularidade	254
Isolamento dos Dados	255
Camada da API	256
Reutilização Operacional	256
Front-ends	259
Comunicação	260
Coreografia e Orquestração	262
Transações e Sagas	265
Classificações das Características da Arquitetura	268
Referências Adicionais	270
<b>18. Escolhendo o Estilo de Arquitetura Certo .....</b>	<b>271</b>
A Mudança de “Moda” na Arquitetura	271
Critérios de Decisão	273
Estudo de Caso Monolítico: Silicon Sandwiches	275
Monolítico Modular	276
Microkernel	277
Estudo de Caso Distribuído: Going, Going, Gone	278

---

## Parte III. Técnicas e Habilidades Sociais

<b>19. Decisões da Arquitetura .....</b>	<b>285</b>
Antipadrões da Decisão de Arquitetura	285
Antipadrão Cobertura dos Ativos	286
Antipadrão Feitiço do Tempo	286
Antipadrão Arquitetura Baseada em E-mail	287
Significante para a Arquitetura	288
Registros de Decisão da Arquitetura	289
Estrutura Básica	290
Armazenando ADRs	296
ADRs como Documentação	298
Usando ADRs para Padrões	298
Exemplo	299

<b>20. Analisando o Risco da Arquitetura .....</b>	<b>301</b>
Matriz de Risco	301
Avaliações do Risco	302
Risk Storming	306
Identificação	308
Consenso	308
Análise de Risco da História com a Metodologia Ágil	313
Exemplos de Risk Storming	313
Disponibilidade	315
Elasticidade	317
Segurança	319
<b>21. Diagramando e Apresentando a Arquitetura .....</b>	<b>321</b>
Diagramando	322
Ferramentas	322
Padrões da Diagramação: UML, C4 e ArchiMate	324
Diretrizes do Diagrama	325
Apresentação	327
Manipulando o Tempo	328
Construções Incrementais	328
Infodecks Versus Apresentações	331
Os Slides São Metade da História	331
Invisibilidade	331
<b>22. Tornando as Equipes Eficientes .....</b>	<b>333</b>
Limites da Equipe	333
Personalidades do Arquiteto	334
Controlador	335
Arquiteto Folgado	336
Arquiteto Eficiente	338
Quanto Controle?	339
Sinais de Aviso da Equipe	343
Utilizando Checklists	346
Checklist da Completude de código do desenvolvedor	348
Checklist dos Testes Unitários e Funcionais	350
Checklist de Release do Software	350
Dando Orientação	351
Resumo	354



<b>23. Habilidades de Negociação e Liderança .....</b>	<b>355</b>
Negociação e Facilitação	355
Negociando com os Acionistas da Empresa	356
Negociando com Outros Arquitetos	358
Negociando com os Desenvolvedores	360
Arquiteto de Software como um Líder	362
Os 4 Cs da Arquitetura	362
Seja Pragmático, Mas Visionário	364
Liderando as Equipes Dando o Exemplo	366
Integração com a Equipe de Desenvolvimento	370
Resumo	374
<b>24. Desenvolvendo uma Trajetória Profissional .....</b>	<b>375</b>
Regra dos 20 Minutos	375
Desenvolvendo um Radar Pessoal	377
Radar da Tecnologia da ThoughtWorks	377
Partes de Visualização Open Source	381
Usando a Rede Social	381
Conselhos de Despedida	383
<b>Apêndice: Perguntas de Autoavaliação .....</b>	<b>385</b>
<b>Índice .....</b>	<b>395</b>

# CAPÍTULO 1

---

## Introdução

A função de “arquiteto de software” aparece no topo de várias listas das melhores profissões no mundo. No entanto, quando os leitores veem *outras* profissões nessas listas (como profissional de enfermagem ou gerente financeiro), há uma trajetória profissional clara para eles. Por que não existe uma trajetória para os arquitetos de software?

Primeiro, o setor não tem uma boa definição própria para arquitetura de software. Quando damos aulas de nível básico, os alunos costumam pedir uma definição concisa do que faz um arquiteto de software e nos recusamos categoricamente a fazer isso, e não somos os únicos. No famoso documento técnico “Who Needs an Architect?”, Martin Fowler ficou conhecido por se recusar a definir a função, remetendo à famosa citação:

Arquitetura é sobre algo importante... seja lá o que for.

— Ralph Johnson

Quando pressionados, criamos o mapa mental mostrado na Figura 1-1, que lamentavelmente está incompleto, mas é um indicativo do escopo da arquitetura de software. De fato, daremos nossa definição concisa de arquitetura de software.

Segundo, como o mapa mental mostra, a função do arquiteto de software incorpora uma enorme quantidade e domínio de responsabilidade que continua a se expandir. Há uma década, os arquitetos de software lidavam apenas com os aspectos puramente técnicos da arquitetura, como modularidade, componentes e padrões. Desde então, devido aos novos estilos de arquitetura que aproveitam uma faixa maior de capacidades (como microsserviços), a função do arquiteto de software foi ampliada. Cobrimos as diversas interseções da arquitetura e do resto da organização na seção “Interseção da Arquitetura e...”.

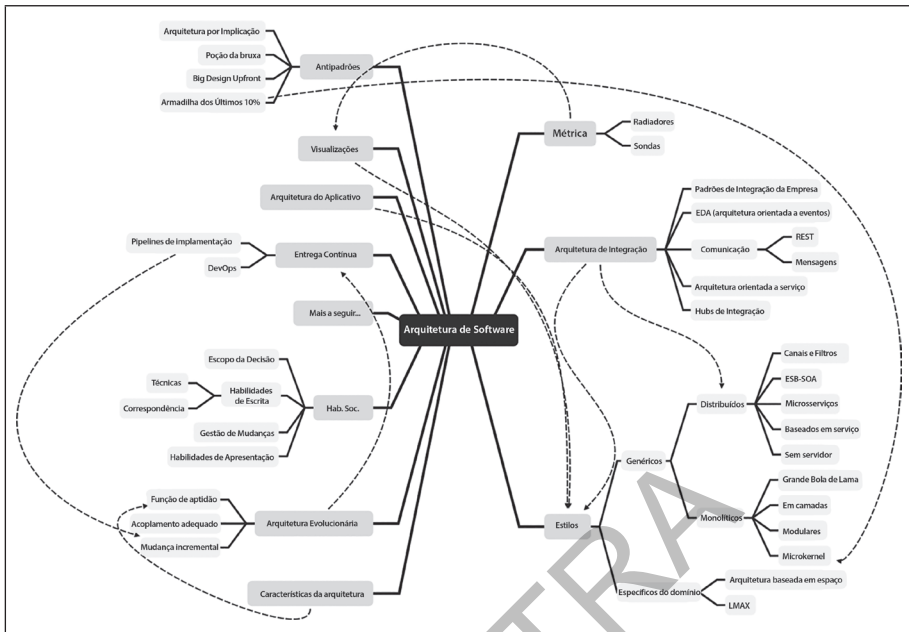


Figura 1-1. As responsabilidades de um arquiteto de software incluem habilidades técnicas, sociais, consciência operacional e muitas outras

Terceiro, o arquiteto de software é um alvo sempre móvel por causa da evolução rápida do ecossistema de desenvolvimento de software. Qualquer definição dada hoje ficará irremediavelmente desatualizada em poucos anos. A definição na Wikipédia de arquitetura de software fornece uma visão geral razoável, mas muitas afirmações estão desatualizadas, como “Arquitetura de software significa fazer escolhas estruturais fundamentais cuja mudança custa caro após a implementação”. Todavia, os arquitetos projetaram estilos arquiteturais modernos, como microsserviços, com a ideia de desenvolvimento incremental; mudanças estruturais nos microsserviços agora não custam caro. Claro, essa capacidade implica decisões difíceis envolvendo preocupações, como o acoplamento. Muitos livros sobre arquitetura de software a tratam como um problema estático; assim que resolvida, pode ser tranquilamente ignorada. Porém, neste livro reconhecemos a natureza dinâmica inerente da arquitetura de software, inclusive a definição em si.

Quarto, grande parte do material sobre arquitetura de software tem apenas relevância histórica. Os leitores da página da Wikipédia não deixarão de notar o conjunto desconcertante de siglas e referências cruzadas para um universo inteiro de conhecimento. Contudo, muitas dessas siglas representam tentativas desatualizadas ou fracassadas. Até as soluções que eram perfeitamente válidas

poucos anos atrás podem não funcionar agora porque o contexto mudou. A história da arquitetura de software está repleta de coisas que os arquitetos tentaram e perceberam os efeitos colaterais prejudiciais. Tratamos de muitas dessas lições neste livro.

Atualmente, por que escrever um livro sobre os fundamentos da arquitetura de software? O escopo dessa arquitetura não é a única parte do mundo de desenvolvimento que muda constantemente. Novas tecnologias, técnicas, capacidades... na verdade, é mais fácil achar coisas que não mudaram na última década do que listar todas as mudanças. Os arquitetos de software devem tomar decisões dentro desse ecossistema em constante mudança. Como tudo muda, inclusive os fundamentos sobre os quais tomamos decisões, os arquitetos devem reexaminar alguns axiomas centrais que embasaram os primeiros textos sobre arquitetura de software. Por exemplo, os primeiros livros sobre essa arquitetura não levam em conta o impacto do DevOps porque ele não existia quando foram escritos.

Ao estudar arquitetura, os leitores devem ter em mente que, como nas artes, ela só pode ser compreendida no contexto. Muitas decisões que os arquitetos tomaram foram baseadas nas realidades do ambiente em que se encontravam. Por exemplo, um dos maiores objetivos da arquitetura no final do século XX incluía fazer o uso mais eficiente dos recursos compartilhados, pois toda a infraestrutura na época era cara e comercial: sistemas operacionais, servidores de aplicações, servidores de bancos de dados etc. Imagine passar por um datacenter de 2002 e dizer para o chefe de operações: “Oi, tenho uma ótima ideia para um estilo revolucionário de arquitetura: cada serviço acontece em seu próprio maquinário isolado, com seu próprio banco de dados dedicado (descrevendo o que agora conhecemos como microsserviços). Então, isso significa que precisarei de cinquenta licenças para o Windows, outras trinta licenças para os servidores de aplicação e, pelo menos, cinquenta licenças para os servidores de banco de dados.” Em 2002, tentar construir uma arquitetura como microsserviços teria um custo impeditivo. Todavia, com o advento do open source nos anos seguintes, junto às práticas de engenharia atualizadas via revolução DevOps, foi possível construir uma arquitetura como a descrita. Os leitores devem lembrar que todas as arquiteturas são um produto de seu contexto.

## Definindo Arquitetura de Software

O setor inteiro se esforçou para definir “arquitetura de software” com precisão. Alguns arquitetos se referem a ela como o *blueprint* do sistema, já outros a definem como *roadmap* para desenvolver um sistema. O problema com essas definições comuns é entender o que há de fato em um blueprint ou em um roadmap. Por exemplo, o que é verificado quando um arquiteto *analisa* uma arquitetura?

A Figura 1-2 mostra um modo de pensar a arquitetura de software. Nessa definição, a arquitetura consiste na *estrutura* do sistema (as linhas pretas grossas que dão suporte à arquitetura), combinada com as *características da arquitetura* (“atributos”) que o sistema deve alicerçar, as *decisões da arquitetura*, e por fim, os *princípios de design*.



Figura 1-2. Arquitetura consiste na estrutura combinada com as características da arquitetura (atributos), as decisões da arquitetura e os princípios do design

A *estrutura* do sistema, como mostrada na Figura 1-3, se refere ao estilo (ou estilos) de arquitetura no qual o sistema está sendo implementado (como microserviços, camadas ou microkernel). Descrever uma arquitetura unicamente pela estrutura não exhibe essa arquitetura em sua totalidade. Por exemplo, suponha que se peça a um arquiteto para que descreva uma arquitetura, e ele responde: “É uma arquitetura de microserviços.” Nesse caso, o arquiteto fala apenas da *estrutura* do sistema, não da *arquitetura* do sistema. O conhecimento das características da arquitetura, das decisões da arquitetura e dos princípios do design também é necessário para entender bem a arquitetura do sistema.

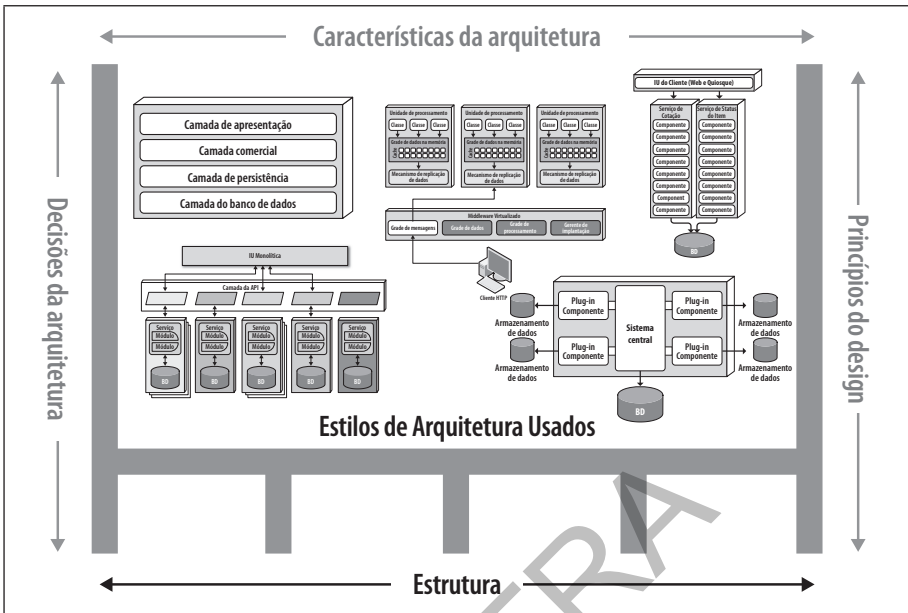


Figura 1-3. Estrutura se refere aos estilos de arquitetura usados no sistema

As características da arquitetura são outra dimensão da definição da arquitetura de software (veja a Figura 1-4). As características definem os critérios de sucesso de um sistema, em geral ortogonal quanto à funcionalidade. Note que todas as características listadas não requerem conhecimento da funcionalidade do sistema, embora sejam necessárias para ele funcionar corretamente. Essas características são tão importantes que dedicamos vários capítulos neste livro para entendê-las e defini-las.

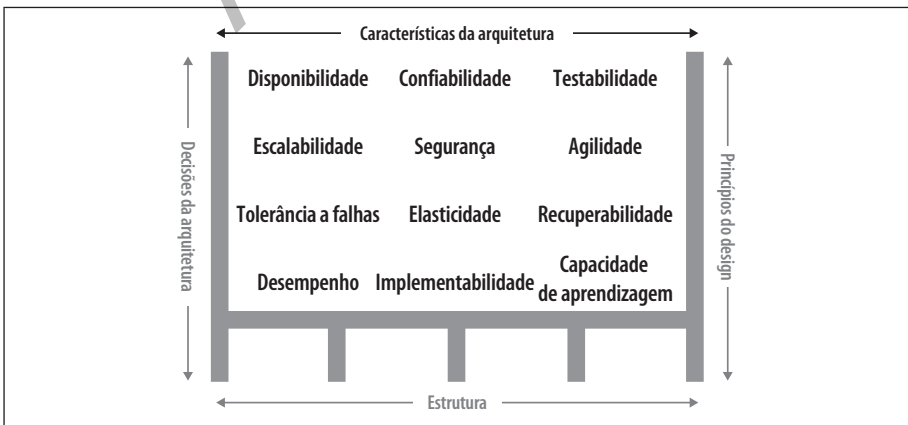


Figura 1-4. As características se referem aos "atributos" que o sistema deve suportar

O próximo fator que define a arquitetura de software são as *decisões da arquitetura*. Essas decisões definem as regras de como um sistema deve ser construído. Por exemplo, um arquiteto pode tomar uma decisão de arquitetura em que apenas as camadas comerciais e de serviços em uma arquitetura em camadas possam acessar o banco de dados (veja a Figura 1-5), impedindo a camada de apresentação de fazer chamados diretos no banco de dados. As decisões da arquitetura formam os limites do sistema e orientam as equipes de desenvolvimento sobre o que é ou não permitido.

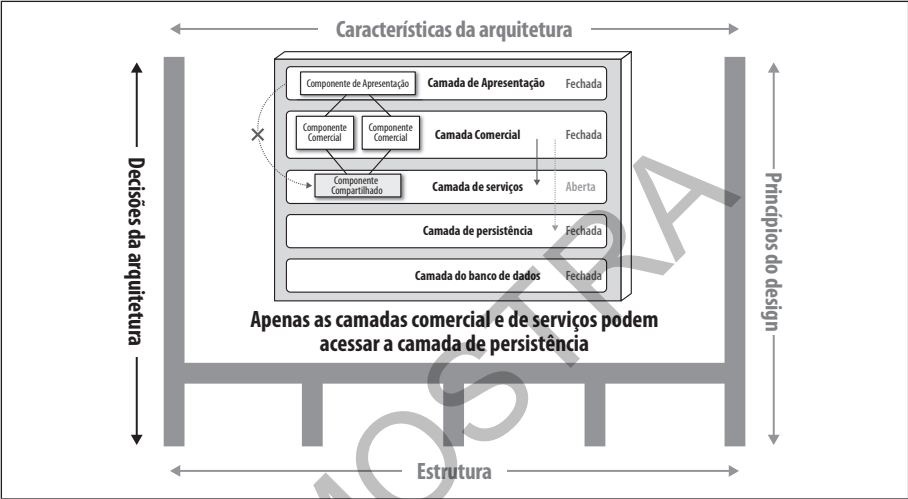


Figura 1-5. Decisões da arquitetura são regras para construir sistemas

Se certa decisão da arquitetura não pode ser implementada em uma parte do sistema devido a alguma condição ou outra restrição, essa decisão (ou regra) pode ser derrubada com algo chamado *variância*. A maioria das organizações tem modelos de variância usados por um Conselho de Revisão de Arquitetura (ARB — sigla em inglês para Architecture Review Board) ou pelo arquiteto-chefe. Esses modelos formalizam o processo de buscar uma variância para certo padrão ou decisão de arquitetura. Uma exceção para determinada decisão de arquitetura é analisada pelo ARB (ou arquiteto-chefe se não existe um ARB), sendo aprovada ou negada com base em justificativas e concessões.

O último fator na definição de arquitetura são os *princípios do design*. Tal princípio difere de uma decisão da arquitetura no sentido de que um princípio do design é uma *diretriz*, não uma *regra* rígida. Por exemplo, o princípio do design mostrado na Figura 1-6 estabelece que as equipes de desenvolvimento devem utilizar a mensageria assíncrona entre os serviços em uma arquitetura de microsserviços para aumentar o desempenho. Uma decisão (regra) da arquitetura