16 - Arquitetura Orchestration-Driven Service-Oriented

2025-12-22 09:43 am

- Arquitetura irrelevante
    - Embora faça sentido no papel.

História e Filosofia

- Restrições de Hardware direcionaram a arquitetura distribuída nos anos 90 para grandes empresas.
- Sistemas operacionais eram caros, não havia opção open-source confiável.
- Partição técnica ao extremo, tentativa de reusar tudo tanto quanto possível.

Topologia

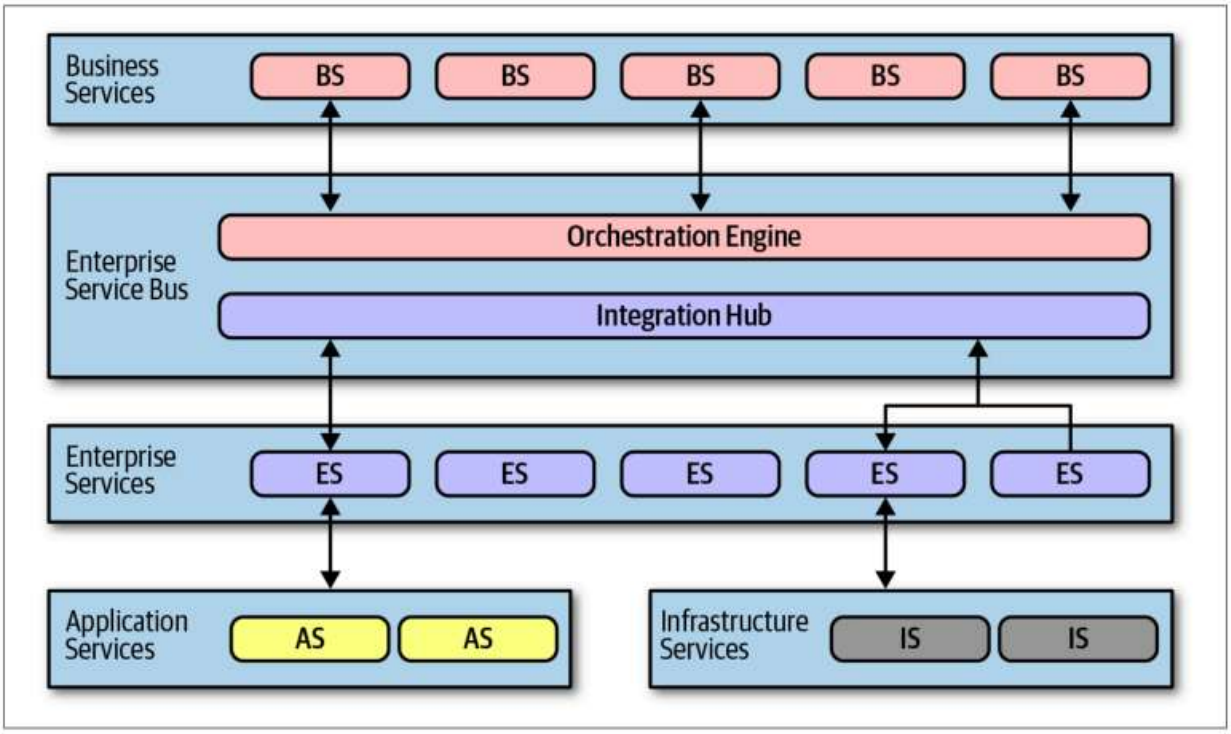


Figure 16-1. Topology of orchestration-driven service-oriented architecture

- Taxonomia de serviços, cada camada com uma responsabilidade.

Taxonomia

- Ideia é reusar código a nível de empresa.

Serviços de Negócio

- Ponto de entrada.
- Sem código, só definição de regras, input, output e schemas.
- Define uma operação (e.g. `ExecuteTrade`).

Serviços da Empresa

- Implementações bem específicas de ações (e.g. `CreateCustomer`)
- Blocos fundamentais dos serviços, acessível a toda a empresa.
- Mas a realidade é mais dinâmica do que essa ideia exige.

Serviços de Aplicação

- Serviços implementados uma vez só para serem reusados.
    - E.g. gerar geolocalização.

Serviços de Infraestrutura

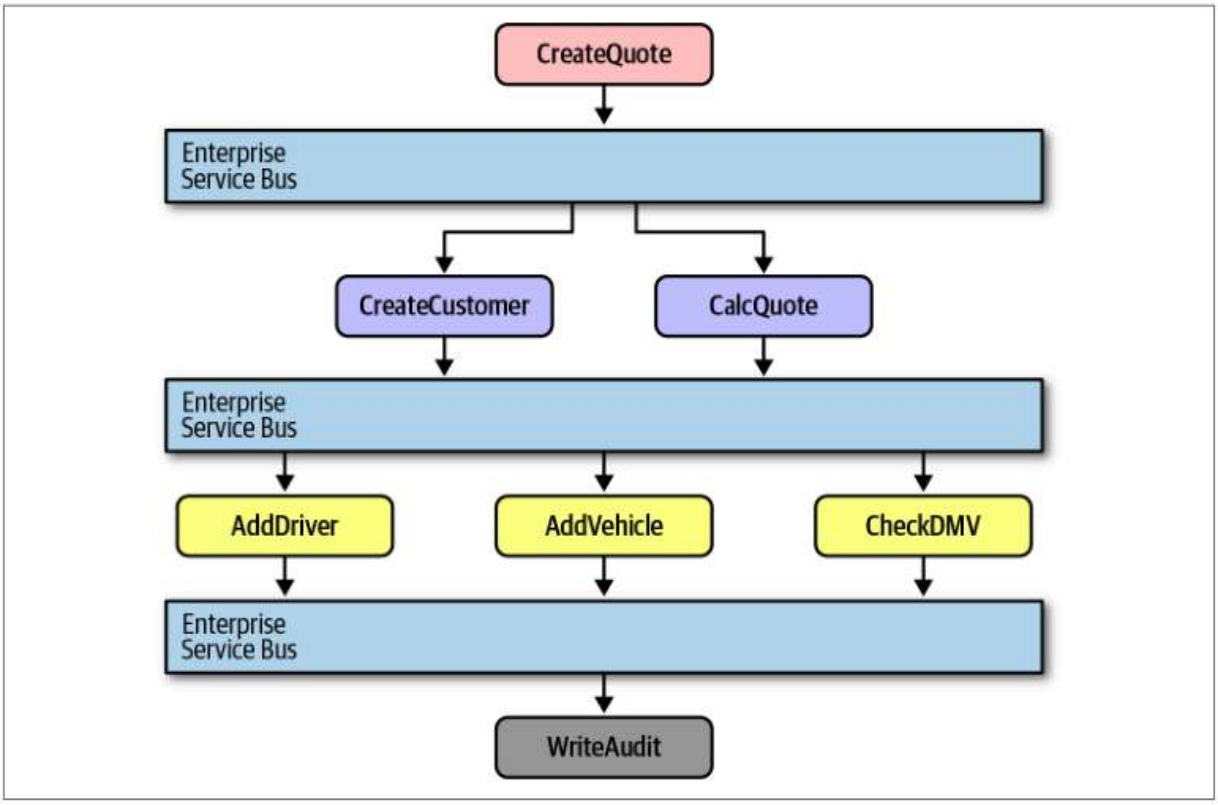
- Serviços de operação
    - Logging, monitoramento, autenticação, etc.

Engine de Orquestração

- Coordena todos os elementos.
- Associado a um banco de dados relacional.
- Define as relações entre Serviços de Negócio e de Empresa, como é a transação, etc.
- Resultado desastroso.
    - Integração impossível, complexa, burocrática.

Fluxo de Mensagens

- Todas as requisições passam pelo engine de orquestração.



- *Figure 16-2. Message flow with service-oriented architecture*

Reuso... e Acoplamento

- O grande objetivo desta arquitetura é o reuso a nível de serviço.
  - .e.g. Todos os setores têm a definição de Cliente
    - Por que não usar sempre o mesmo, uma vez só?
  - Mas se precisar mudar a definição de Cliente para um serviço, isso pode se propagar para todos os outros, mesmo não-relacionados.
- Comportamento fica num lugar só.
- Na prática, desastroso.

Características

Architecture characteristic	Star rating
Partitioning type	Technical
Number of quanta	1
Deployability	★
Elasticity	★★★
Evolutionary	★
Fault tolerance	★★★
Modularity	★★★
Overall cost	★
Performance	★★
Reliability	★★
Scalability	★★★★
Simplicity	★
Testability	★

- Difícil de testar, deployar.
- Complexo.
- Foi um marco na arquitetura de software para ensinar coisas a serem evitadas.