Fundamentos da Arquitetura de Software

Uma abordagem de engenharia

Mark Richards e
Neal Ford

ALTA BOOKS
GRUPO EDITORIAL
Rio de Janeiro, 2024

Sumário

Parte I. Fundamentos

Parte III. Técnicas e Habilidades Sociais

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…".

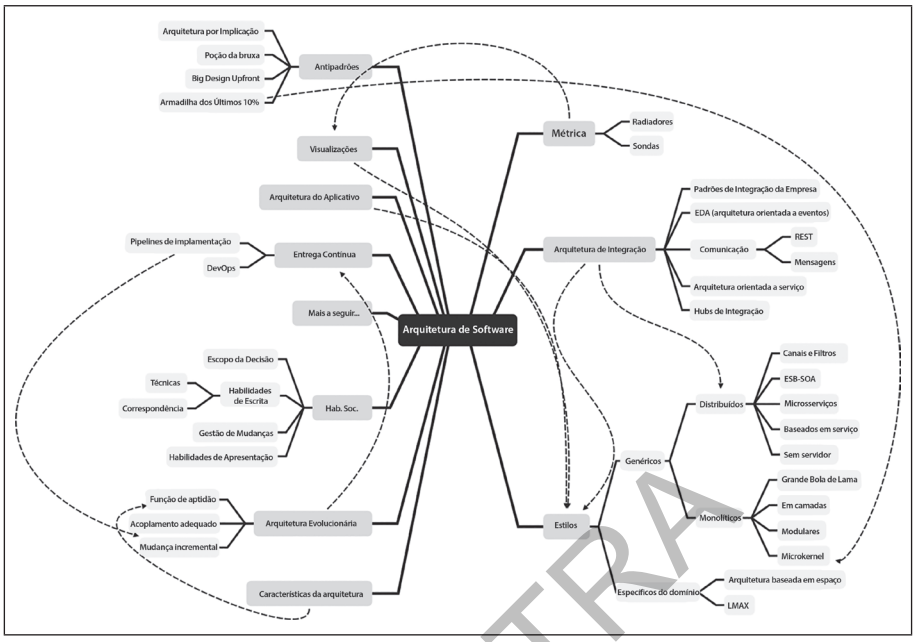

Figura 1-1. As responsabilidades de um arquiteto de software incluem habilidades técnicas, sociais, consciência operacional e muitas outras

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 microsserviços, camadas ou microkernel). Descrever uma arquitetura unicamente pela estrutura não exibe 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 microsserviços." 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.

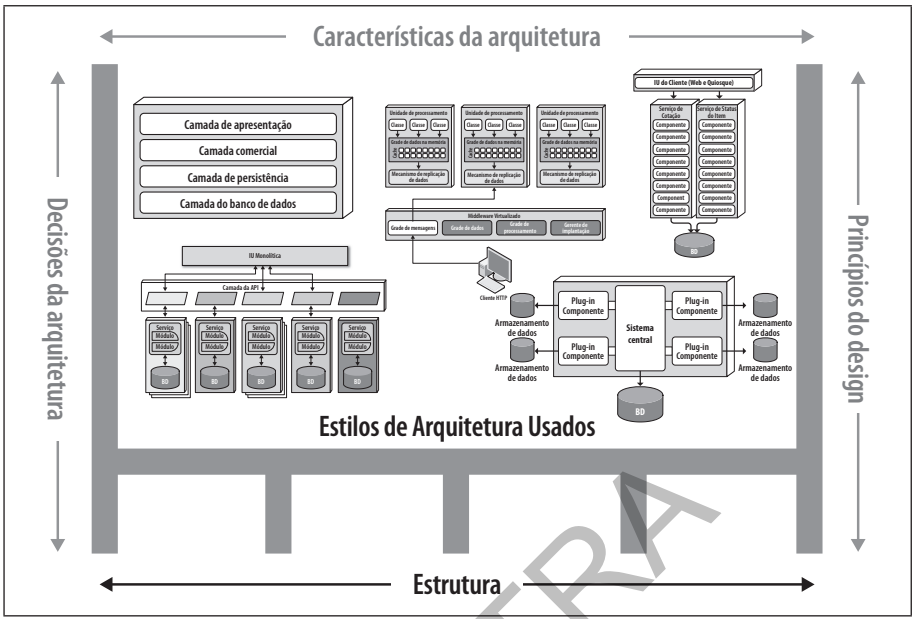

Figura 1-3. Estrutura se refere aos estilos de arquitetura usados no sistema

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.

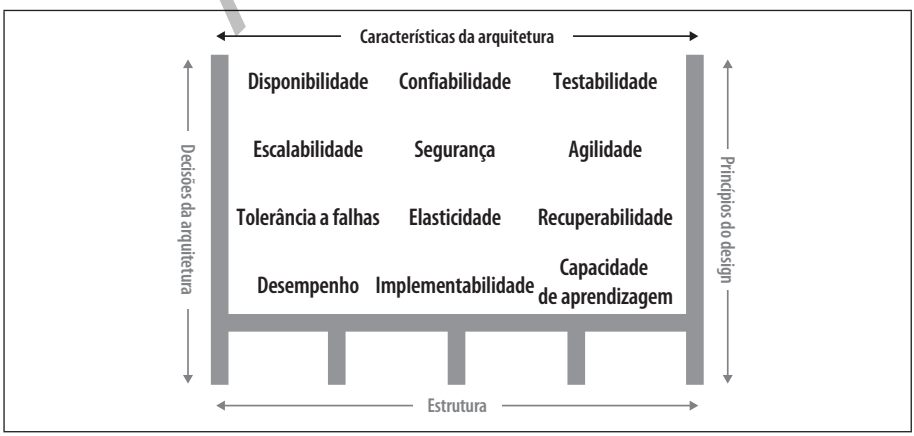

Figura 1-4. As características se referem aos "atributos" que o sistema deve suportar

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.

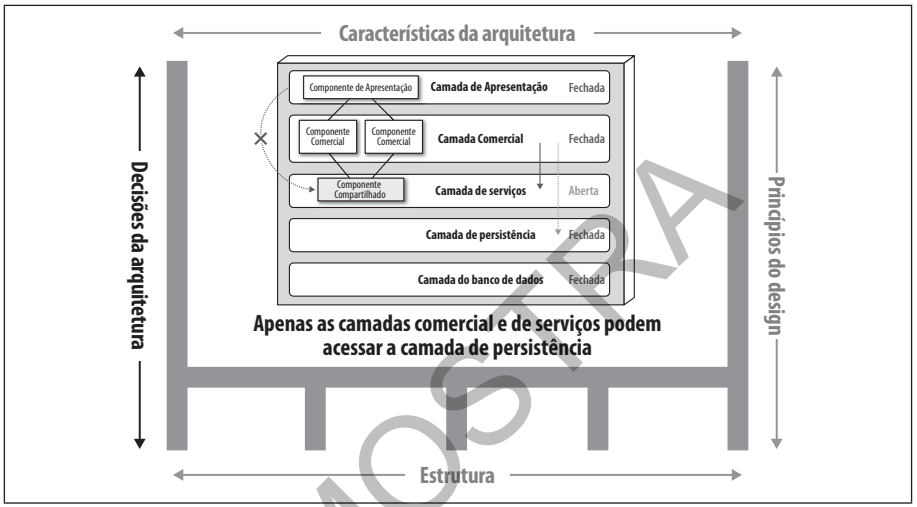

Figura 1-5. Decisões da arquitetura são regras para construir sistemas

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