DOMAIN-DRIVEN DESIGN

Zdeněk Merta

Domain-Driven Design
je způsob vývoje a údržby software
v komplexních problémových doménách.

DOMAIN-DRIVEN DESIGN

- Collaborative Modelling
- Strategic Design
- Tactical Design

KOMPLEXITA SOFTWARE

KOMPLEXITA SOFTWARE

- Doménová komplexita
- Technická komplexita

DOMÉNOVÁ KOMPLEXITA

- Složitost problémové domény
- Složitost business problémů, které v ní řešíme

TECHNICKÁ KOMPLEXITA

- Složitost technického řešení
- Počet použitých technologií
- Znalost a prověřenost použitých technologií

Domain-Driven Design
se soustředí na doménovou komplexitu.

BOJ SE SOFTWAROVOU KOMPLEXITOU

- Rozdělení a integrace
- Abstrakce

ROZDĚLENÍ A INTEGRACE

- Rozdělit problém na menší logické celky
- Integrovat je
- Opakovat dokud nemáme kompletní řešení

ABSTRAKCE

- Eliminovat nepodstatné
- Soustředit se na důležité

OBLASTI ŘEŠENÍ

- Problem Space
- Solution Space

SUBDOMÉNY

Pro zvládnutí komplexity v solution space dělíme problem space do menších částí.

Jednotlivé části obvykle odpovídají organizační struktuře společnosti.

Conway's Law

Ne každá část problému vyžaduje stejné úsilí a stejnou kvalitu řešení.

SUBDOMÉNY

- Core
- Supporting
- Generic

CORE DOMAIN

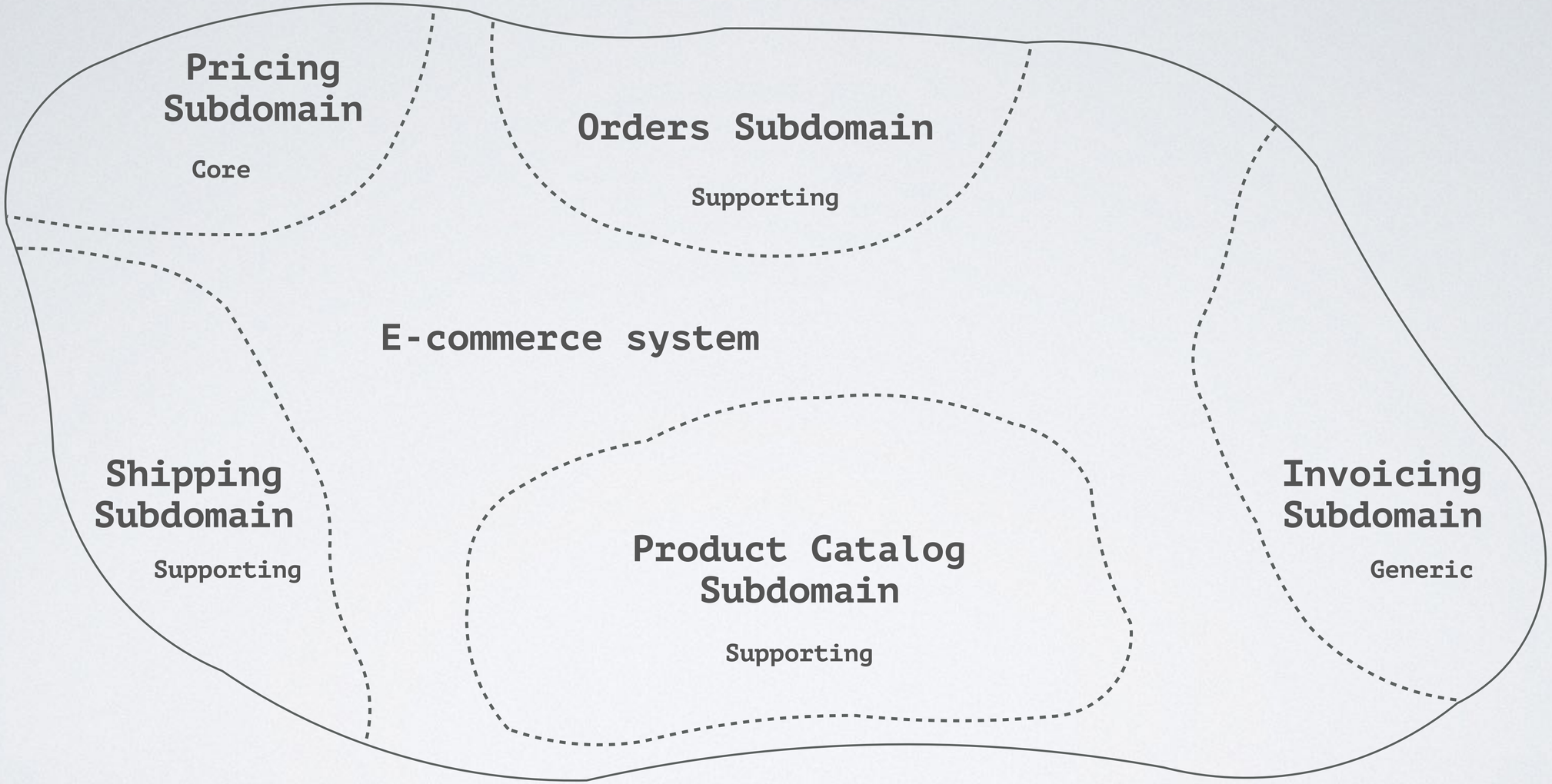
- Nejdůležitější část problémové domény
  - To, co má vydělat peníze
  - To, co má odlišit od konkurence a poskytnout konkurenční výhodu
  - To, proč software vzniká
- Vyžaduje
  - Největší úsilí
  - Nejlepší vývojáře

SUPPORTING DOMAIN

- Nutná pro podporu businessu, ale ten na ni nestojí a nepadá
- Není nutné do ní tolik investovat
- Mohou na ni pracovat méně zkušení vývojáři
- Vhodná pro outsourcing

GENERIC DOMAIN

- Obecná část problémové domény
- Obvykle je možné ji nahradit hotovým řešením od třetí strany


Pricing Subdomain
Core
Orders Subdomain
Supporting
E-commerce system
Shipping Subdomain
Supporting
Product Catalog Subdomain
Supporting
Invoicing Subdomain
Generic

Subdomény patří do problem space.

MODEL

MODEL

- Řešení komplexních problémů
- Abstrakce reality
- Obsahuje pouze koncepty relevantní pro řešení business problémů

Modely nejsou ani správné, ani špatné,
jsou pouze užitečné a neužitečné.

DOMAIN MODEL

DOMAIN MODEL

- Abstrakce problémové domény
- Spolupráce mezi doménovými a technickými experty

VYJÁDŘENÍ DOMAIN MODELU

- Kód
- Konverzace mezi doménovými a technickými experty
- Dokumentace a diagramy

Většina vývojářů se s Domain Modelem nikdy nesetkala.

Jejich denním chlebem je Data Model.

UBIQUITOUS LANGUAGE

Všechna vyjádření modelu spojuje společný jazyk.

UBIQUITOUS LANGUAGE

- Společný jazyk všech členů vývojového týmu
- Tvoří jej businesové termíny, ne technické.
- Pojmenovává doménové koncepty a business procesy.

BOUNDED CONTEXT

V komplexních problémových doménách nikdy neexistuje pouze jeden model.

Přesto se o něj vývojové týmy často pokouší.

ROZSÁHLÝ MODEL - PROBLÉMY

- Vysoká komplexita
- Nejednoznačnost v jazyce
- Ztráta integrity

Z rozsáhlých modelů bohužel často vzniká Big Ball of Mud.

BIG BALL OF MUD

- Rozsáhlý a komplexní model
- Velké množství vzájemně provázaných a propojený částí
- Funguje to, ale nikdo pořádně neví, jak a proč
- Implementace nových funkcí je:
  - Pomalá
  - Drahá
  - Riskantní

Komplexní a rozsáhlé modely dělíme
na více menších modelů
s explicitními hranicemi.

BOUNDED CONTEXT

- Chrání model a explicitně definuje hranice jeho použitelnosti
- Koncepty uvnitř mají dobře definovaný kontext

HRANICE BOUNDED CONTEXTU

- Jednoznačnost jazyka
- Struktura organizace
- Vývojové týmy

IMPLEMENTACE BOUNDED CONTEXTU

- Monolith - modul, namespace, package
- Microservices - služba

Bounded Contexty patří do solution space.

V ideálním případě platí:

1 Bounded Context = 1 Subdomain

CONTEXT MAP

Každá komplexní aplikace se skládá
z více Bounded Contextů.

Context Map mapuje Bounded Contexty
a vztahy mezi nimi.

CONTEXT MAP OBSAHUJE

- Bounded Contexty a vztahy mezi nimi
- Způsob jejich integrace
- Týmy, které na nich pracují
- Riziková místa
- Další užitečné informace - Bandwidth, plánované zrušení/nahrazení, …

CONTEXT MAP

- Zachycuje pouze aktuální stav
- Pro zachycení budoucího stavu je nutné udělat novou mapu

CONTEXT MAP

- Poskytuje high-level pohled na systém
- Pomáhá dělat strategická rozhodnutí
- Umožňuje rychlou orientaci novým členům týmu

VZTAHY MEZI BOUNDED CONTEXTY

- Organizační vztahy - popisují vztahy mezi vývojovými týmy
- Technické vztahy - popisují způsob integrace

UPSTREAM / DOWNSTREAM

- Upstream ovlivňuje Downstream
- Způsoby ovlivnění:
  - Technické
  - Plánování
  - Reakce na požadavky

SHARED KERNEL

- Více týmů sdílí společnou část modelu, ve zbytku se liší
- Zvýšené nároky na komunikaci mezi týmy
  - Každá úprava musí být schválena všemi
  - Nutnost domluvit se, kdo bude dělat vývoj a údržbu
- Sdílet co nejméně

PARTNERSHIP

- Spolupráce mezi dvěma týmy, které pracují na vlastních Bounded Contextech a mají společné cíle
- Společně plánují vývoj a integraci vzájemně závislých částí
- Vývoj směřují do stejného release

CUSTOMER / SUPPLIER

- Poskytovatel (Supplier) je ochotný spolupracovat s konzumentem (Customer)
- Konzument může ovlivnit poskytovaný model

CONFORMIST

- Poskytovatel nechce nebo nemůže spolupracovat s konzumentem
- Konzument nemůže poskytovaný model ovlivnit a musí se přizpůsobit, stává se z něj konformista

ANTI-CORRUPTION LAYER

- Poskytovatel nechce nebo nemůže přizpůsobit model požadavkům konzumenta
- Konzument se nechce přizpůsobit modelu poskytovatele
- Konzument transformuje model poskytovatele na svůj vlastní (vytvoří adaptér)
- Výhody:
  - Oddělenost modelů
  - Minimalizace dopadů změn v modelech

SEPARATE WAYS

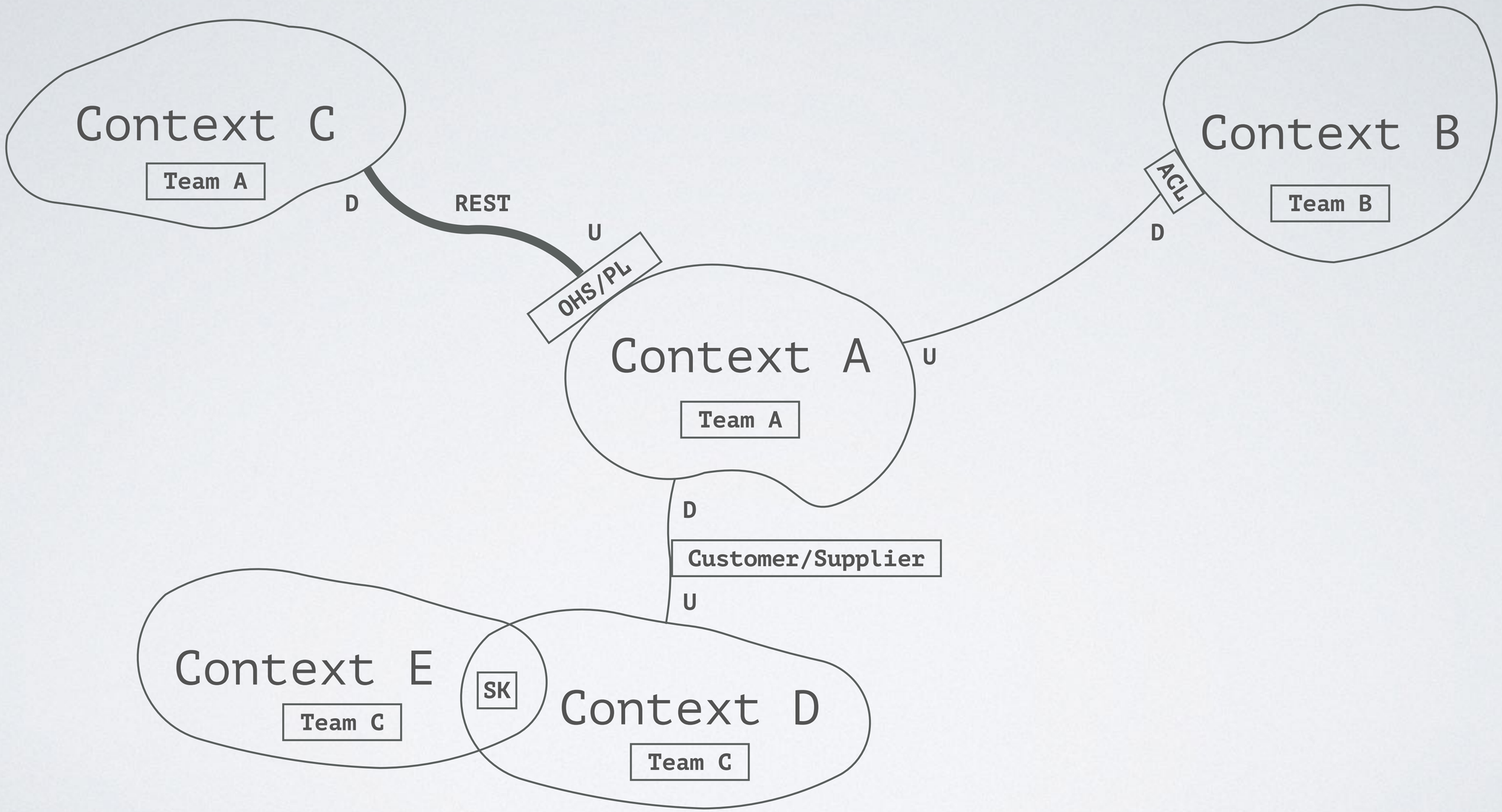
- Bounded Contexty se neintegrují
- Když je integrace:
  - Zbytečná
  - Příliš drahá
  - Dá se nahradit jednodušším řešením (ručně, kompozicí UI, …)

OPEN HOST SERVICE

- Poskytovatel nabízí službu, kterou může použít kdokoliv
- Užitečné, pokud má o integraci zájem více konzumentů

PUBLISHED LANGUAGE

- Poskytnutí obecného jazyka pro výměnu informací mezi dvěma Bounded Contexty
- Dobře dokumentovaný jazyk (např. XML Schema, JSON Schema, Avro, …)
- Často se kombinuje s Open Host Service


Context C
Team A
D
REST
U
OHS/PL
Context A
Team A
U
ACL
D
Context B
Team B
D
Customer/Supplier
U
Context E
Team C
SK
Context D
Team C

ARCHITEKTURA BOUNDED CONTEXTU

Domain-Driven Design nevyžaduje žádnou speciální architekturu.

Zvolená architektura by měla odpovídat komplexitě a důležitosti problému.

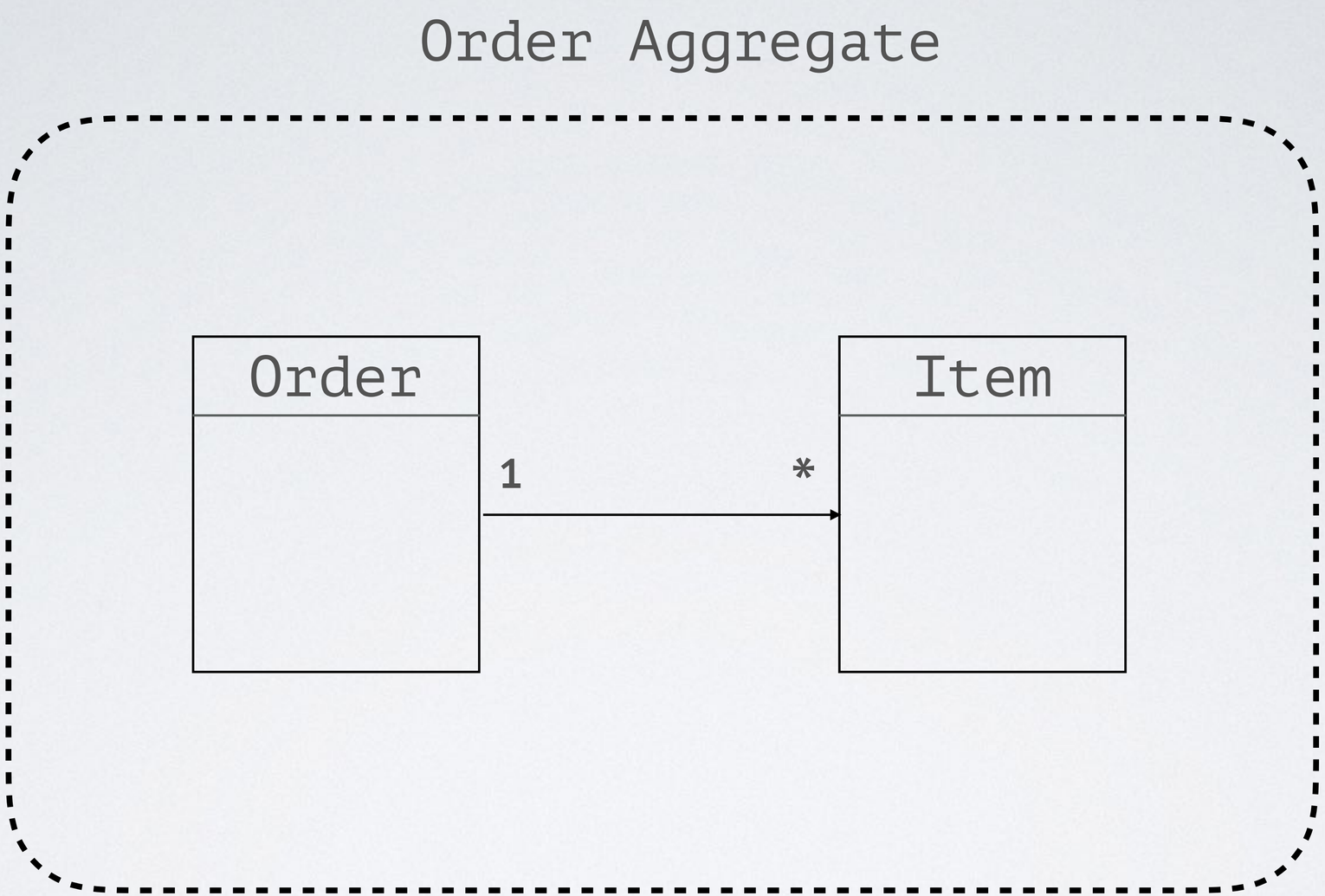
ARCHITEKTURA BOUNDED CONTEXTU

- Rich Domain Model
- CRUD Model
- Anemic Domain Model
- Transaction Script Model
- …

TAKTICKÝ DESIGN

AGGREGATE

- Reprezentuje doménový koncept nebo vztahy mezi doménovými koncepty
- Definuje hranice konzistence konceptu - vše uvnitř musí být vždy a okamžitě konzistentní
- Business pravidla určují, co musí být okamžitě konzistentní
- Reprezentace:
  - Graf objektů (OOP)
  - Datové struktury a funkce (FP)


Order Aggregate
Order
Item
1
*

PRAVIDLA NÁVRHU AGGREGATE

- Aggregate chrání business invarianty
- Aggregate je co nejmenší
- Aggregate se odkazuje na ostatní Aggregates pomocí identifikátoru
- Změna ostatních Aggregates pouze pomocí Eventual Consistency.

V ideálním případě platí:

1 Use Case = 1 Transakce = 1 Aggregate

DOMAIN EVENT

- Informuje o businessově důležité události v doméně
- Domain Model na ně může reagovat
- Integrace s dalšími Bounded Contexty

EVENT SOURCING

Event Sourcing
je způsob ukládání aplikačního stavu.

V tradičních systémech
ukládáme pouze aktuální stav.

V Event Sourced systémech ukládáme všechny změny vedoucí k aktuálnímu stavu.

Změnám stavu říkáme události,
v některých systémech fakta.

EVENT

- Reprezentuje změnu stavu
- Stala se v minulosti
- Není možné ji měnit

EVENT STREAM

- Ukládání událostí
- Sekvence událostí seřazených podle jejich vzniku
- Události se pouze přidávají (append only)
- Mažou se pouze z legislativních důvodů

PROJECTION

- Pohled na data generovaný pomocí historických událostí
- Příklad: Zjištění zůstatku na běžném účtu
  - Přehrajeme historické události
  - Reagujeme na ty, které mění zůstatek

ŽURNÁL FINANČNÍCH TRANSAKCÍ

Datum	Popis	Debet	Credit
01.01.2017	Založení účtu		50 000
05.01.2017	Nákup monitoru (Alza)	12 000	
10.01.2017	Plat		50 000
15.01.2017	Večeře (Bruxx)	2 000	
25.09.2017	Nákup potravin (Tesco)	1 000	

- Tabulka reprezentuje Event Stream jednoho konkrétního účtu
- Řádky reprezentují příslušné události

EVENT STORE

- Úložiště událostí pomocí Event Streamů
- Pokročilejší implementace podporují:
  - Operace s událostmi (emitování nových, odkazy, …)
  - Operace s Event Streamy (mazání, kopírování, …)
  - Projections (vytváření, editaci a mazání)

VLASTNÍ IMPLEMENTACE EVENT STORE

- Základ je jednoduchý
- Jako úložiště je možné použít RDBMS, NoSQL, file systém, …
- Implementace je obvykle naivní a v produkci selhává

EVENT SOURCING - VÝHODY

- Přidaná hodnota pro business - konkurenční výhoda
- Integrace
- Perfektní auditní log
- Debugování
- Testování

EVENT SOURCING - PROBLÉMY

- Výkon
- Verzování událostí
- Omezené možnosti dotazů

EVENT SOURCING
V DOMAIN-DRIVEN DESIGNU

EVENT SOURCING V DOMAIN-DRIVEN DESIGNU

- Ukládání stavu Aggregates
- Každý Aggregate má svůj Event Stream
- Aktuální stav Aggregate je Projection z historických událostí

CQRS

CQRS

- Command and Query Responsibility Segregation
- Dělí systém na dvě části:
  - Command - operace vedoucí ke změně stavu a zápis změn
  - Query - získání různých pohledů na stav

CQRS - TYPY ZPRÁV

- Command Message
- Event Message

COMMAND MESSAGE

- Proveď operaci (např. OpenAccount)
- Vykonání operace může být odmítnuto
- Obvykle platná v rámci jednoho Bounded Contextu

EVENT MESSAGE

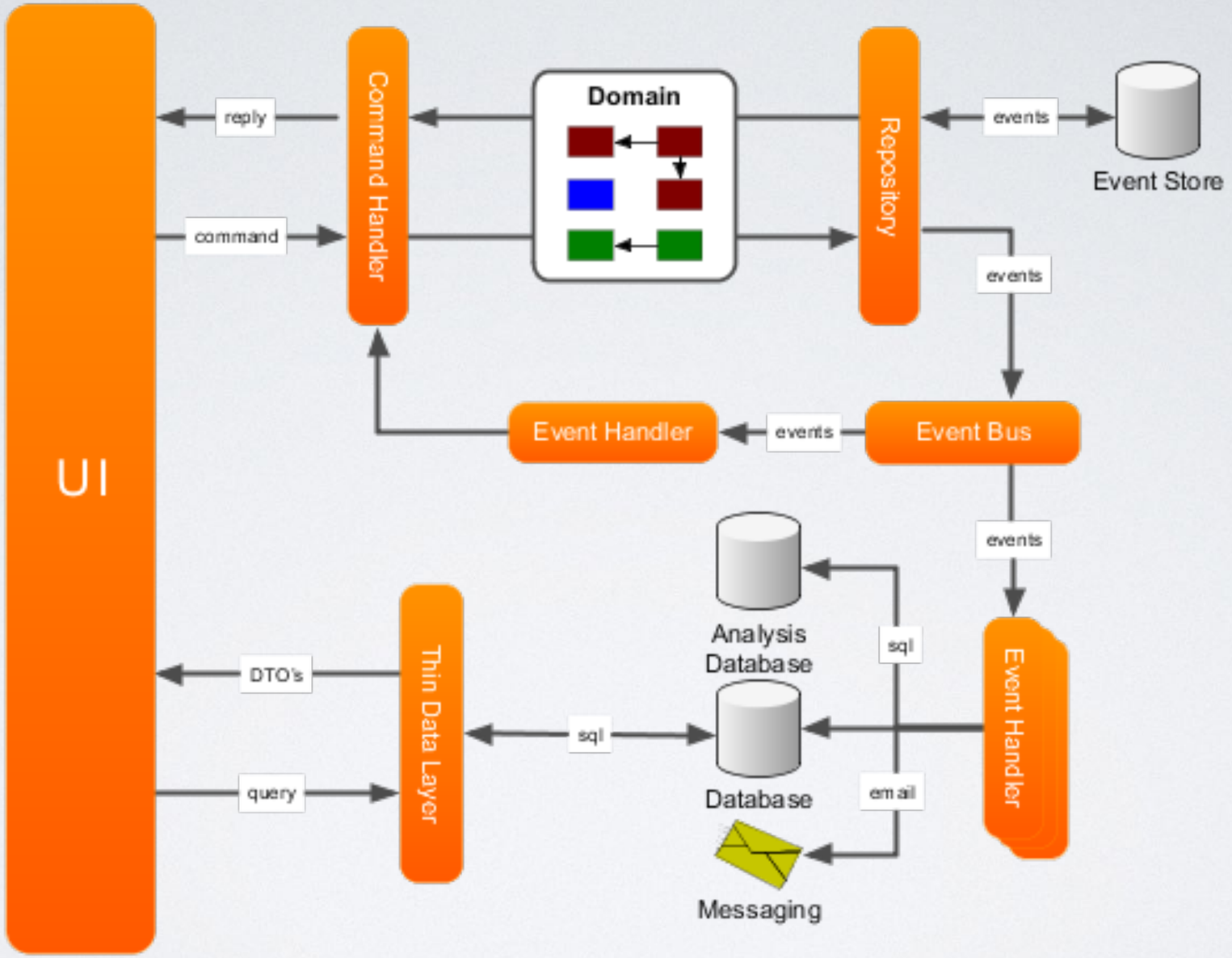
- Informuje o vykonání businessově důležité události (např. AccountOpened)
- Systém na ni může reagovat
- Název v minulém čase
- Nelze ji odmítnout

CQRS - VÝHODY

- Rozdělení odpovědností
- Flexibilita - jeden "Source of truth", ale více pohledů
- Nezávislá optimalizace a škálování

CQRS - NEVÝHODY

- Komplexita
- Duplikování
- Eventual Consistency (potenciálně)


UI
reply
command
Command Handler
Domain
Repository
events
Event Store
events
Event Handler
events
Event Bus
events
Analysis Database
sql
Event Handler
DTO's
Thin Data Layer
sql
Database
query
email
Messaging

CQRS
V DOMAIN-DRIVEN DESIGNU

CQRS, DOMAIN-DRIVEN DESIGN, EVENT SOURCING

- Vhodný doplněk Domain-Driven Design
  (řeší omezené možnosti dotazů v Event Sourcingu)
- Command část - Aggregates, Domain Events, Event Sourcing
- Query část - Projections
- Synchronizace obou částí je Eventual Consistency

OTÁZKY?