

## Program szkolenia:

# Domain Driven Design - projektowanie modeli złożonych domen (część 1)

## Informacje:

<b>Nazwa:</b>	<b>Domain Driven Design - projektowanie modeli złożonych domen (część 1)</b>
<b>Kod:</b>	<b>ddd-workshop-DDD</b>
<b>Kategoria:</b>	Warsztaty eksperckie DDD
<b>Odbiorcy:</b>	developerzy, analitycy, architekci
<b>Czas trwania:</b>	3 dni
<b>Forma:</b>	60% wykłady / 40% warsztaty

Autorski program oparty na 11 latach doświadczenia w stosowaniu i nauczaniu DDD. Warsztat symuluje krytyczne etapy w pracy nad projektem, w którym mamy do czynienia z nietrywialnymi wymaganiami, kilkoma zespołami i wieloma interesariuszami biznesowymi. Zakładamy też, że odpowiedni model da nam przewagę konkurencyjną.

### Zakres

- techniki z poziomu Wzorców Taktycznych: Building Blocks DDD wraz z najlepszymi praktykami oraz elementami rozszerzonymi.
- techniki z poziomu Wzorców Strategicznych: Domain Distillation, Bounded Context Mapping,
- poznasz praktyczne podejścia i sposoby prowadzenia sesji modelowania z wykorzystaniem Event Stormingu.

### Forma

Pierwszego dnia zaczynamy od Event Stormingu procesowego aby odkryć pod-domeny. Następnie tworzymy mapę kontekstów, z których dzięki destylacji wyłaniają się generyczne archetypy modeli biznesowych. Na tej podstawie podejmujemy strategiczne decyzje na poziomie współpracy zespołów i izolacji modeli. Kończymy z projektem krytycznych komponentów ilustrujących problemy techniczne: integracja, optimistic locking, skalowanie.

Drugiego dnia pochylamy się nad wybranymi kontekstami aby stworzyć model taktyczny z wykorzystaniem building blocks. Skupiamy się na granicy agregatów i politykach.

Trzeciego dnia zmieniamy wymagania aby sprawdzić jak zareaguje nasz model. Dodajemy również modelowanie funkcyjne dla dynamicznie zmieniających się reguł.

Formuła warsztatu to kolejne omawianie technik i ich demonstracja przez trenera, po czym uczestnicy w podgrupach samodzielnie przechodzą tą samą ścieżkę na innych wymaganiach. W zadaniach są ukryte pułapki odzwierciedlające realne problemy projektowe, które omawiamy wspólnie po każdym etapie.

### Korzyści płynące z wykorzystania DDD

- Techniki i wzorce strategiczne, których celem jest rozwiązanie typowych problemów organizacyjnych

## BO·TT·EGA

IT minds

- efektywne techniki prowadzenia sesji modelowania z uczestnictwem Ekspertów Domenowych
- skupienie wysiłku na Core Domain: inwestycja najlepszych ludzi i technik DDD jedynie w krytyczne moduły
- techniki separacji osobnych domen (Bounded Context) wyznaczanych przez granice wiedzy Ekspertów Domenowych, redukcja ryzyka związanego z tworzeniem monolitów
- techniki integracji modułów otwartych na zmiany i skalowanie wydajnościowe
- strategię współpracy zespołów (w tym w modelu outsourcing) pracujących nad osobnymi modułami
- Techniki i wzorce taktyczne, których celem jest rozwiązanie typowych problemów technicznych i jakościowych
  - język wzorców dla tworzenia modelu - Building Blocks
  - architektury odpowiednie do tworzenia rozszerzalnych systemów
  - podejścia wspierające i ułatwiające testowanie automatyczne
- Płynna integracja ze Scrum dzięki podejściu Modeling Whirlpool

Projekt referencyjny

Sprawdź naszą implementację przykładowego projektu DDD+CqRS: [Sample Projects](#).

Implementacja DDD

Techniki implementacji DDD (architektura aplikacyjna i systemowa, wykrzyknienie IoC i ORM) są omawiane na szkoleniu [DDD-implementacja](#), które powinno nastąpić w drugiej kolejności, po szkoleniu z zakresu modelowania.

## Zalety szkolenia:

- Uświadomisz sobie kompetencje miękkie jakie powinien posiadać Modelarz
- Dowiesz się jak dzielić system na moduły i odkrywać Bounded Contexty
- Uświadomisz sobie decyzje architektoniczne, jakie musisz podjąć na każdym poziomie z C4 oraz poznasz sposoby zdobywania odpowiedzi na te pytania
- Poznasz różne podejścia do modelowania: obiektowe, funkcyjne, oparte o archetypy

## Szczegółowy program:

### 1. Zadanie wstępne - praca nad systemem, którego granice modułów są niepoprawne

1.1. Próba integracji modułów przy pomocy zdarzeń i komend

1.2. Analiza problemów

1.3. Analiza genezy

### 2. Wstęp do wizualizacji architektury systemów

2.1. Podejście C4

2.2. Drivery architektoniczne na każdym poziomie

2.3. Pytania na jakie musimy sobie odpowiedzieć na każdym poziomie

### 3. Proces i techniki modelowania

3.1. Event Storming

3.1.1. Notacja

3.1.2. Fazy: rozbieżna eksploracja, analiza, synteza i krystalizacja modelu

3.1.3. Facylitacja sesji - Tips and Tricks

3.2. Trzy poziomy Event Stormingu

3.2.1. Big Picture - otoczenie systemu

3.2.1.1. Notacje wspomagające

3.2.2. Process Level

3.2.2.1. Techniki odkrywania pod-domen biznesowych

3.2.2.2. Techniki destylacji generycznych pod-domen

3.2.2.3. Heurystyki proponowania Bounded Contextów

3.2.2.4. Mapowanie Bounded Contextów

3.2.3. Tactical design

3.2.3.1. Określanie granic agregatów na podstawie spójności reguł biznesowych

3.3. Podejście "od procesu" vs podejście "od domeny" w kontekście modelu Cynefin

3.4. Wstęp do archetypów modeli biznesowych na przykładzie generycznych pod-domen

3.5. Techniki lingwistyczne

3.5.1. User Story vs Domain Story

3.5.2. Gibberish Game - usuwanie dwuznaczności i odkrywanie nowych koncepcji domenowych

3.6. Techniki wizualizacji

3.6.1. Grupowanie operacji wokół niezmienników

3.6.2. Metafory wizualne realnych Agregatów

## 4. Wzorce Taktyczne - Building Blocks

4.1. Koncepcja języka Wzorców DDD

4.1.1. Potrzeba większej ilości building blocks niż serwis i encja (procedura i struktura danych)

4.2. Encje

4.2.1. Obiekty do który możemy odnieść się w formie "ten obiekt"

4.3. Agregaty

4.3.1. Hermetyzacja i otwarcie na rozbudowę

4.3.2. Strategie określania granicy agregatu

4.3.3. Modelowanie niezmienników

4.3.4. Techniki lingwistyczne

4.3.4.1. Pełne zdania: podmiot.orzeczenie(dopełnienie, przydawka)

4.3.4.2. Odwrócenie kolejności: Słowo-Znaczenie(Kontekst)-Reguły

4.4. Value objects

4.4.1. Obiekty do który możemy odnieść się w formie "taki obiekt"

4.4.2. Zwiększenie siły wyrazu

4.4.3. Styl funkcyjny

#### 4.5. Serwisy Domenowe

##### 4.5.1. Model procedur biznesowych

#### 4.6. Repozytoria

##### 4.6.1. Abstrakcja magazynu danych

##### 4.6.2. Orientacja na model domenowy zamiast na model danych

#### 4.7. Fabryki

##### 4.7.1. Walidacja

##### 4.7.2. Logika biznesowa podczas składania obiektów

##### 4.7.3. Wsparcie dla testability

#### 4.8. Polityki (strategie)

##### 4.8.1. Modelowanie w stylu funkcyjnym

##### 4.8.2. Open Close Principle (SOLID) w praktyce

##### 4.8.3. Podejście Supple Design

##### 4.8.4. Dekorowanie

##### 4.8.5. Umiejscowienie w 4 poziomach modelu

#### 4.9. Zdarzenia biznesowe

##### 4.9.1. Decoupling Bounded Context

##### 4.9.2. Anticorruption Layer

#### 4.10. Specyfikacje

##### 4.10.1. Modelowanie złożonych warunków biznesowych

#### 4.11. Praktyczne przykłady modelowania biznesowego z wykorzystaniem Building Blocks

### 5. Wzorce Strategiczne

#### 5.1. Bounded Context

##### 5.1.1. Mapowanie

##### 5.1.2. Destylacja

### 5.1.3. Strategie integracji

5.1.3.1. Shared Kernel - najlepsze praktyki

5.1.3.2. Published Language - kiedy warto stosować

5.1.3.3. Customer-Supplier

### 5.1.4. Cztery poziomy modelu

## 6. Architektura aplikacji - uwspólnienie modelu analitycznego i projektowego

6.1. Podział logiki na aplikacyjną i domenową

6.2. Podejście warstwowe - rozmieszczenie building blocks na warstwach

6.2.1. Warstwa interfejsów (prezentacji)

6.2.2. Warstwa logiki - rozwarstwienie na dwie warstwy logiki

6.2.2.1. Logika aplikacyjna (API)

6.2.2.2. Model tego CO system powinien robić (User Case lub User Story)

6.2.2.3. Zapis Domain Story w formie czytelnej prozy

6.2.2.4. Logika domenowa (Building Blocks DDD)

6.2.2.5. Model tego JAK i DLACZEGO system powinien się tak zachowywać

6.2.3. Warstwa infrastruktury

6.2.3.1. Dostęp do danych

6.2.3.2. Infrastruktura techniczna

6.3. Przełożenie Use Case/User Story na warstwę aplikacji

6.4. Przełożenie modelu biznesowego na building blocks warstwy domenowej

## 7. Architektura systemu

7.1. Dobór strategii integracji na podstawie wyników Event Stormingu procesowego

7.2. Zarys architektury integracji modułów

7.2.1. Event Driven Architecture

7.2.2. Microservices