

Engenharia de Softwares Escaláveis

Design Patterns e Domain-Driven Design com Java

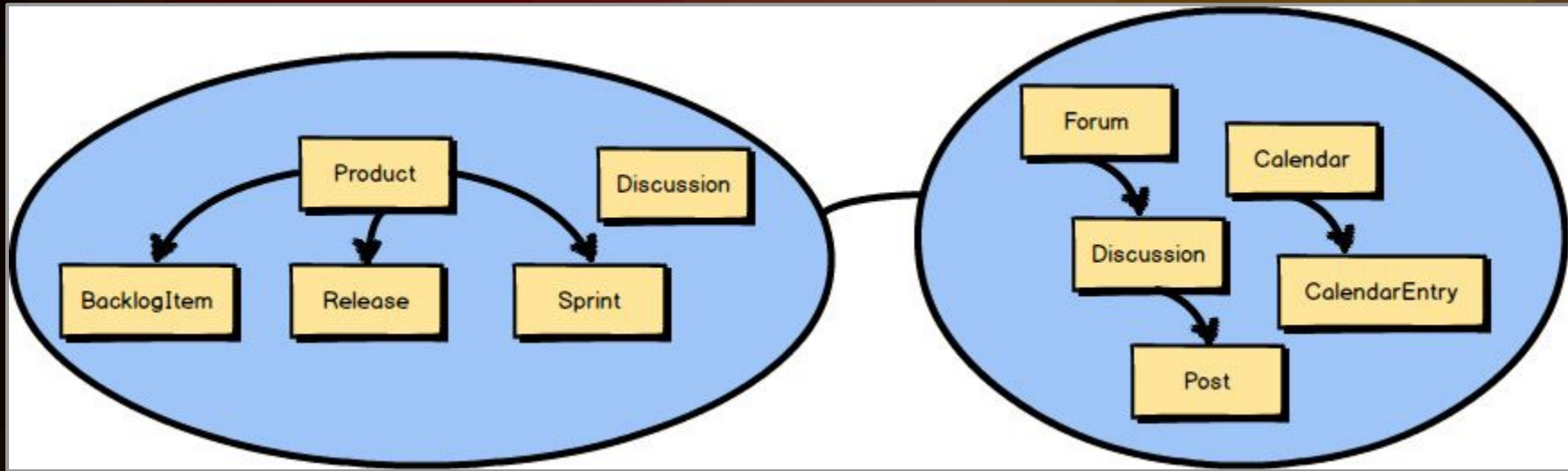
Agenda

Etapas 5: Mapeamento de Contexto e Integração entre Contextos.

- Introdução ao Mapeamento de Contextos.
- Tipos de Relacionamentos entre Equipes.
- Mapas de Contextos.
- Context Mapper.



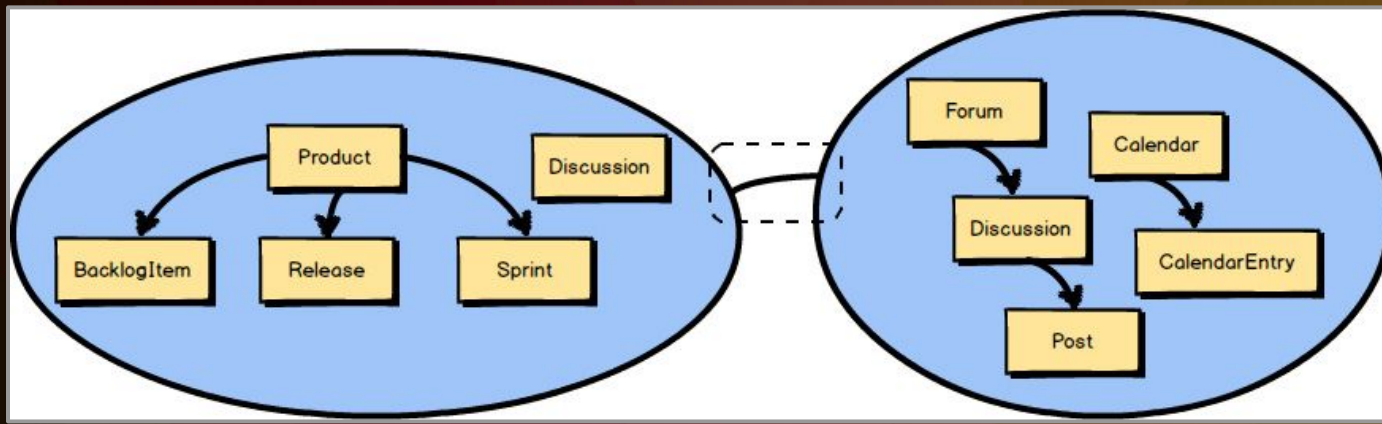
Introdução ao Mapeamento de Contextos



Vimos pelos exemplos do nosso livro-texto que o domínio central do Agile Project Management (o sistema de gestão de projetos de software) teria que ser integrado a outros contextos delimitados.

Essa integração é conhecida em DDD como **Mapeamento de Contexto**.

Um **Mapa de Contexto** não é uma Arquitetura Corporativa ou um diagrama de topologia de sistema.



Mapeamento de Contexto é destacado neste diagrama pela linha dentro da caixa tracejada.

É essa linha entre os dois contextos que representa um **Mapeamento de Contexto**.

Em outras palavras, a linha indica que os dois **Bounded Contexts** estão mapeados de alguma forma - **haverá alguma dinâmica e integração entre as equipes dos dois contextos**.

Considerando que em dois **Bounded Contexts** diferentes existem duas **Linguagens Ubíquas**, esta linha pode representar a tradução que existe entre essas duas linguagens.

Quando falamos sobre **Mapeamento de Contexto**, o que nos interessa é que tipo de relacionamento e integração entre equipes é representado pela linha entre quaisquer dois **Bounded Contexts**.

Limites e contratos bem definidos entre eles apoiam mudanças controladas ao longo do tempo.

Existem vários tipos de **Mapeamentos de Contexto**, tanto de equipe quanto técnicos, que podem ser representados pela linha.

Em alguns casos, tanto um relacionamento entre equipes quanto um mapeamento de integração serão combinados.

Tipos de Relacionamientos entre Equipipes



Os **padrões de cooperação** referem-se a contextos delimitados implementados por equipes com comunicação bem estabelecida.

No caso mais simples estão contextos delimitados implementados por uma única equipe.

Isto também se aplica a equipes com objetivos dependentes, onde o sucesso de uma equipe depende do sucesso da outra e vice-versa.

O principal critério aqui é a qualidade da comunicação e colaboração das equipes.

Parceria

No modelo de parceria, a integração entre equipes de **Bounded Contexts** diferentes é coordenada de forma *ad hoc*.

Uma equipe pode notificar a outra sobre uma mudança na API, e a segunda equipe cooperará e se adaptará – sem dramas ou conflitos.

A coordenação da integração aqui é bidirecional.

Nenhuma equipe dita a linguagem usada para definir os contratos.

As equipes podem resolver as diferenças e escolher a solução mais adequada.

Além disso, ambos os lados cooperam na resolução de quaisquer problemas de integração que possam surgir.

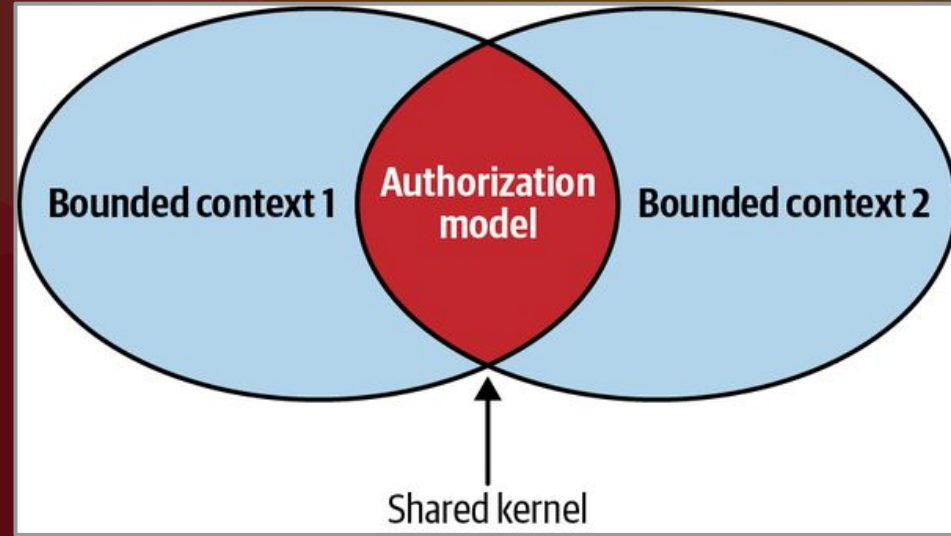
Nenhuma das equipes está interessada em bloquear a outra.

Kernel Compartilhado

Apesar dos **Bounded Contexts** serem limites do modelo, ainda podem haver casos em que o mesmo modelo de um subdomínio, ou parte dele, será implementado em múltiplos **Bounded Contexts**.

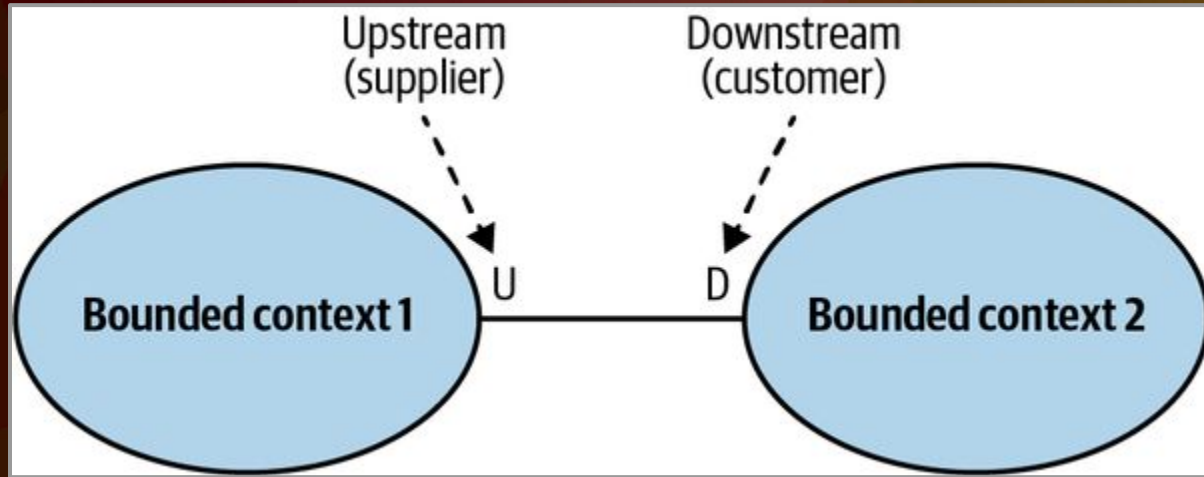
É fundamental ressaltar que o **modelo compartilhado** é desenhado de acordo com as necessidades de todos os **Bounded Contexts**.

Além disso, o **modelo compartilhado** deve ser consistente em todos os contextos delimitados que o utilizam.





O segundo grupo de padrões de colaboração são os padrões **Cliente-Fornecedor**. Um dos **Bounded Contexts** – o fornecedor – fornece um serviço aos seus clientes. O provedor de serviços está “upstream” e o cliente ou consumidor está “downstream”. Ao contrário do **modelo de cooperação**, ambas as equipes podem ter sucesso de forma independente e isso, em geral, leva a um desequilíbrio de forças entre as duas.



Fornecedor (upstream) e o Cliente (downstream).

Conformista

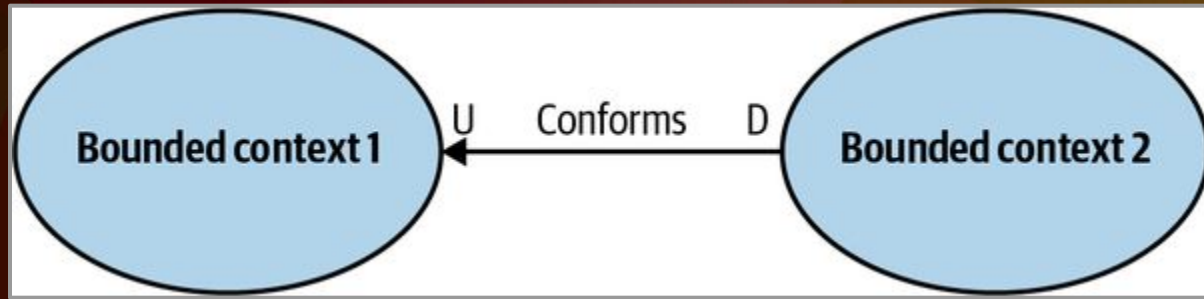
Em alguns casos, o equilíbrio de poder favorece a equipe **Fornecedor**, que não tem motivação real para apoiar as necessidades dos seus clientes.

Em vez disso, ela apenas fornece o contrato de integração, definido de acordo com o seu próprio modelo – é pegar ou largar.

Tais desequilíbrios de poder podem ser causados pela integração com prestadores de serviços externos à organização ou simplesmente pela política organizacional.

Se a equipe **Cliente** puder aceitar o modelo da equipe **Fornecedor**, o relacionamento dos **Bounded Contexts** é chamado de **Conformista**.

O *downstream* está em conformidade com o modelo do **Bounded Context upstream**.



Cliente (downstream) que se conforma com o Fornecedor (upstream).

Camada Anticorrupção

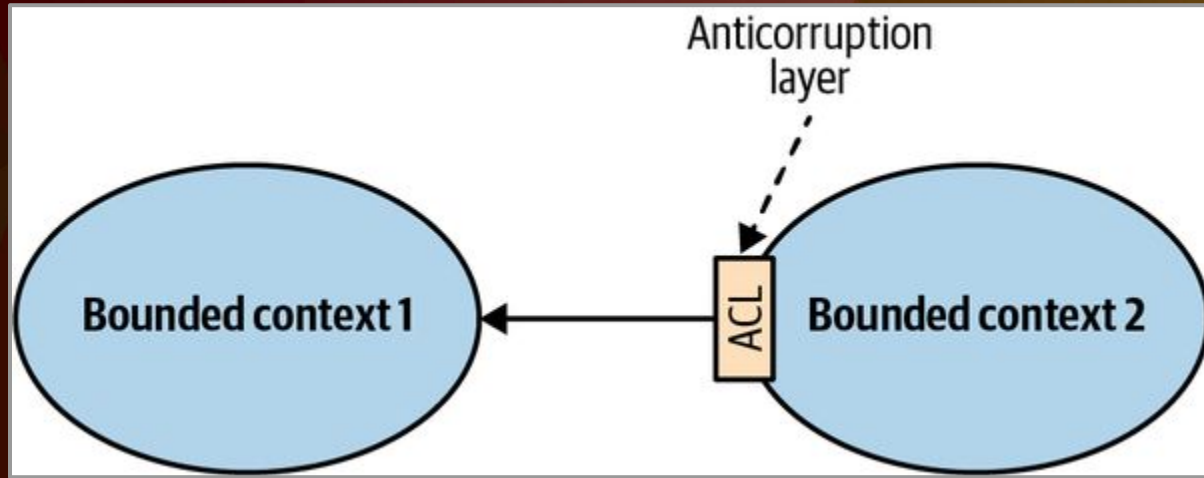
Tal como no padrão conformista, o equilíbrio de poder nesta relação ainda está voltado para o serviço **Fornecedor** (upstream).

No entanto, neste caso, o **Bounded Context Cliente** não está disposto a se conformar.

Em vez disso, pode traduzir o modelo do **Bounded Context Fornecedor** num modelo adaptado às suas próprias necessidades através de uma **Camada Anticorrupção**.

O padrão da **Camada Anticorrupção** aborda cenários em que não é desejável ou não vale o esforço para se adequar ao modelo do **Fornecedor**.

Do ponto de vista da modelagem, a tradução do modelo do **Fornecedor** isola o consumidor de conceitos estranhos que não são relevantes para o seu contexto limitado.



Cliente (downstream) que implementa uma Camada Anticorrupção no relacionamento com o Fornecedor (upstream).

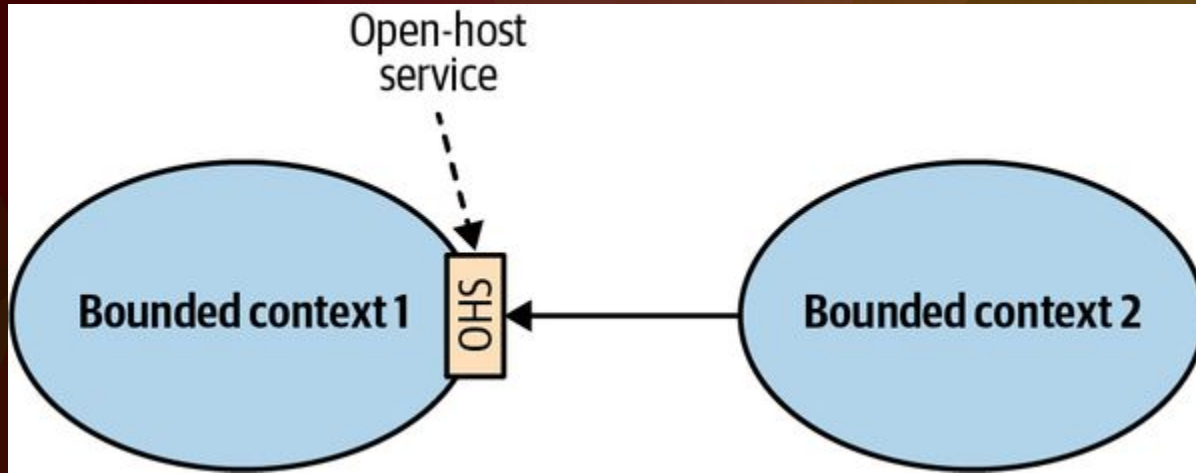
Serviço de Host Aberto

Este padrão aborda casos em que o poder é direcionado para os consumidores.

O fornecedor está interessado em proteger seus consumidores e prestar o melhor serviço possível.

Para proteger os consumidores de mudanças no seu modelo de implementação, o fornecedor upstream desacopla o modelo de implementação da interface pública.

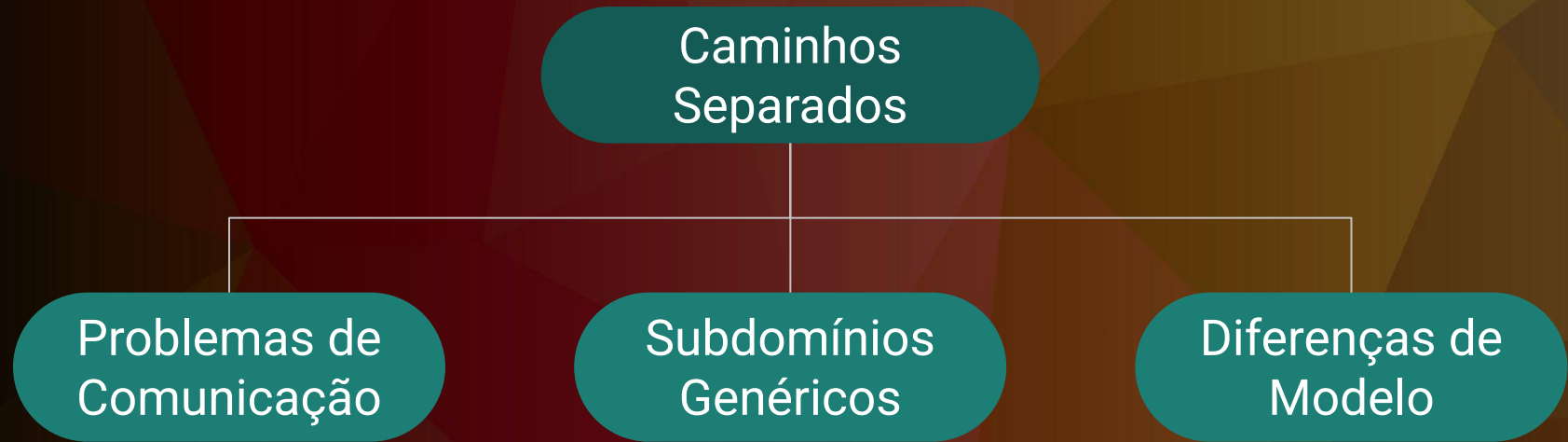
Essa dissociação permite que o fornecedor evolua sua implementação e seus modelos públicos em ritmos diferentes.



A interface pública do fornecedor não se destina a estar em conformidade com a sua linguagem onipresente.

Em vez disso, pretende-se expor um protocolo conveniente para os consumidores, expresso numa linguagem orientada para a integração.

Como tal, o protocolo público é chamado de **Linguagem Publicada**.



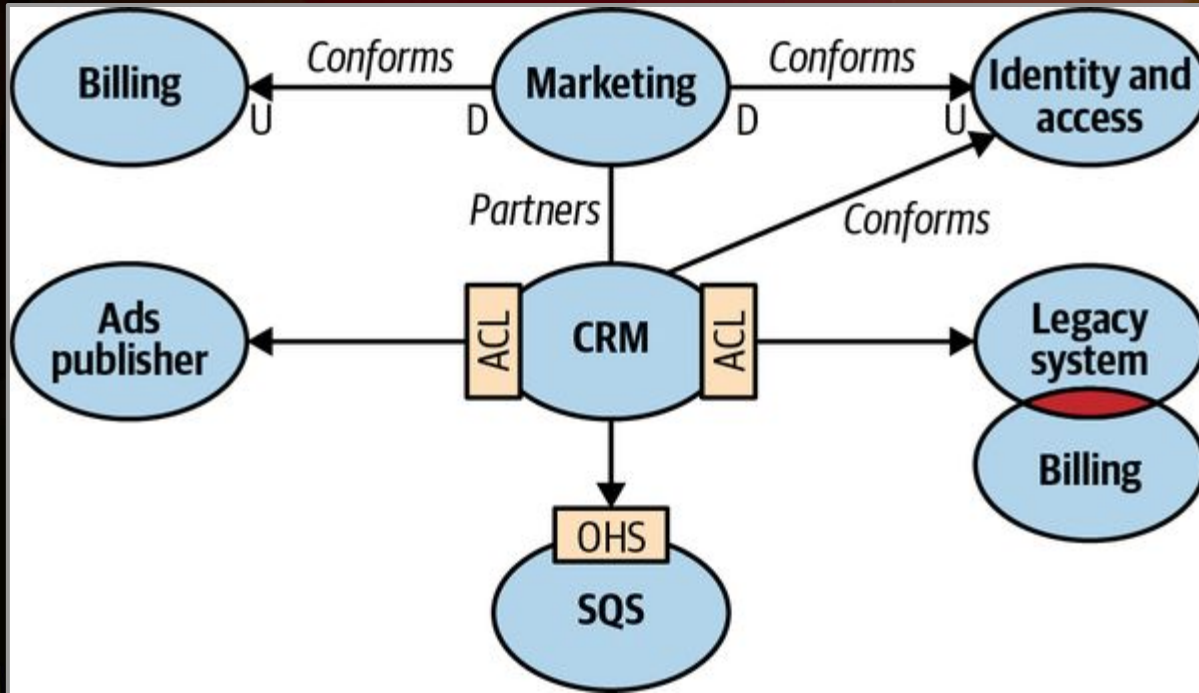
A última opção de colaboração é **não colaborar de forma alguma**.

Esse padrão pode surgir por diversos motivos, nos casos em que as equipes não querem ou não conseguem colaborar.

O padrão de caminhos separados deve ser evitado ao integrar subdomínios principais.

Duplicar a implementação de tais subdomínios desafiaria a estratégia da empresa de implementá-los da forma mais eficaz e otimizada.

Mapas de Contextos



O mapa de contexto é uma representação visual dos **Bounded Contexts** do sistema e das integrações entre eles.

Esta notação visual fornece informações estratégicas valiosas em vários níveis.

Design de alto nível

Um mapa de contexto fornece uma visão geral dos componentes do sistema e dos modelos que eles implementam.

Padrões de Comunicação

Um mapa de contexto descreve os padrões de comunicação entre equipes.

Por exemplo, quais equipes estão colaborando e quais preferem padrões de integração “menos íntimos”, como a camada anticorrupção e padrões de caminhos separados.

Questões Organizacionais

Um mapa de contexto pode fornecer informações organizacionais.

Por exemplo, o que significa se todos os consumidores downstream de uma determinada equipe upstream recorrem à implementação de uma camada anticorrupção, ou se todas as implementações do padrão de vias separadas estão concentradas em torno da mesma equipe?



Context Mapper

The screenshot shows a web browser window with the address bar displaying `contextmapper.org/docs/home/`. The browser's address bar includes navigation icons (back, forward, refresh, home) and a search bar. Below the address bar, there is a green navigation bar with the text "Welcome | Context Mapper" and a "+" icon. The main content area has a header with the Context Mapper logo and navigation links: "Documentation", "Project Background", "Getting Involved", and "News". A search bar is also present in the header. The left sidebar contains a list of navigation items: "Introduction", "Welcome" (highlighted in blue), "Getting started", "Create CML Project", "Examples", "Usage as Library", "Frequently Asked Questions", "Language Reference (Core)", and "Analysis and Design". The main content area features a large "Welcome" heading, followed by a paragraph describing Context Mapper as a modular and extensible modeling framework for Domain-driven Design (DDD). It mentions the "core component" and provides links to the Visual Studio Code Marketplace and Eclipse Marketplace. The "Installation" section follows, with a link to the Visual Studio Code Marketplace and a bullet point stating that it does not support all features available in Eclipse yet, with a link to a feature support table.

Welcome | Context Mapper

contextmapper.org/docs/home/

Administrativo Conteúdo Ferramentas Livros

Todos os favoritos

CONTEXT MAPPER

Documentation Project Background Getting Involved News

Search...

Introduction

Welcome

Getting started

Create CML Project

Examples

Usage as Library

Frequently Asked Questions

Language Reference (Core)

Analysis and Design

Welcome

Context Mapper is a modular and extensible modeling framework for **Domain-driven Design (DDD)** and its strategic patterns. The **core component** provides a DSL to create context maps featuring these DDD patterns. The model behind the language and its semantic rules express **our interpretation of the DDD patterns** and how these patterns can be combined in a concise and consistent manner. At present, Context Mapper comes as an Eclipse plugin, a Visual Studio Code extension, or as a standalone Java **library version**:

Installation

Visual Studio Code Marketplace: [Context Mapper](#)

- Does not support all features we have in Eclipse yet. You can find a feature support table [here](#).

Eclipse Marketplace: [Context Mapper](#)

<https://contextmapper.org/docs/home/>

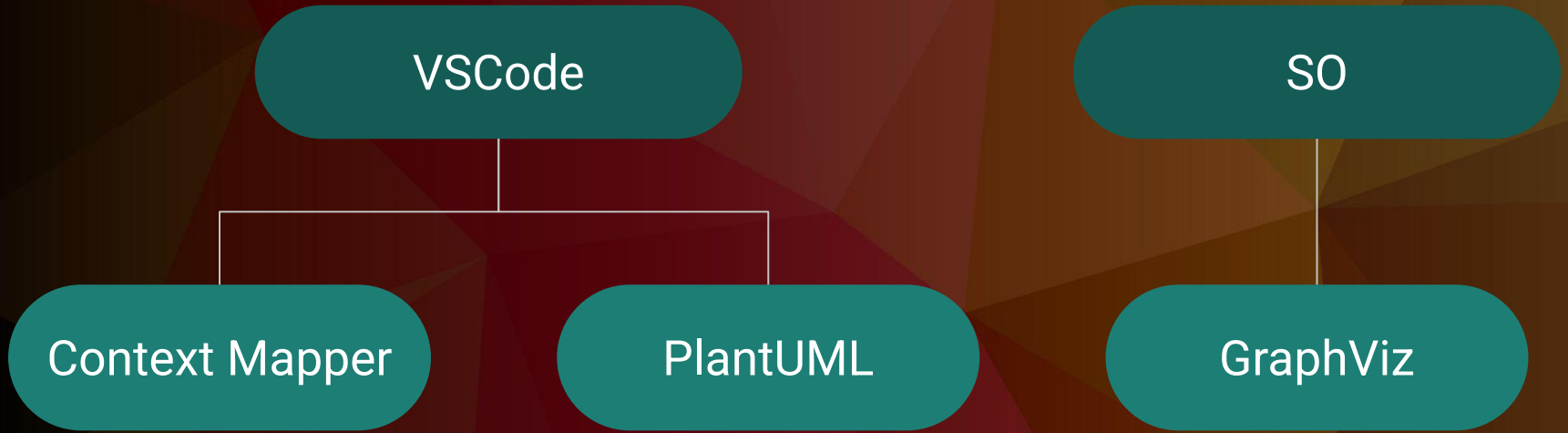
VSCode

SO

Context Mapper

PlantUML

GraphViz




Browser tabs: Welcome | Context Mapper x ContextMapper/context-ma x +

Address bar: github.com/ContextMapper/context-mapper-examples

Navigation: Administrative Conteúdo Ferramentas Livros | Todos os favoritos

Buttons: README Apache-2.0 license



CONTEXT MAPPER

DSL Examples

Build passing Gitpod ready-to-code License Apache 2.0

This project contains example DDD Context Maps written in the ContextMapper DSL. The examples are provided for two different types of users. The simpler business analysis examples should be easy to understand for business analysts without technical background, while the detailed examples are meant for software architects and/or engineers.





Find out more about our DSL and tools on our website <https://contextmapper.org/> and [papers](#) published by [OST \(former HSR\)](#).

Start exploring the examples in the Context Mapper online IDE right now:

[Open in Gitpod](#)

Packages
No packages published

Contributors 4

-  **stefan-ka** Stefan Kapferer
-  **socadk** Doc SoC
-  **boxleytw** Brian Oxley
-  **mLeveIST** Miguel Levezinho

<https://github.com/ContextMapper/context-mapper-examples>

Insurance-Example-Stage-1.cml - context-mapper-examples-master - Visual Studio Code

Arquivo Editar Seleção Ver Acessar Executar Terminal Ajuda

Insurance-Example-Stage-1.cml x Insurance-Example-Stage-1_ContextMap.png x

```
bin > main > insurance-example > Insurance-Example-Stage-1.cml >
1  /* Example Context Map written with 'ContextMapper'
2  ContextMap InsuranceContextMap
3      type = SYSTEM_LANDSCAPE
4      state = TO_BE
5
6      /* Add bounded contexts to this context map:
7      contains CustomerManagementContext
8      contains CustomerSelfServiceContext
9      contains PrintingContext
10     contains PolicyManagementContext
11     contains RiskManagementContext
12     contains DebtCollection
13
14     /* Define the context relationships: */
15
16     CustomerSelfServiceContext <- CustomerManagementContext
17
18     CustomerManagementContext <- PrintingContext
19
20     PrintingContext -> PolicyManagementContext
21
22     RiskManagementContext <-> PolicyManagementContext
23
24     PolicyManagementContext <- CustomerManagementContext
25
26     DebtCollection <- PrintingContext
27
28     PolicyManagementContext <-> DebtCollection
29
30
31     /* Bounded Context Definitions */
32     BoundedContext CustomerManagementContext implements
33
34     BoundedContext CustomerSelfServiceContext implements
```

PROBLEMAS 216 SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL PORTAS

SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation

CML Language Server

Ln 7, Col 39 Tamanho da Tecla Tab: 4 UTF-8 LF Context Mapper DSL