Engenharia de Softwares Escaláveis

Design Patterns e Domain-Driven Design com Java

Agenda

Etapa 5: Mapeamento de Contexto e Integração entre Contextos.

- Enterprise Application Integration.
- Promovendo Integrações.

Enterprise Application Integration

Depois de criarem ilhas de automação ao longo de gerações de tecnologia, os utilizadores e gestores empresariais exigem que sejam construídas pontes contínuas para as unir.

Na verdade, eles estão exigindo que sejam encontradas maneiras de vincular aplicativos em um aplicativo empresarial único e unificado.

O desenvolvimento da **EAI - Enterprise Application Integration**, que permite que muitos dos aplicativos convencionais que existem hoje compartilhem processos e dados, nos permite finalmente atender a essa demanda.

EAI - Enterprise Application Integration representa a solução para um problema que existe desde que os aplicativos migraram dos processadores centrais.

EAI é o compartilhamento irrestrito de dados e processos de negócios entre quaisquer aplicativos e fontes de dados conectados na empresa.

EAI é uma referência aos meios computacionais e aos princípios de arquitetura de sistemas utilizados no processo de **Integração de Aplicações Corporativas**.

Os procedimentos e ferramentas de EAI viabilizam a interação entre sistemas corporativos heterogêneos por meio da utilização de serviços.

Os estilos de integração entre sistemas utilizando-se do **EAI** são:

- **File Transfer** - Integração entre aplicativos através da troca de arquivos em formato de texto definido.
- **Shared Database** - Integração entre aplicativos através da troca de dados entre bases de dados ou tabelas.
- **Remote Procedure Call** - Integração entre aplicativos através da chamada a programas remotos os quais são responsáveis pela extração, envio/recebimento e persistência dos dados no sistema.
- **Messaging** - Integração entre aplicativos de um middleware orientado a mensagem (MOM) o qual é responsável pela entrega dos dados aos sistema integrados.

Forma de Integração	Como Funciona	Exemplos de Tecnologias	Vantagens	Desvantagens
Camada de Dados (Data-Level / ETL)	Acesso direto ao banco de dados ou replicação de dados entre sistemas	ETL (Informatica, Talend, Pentaho), replicação Oracle/SQL Server	Simples de implementar; rápido para relatórios e BI	Alto acoplamento; risco de inconsistência; difícil manutenção
APIs (Application-Level)	Sistemas expõem interfaces (REST, SOAP, gRPC) para troca de dados	REST APIs, SOAP Web Services, GraphQL	Baixo acoplamento; padrão moderno; fácil integração externa	Requer governança de APIs; segurança deve ser bem planejada
Mensageria (Message-Oriented Middleware)	Troca de mensagens assíncronas via filas/tópicos	RabbitMQ, Apache Kafka, IBM MQ, ActiveMQ	Escalável; resiliente; desacoplamento entre sistemas	Maior complexidade de operação; necessidade de monitoramento
EDI (Electronic Data Interchange)	Troca de documentos padronizados (XML, EDIFACT, ANSI X12)	Softwares EDI, VANs de comunicação	Padrão consolidado em setores (logística, financeiro)	Pouca flexibilidade; tecnologia antiga e custosa
SOA / ESB (Service-Oriented Architecture)	Barramento de serviços centraliza integração e orquestra processos	MuleSoft, WSO2, IBM Integration Bus	Governança centralizada; reuso de serviços; suporte a transformações	Pode se tornar monolítico; custo alto; sobrecarga no ESB
Microservices + Event-Driven	Sistemas se integram via APIs leves e eventos assíncronos	Spring Cloud, gRPC, Kafka, Serverless	Escalável; flexível; adequado para nuvem; alta resiliência	Complexidade arquitetural; exige cultura DevOps madura

Promovendo Integrações

Você pode estar se perguntando que tipo específico de interface seria fornecida para permitir a integração com um determinado **contexto limitado**.

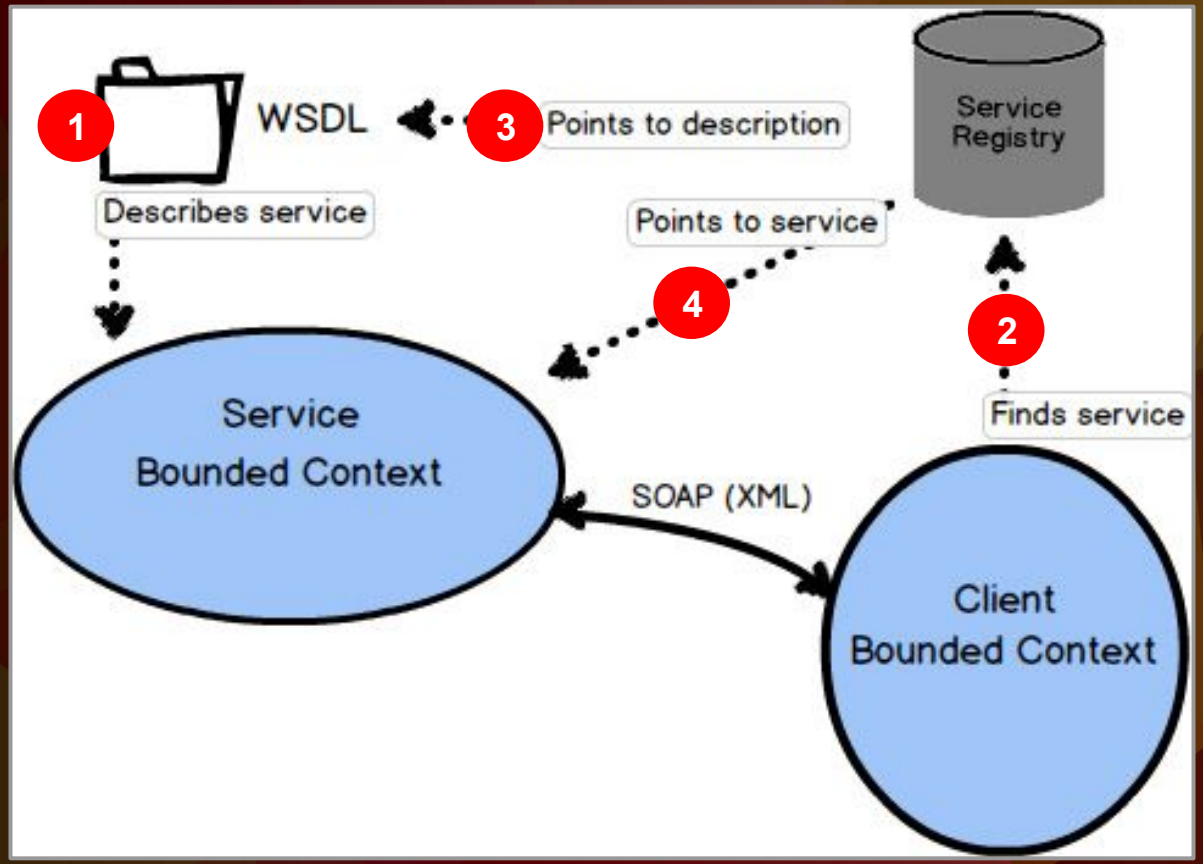
Isso depende do que a equipe proprietária do **contexto limitado** fornece.

Pode ser **RPC via SOAP** ou interfaces **RESTful** com recursos, ou pode ser uma interface de mensagens usando **filas** ou **Publish-Subscribe**.

Nas situações menos favoráveis, você pode ser forçado a usar integração de **banco de dados** ou **sistema de arquivos**.

A integração de **banco de dados** realmente deve ser evitada e, se você for forçado a integrar dessa forma, certifique-se de isolar seu modelo de consumo por meio de uma **camada anticorrupção**.

RPC com SOAP


1
WSDL
3
Points to description
Service Registry
Describes service
Points to service
4
2
Finds service
Service
Bounded Context
SOAP (XML)
Client
Bounded Context

RPC com SOAP

As chamadas de procedimento remoto, ou **RPC**, podem funcionar de várias maneiras.

Uma maneira popular de usar **RPC** é por meio do **Simple Object Access Protocol**, ou **SOAP**.

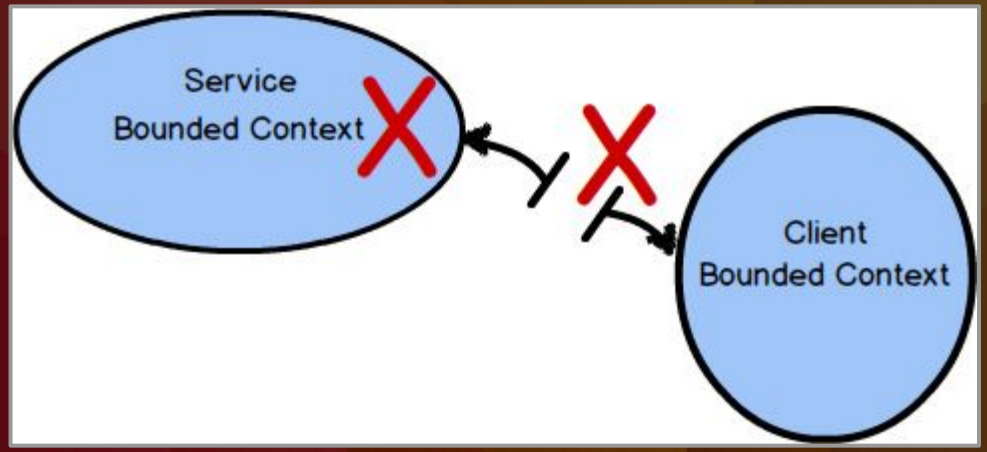
A idéia por trás do **RPC** com **SOAP** é fazer com que o uso de serviços de outro sistema pareça um procedimento local simples ou uma invocação de método.

Ainda assim, a solicitação **SOAP** deve viajar pela rede, chegar ao sistema remoto, ser executada com êxito e retornar resultados pela rede. Isso acarreta o potencial de falha completa da rede ou, pelo menos, de latência imprevista na primeira implementação da integração.

RPC com SOAP

O principal problema com o **RPC**, usando **SOAP** ou outra abordagem, é que pode faltar robustez.

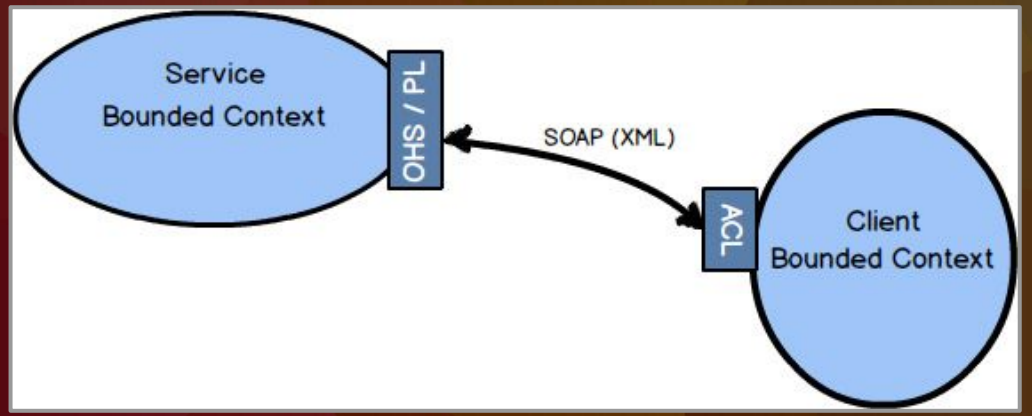
Se houver um problema com a rede ou com o sistema que hospeda a **API SOAP**, sua chamada de procedimento aparentemente simples falhará completamente, fornecendo apenas resultados de erro.



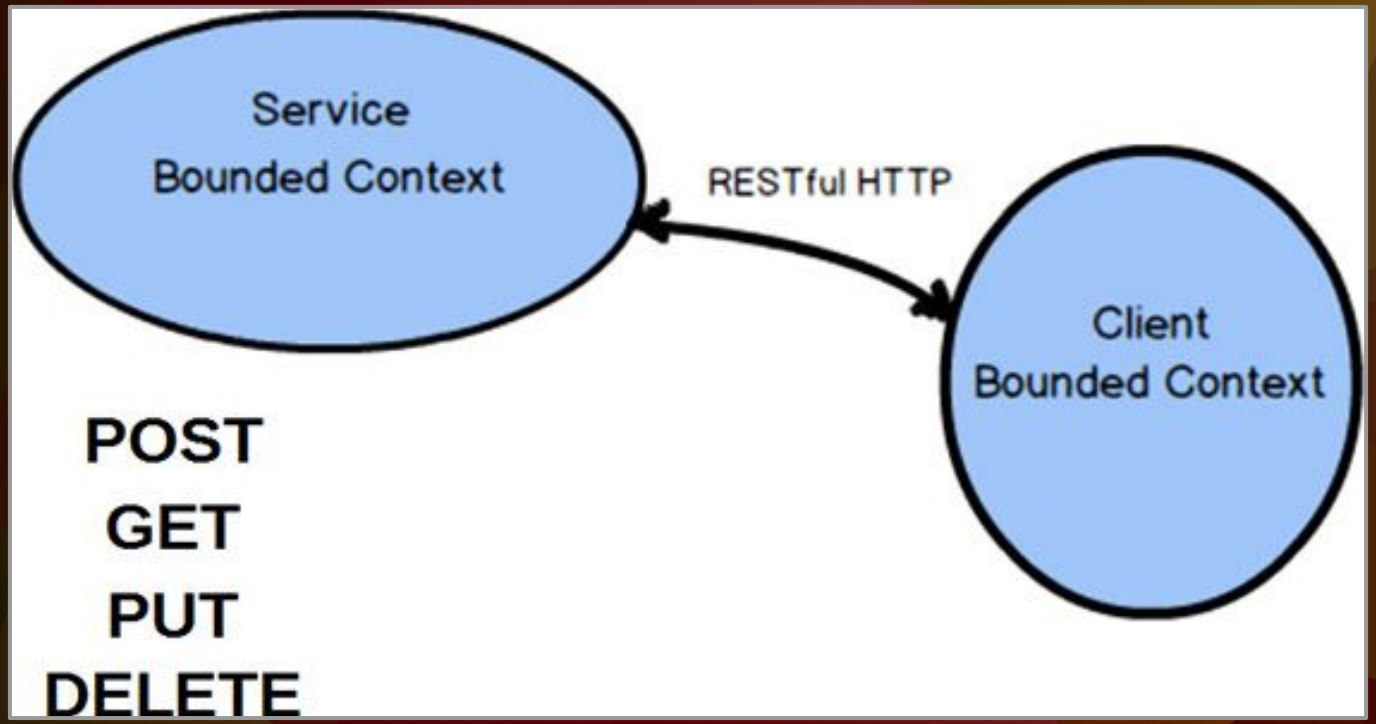
RPC com SOAP

Se você puder influenciar o design do serviço, seria de seu interesse se houvesse uma **API** bem projetada que fornecesse um **serviço de host aberto** com um **idioma publicado**.

De qualquer forma, o **contexto limitado** do seu cliente pode ser projetado com uma **camada anticorrupção** para isolar seu modelo de influências externas indesejadas.



HTTP RESTful



HTTP RESTful

A integração usando **RESTful HTTP** concentra a atenção nos recursos que são trocados entre Bounded Contexts, bem como nas quatro operações principais: POST, GET, PUT e DELETE.

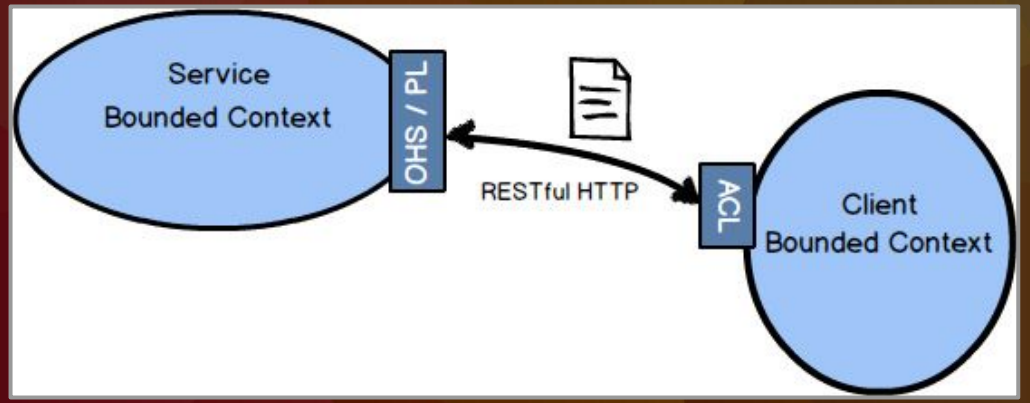
Muitos acham que a abordagem **REST** para integração funciona bem porque os ajuda a definir boas APIs para computação distribuída.

É difícil argumentar contra esta afirmação, dado o sucesso da Internet e da Web.

HTTP RESTful

Um **contexto limitado** de serviço que ostenta uma interface **REST** deve fornecer um **serviço de host aberto** e uma **linguagem publicada**.

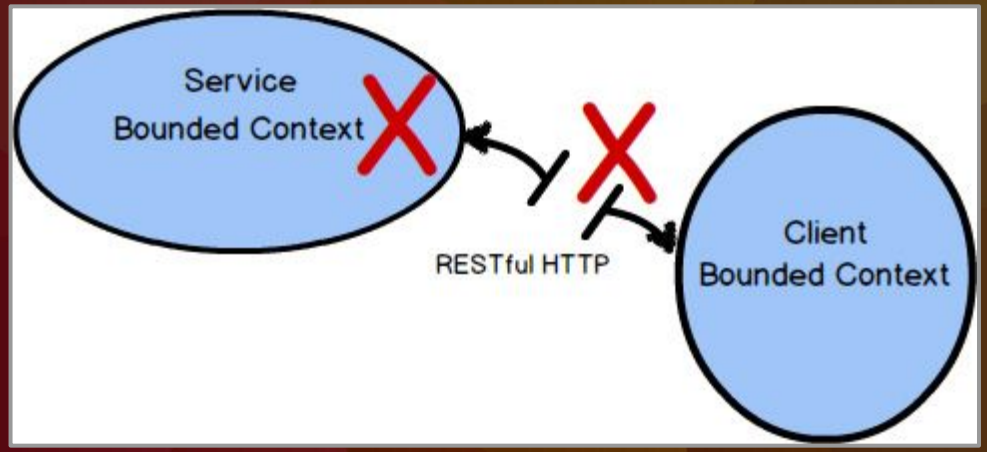
Os recursos merecem ser definidos como uma **linguagem publicada** e, combinados com seus **URIs REST**, eles formarão um **serviço de host aberto** natural.



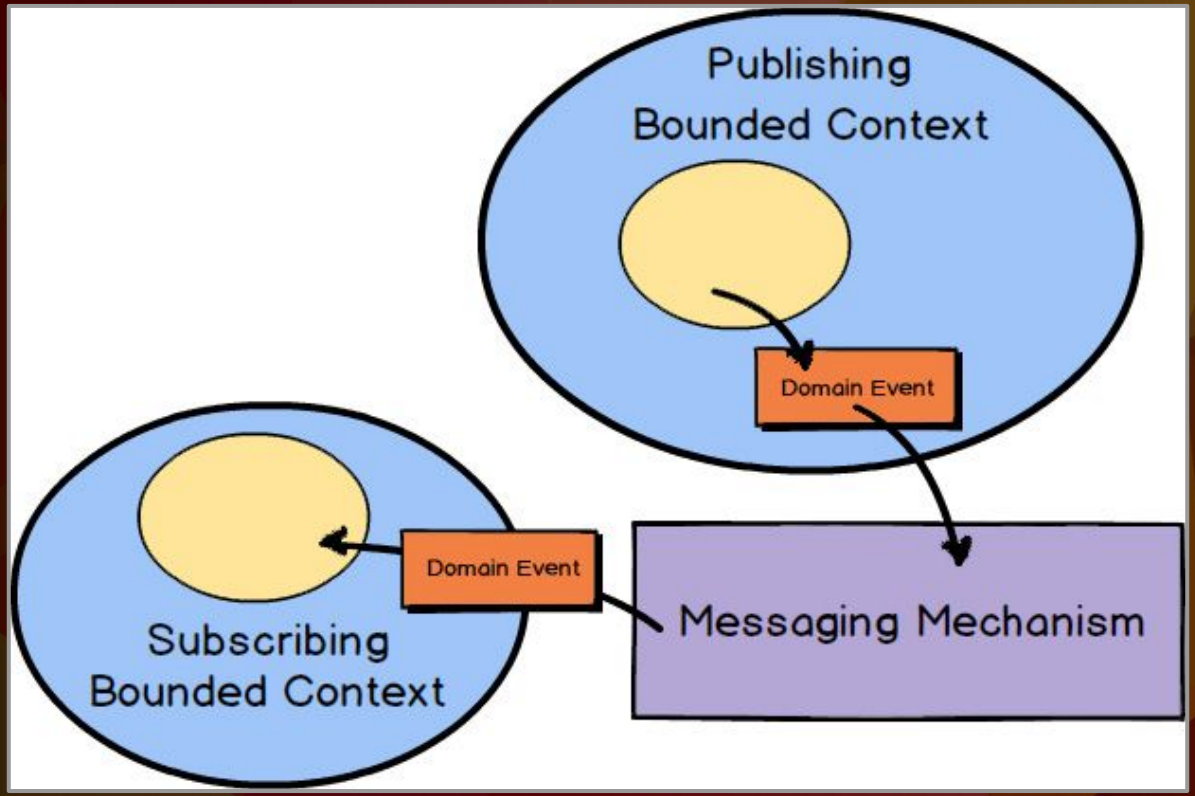
HTTP RESTful

RESTful tenderá a falhar por muitos dos mesmos motivos que o **RPC**: falhas na rede e no provedor de serviços ou latência imprevista.

No entanto, o **RESTful HTTP** é baseado na premissa da Internet, e quem pode encontrar falhas no histórico da Web quando se trata de confiabilidade, escalabilidade e sucesso geral?



Messaging


Publishing
Bounded Context
Domain Event
Domain Event
Subscribing
Bounded Context
Messaging Mechanism

Messaging

Ao usar mensagens assíncronas para integração, muito pode ser realizado por um cliente **Bounded Context** assinando os Eventos de Domínio publicados por seu próprio ou por outro **Bounded Context**.

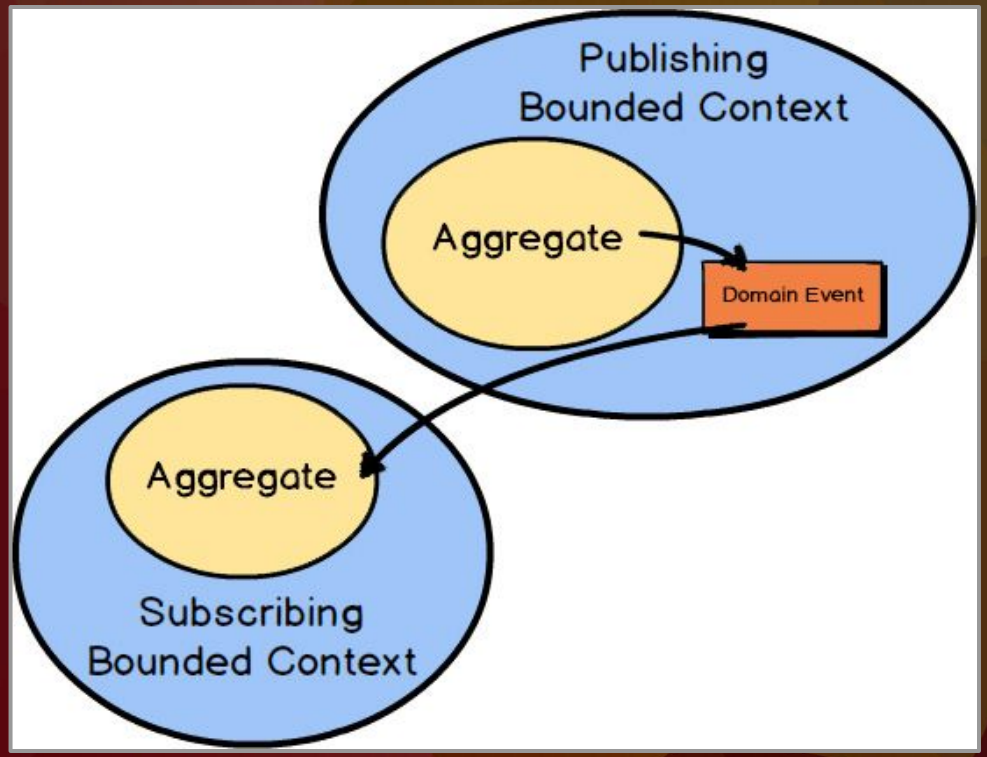
O uso de mensagens é uma das formas mais robustas de integração porque você remove grande parte do acoplamento temporal associado a formulários de bloqueio, como **RPC** e **REST**.

Como você já antecipa a latência da troca de mensagens, tende a construir sistemas mais robustos porque nunca espera resultados imediatos.

Messaging

Normalmente, um **agregado** em um **contexto limitado** publica um **evento de domínio**, que pode ser consumido por qualquer número de partes interessadas.

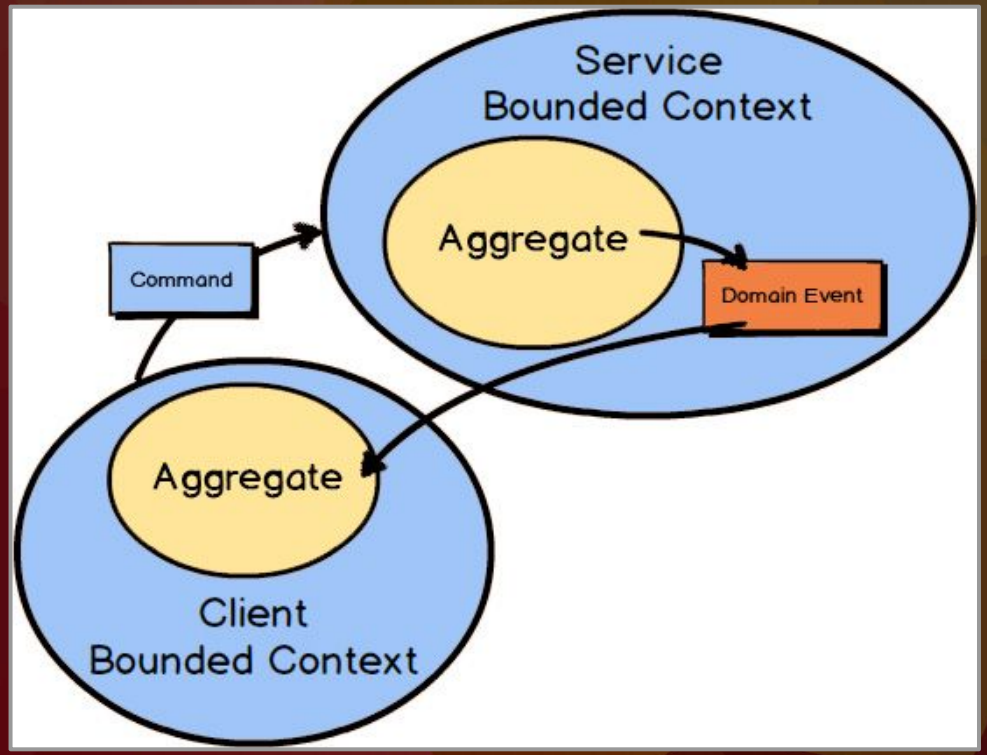
Quando um **contexto limitado** assinante recebe o **evento de domínio**, alguma ação será tomada com base em seu tipo e valor.



Messaging

Um **contexto limitado** assinante sempre pode se beneficiar de acontecimentos não solicitados no **contexto limitado** de publicação.

Um **contexto limitado** do cliente poderá enviar proativamente uma mensagem de comando para um **contexto limitado** de serviço para forçar alguma ação.



Messaging

A qualidade da solução global dependerá fortemente da qualidade do mecanismo de mensagens escolhido.

O mecanismo de mensagens deve suportar entrega pelo menos uma vez para garantir que todas as mensagens serão recebidas eventualmente.

Isso também significa que o **contexto limitado** inscrito deve ser implementado como um **Receptor Idempotente** (receptor de uma solicitação que realiza uma operação de forma que produza o mesmo resultado mesmo que seja realizada várias vezes).