

## 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-DDD</b>   |
| <b>Kategoria:</b>      | Domain Driven Design   |
| <b>Grupa docelowa:</b> | developerzy<br>architekci<br>analitycy                                       |
| <b>Czas trwania:</b>   | 3 dni  |
| <b>Forma:</b>          | 60% wykłady / 40% warsztaty  |

Autorski program oparty na 7 latach doświadczenia w stosowaniu i nauczaniu DDD.  
Zarys DDD

Domain Driven Design jest obecnie jedyną metodyką, która w sposób kompleksowy wspiera od strony technicznej zwinne podejście do wytwarzania oprogramowania.

Kompleksowe podejście DDD obejmuje: modelowanie złożonych domen we współpracy z Ekspertem Domenowym, zalecane architektury i projektowanie z wykorzystaniem sprawdzonych Building Blocks aż po techniki tworzenia testowalnego kodu, który z założenia jest otwarty na iteracyjny proces rozbudowy i kruszenia wiedzy (Knowledge Crunching).

### Korzyści płynące z wykorzystania DDD

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

### Zakres

Techniki modelowania problemów biznesowych

- techniki z poziomu Wzorców Taktycznych: Building Blocks DDD wraz z najlepszymi praktykami oraz elementami rozszerzonymi.

- techniki z poziomu Wzorców Strategicznych: Domain Distillation, Bounded Context,
- poznasz praktyczne podejścia i sposoby prowadzenia sesji modelowania.

#### 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.

#### Forma

Przyrostowe stosowanie wiedzy w praktyce

Szkolenie prowadzone jest w formie, która łączy następujące po sobie na przemian wykłady, warsztaty i dyskusje.

Podczas wykładów trener prezentuje kolejne rozdziały wiedzy merytorycznej, uzupełniając je o komentarze na podstawie własnych doświadczeń.

Podczas warsztatów tworzymy dwa moduły systemu klasy ERP. Kolejne zadania polegają na przyrostowym dodawaniu nowych funkcjonalności w sposób ilustrujący zagadnienia teoretyczne poznane podczas poprzedzającego je wykładu.

Podczas dyskusji uczestnicy mają dostęp do wiedzy eksperckiej trenera oraz mają możliwość zweryfikowania swoich rozwiązań z wypracowanymi przez innych uczestników szkolenia.

#### Realna sesja modelowania Modelling Whirlpool

W pierwszej fazie warsztatów uczestnicy rozwiązują postawione przed nimi problemy pracując nad modelem w grupach - faza ta ma na celu nabranie biegłości w posługiwaniu się technikami DDD.

Po zapoznaniu się z technikami DDD przychodzi czas na przeprowadzenie realnej sesji Modelling Whirlpool gdzie członkowie każdej grupy grają role: Ekspert Domenowy, Modelarz, Koordynator - faza ta ma ba celu ugruntowanie się wiedzy gotowej do stosowania w codziennej praktyce.

#### Projekt referencyjny

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

#### Zalety szkolenia:

- Uświadomisz sobie kompetencje miękkie jakie powinien posiadać Modelarz
- Techniki modelowania: lingwistyczne i wizualne
- Prezentacja alternatywnych podejść wraz z omówieniem konsekwencji

## Szczegółowy program:

### 1. Wstęp do Domain Driven Design - unifikacja analizy i projektowania

#### 1.1. Stosowalność DDD

1.1.1. Klasa złożoności systemu

1.1.2. Głębokość systemu - co system robi "pod maską"

1.1.3. Systemy sterujące światem vs systemy zbierające dane

1.1.4. Kiedy nie stosować DDD

1.1.5. Podejście DDD Lite - zwiększenie jakości technicznej

#### 1.2. Role w procesie - odpowiedzialność i cechy osobowości i umiejętności

1.2.1. Ekspert domenowy - źródło wiedzy, walidator modelu

1.2.2. Modelarz (architekt domeny) - twórca modelu

1.2.3. Facilitator - koordynator procesu we wstępnej fazie

#### 1.3. Wprowadzenie Ubiquitous Language

1.3.1. Wspólna płaszczyzna porozumienia pomiędzy Ekspertami Domenowymi i zespołem developerskim

### 2. Proces i techniki modelowania

2.1. Podejście "od procesu" vs podejście "od domeny"

2.2. Zwinny Proces Modelowania "Model Exploration Whirlpool"

2.2.1. Fazy

2.2.2. Artefakty

2.2.3. Walidacja modelu

2.3. Techniki lingwistyczne

2.3.1. User Story vs Domain Story

2.3.2. Techniki pełnych zdań zamiast zbierania rzeczowników

2.3.3. Eksploracja domeny przy pomocy zdań podmiot.orzeczenie(dopełnienie, przydawka)

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

2.3.5. Słowo-Znaczenie(Kontekst)-Reguły

2.3.5.1. Odwrócenie kolejności w celu odkrywania ukrytych koncepcji domenowych

2.4. Techniki wizualizacji

2.4.1. Grupowanie operacji wokół niezmienników

2.4.2. Metafory wizualne realnych Agregatów

2.4.3. Poziomy modelu

2.4.4. Separacja modelu pod kątem podatności na zmiany i niestabilności

2.5. Przełożenie Use Case/User Story na warstwę aplikacji

2.6. Przełożenie modelu biznesowego na building blocks warstwy domenowej

### 3. Wzorce Taktyczne - Building Blocks

3.1. Koncepcja języka Wzorców DDD

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

3.2. Encje

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

3.3. Agregaty

3.3.1. Hermetyzacja i otwarcie na rozbudowę

3.3.2. Strategie określania granicy agregatu

3.3.3. Modelowanie niezmienników

3.3.4. Techniki lingwistyczne

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

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

3.4. Value objects

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

3.4.2. Zwiększenie siły wyrazu

3.4.3. Styl funkcyjny

3.5. Serwisy Domenowe

3.5.1. Model procedur biznesowych

3.6. Repozytoria

3.6.1. Abstrakcja magazynu danych

3.6.2. Orientacja na model domenowy zamiast na model danych

3.7. Fabryki

3.7.1. Walidacja

3.7.2. Logika biznesowa podczas składania obiektów

3.7.3. Wsparcie dla testability

3.8. Polityki (strategie)

3.8.1. Modelowanie w stylu funkcyjnym

3.8.2. Open Close Principle (SOLID) w praktyce

3.8.3. Podejście Supple Design

3.8.4. Dekorowanie

3.8.5. Umiejscowienie w 4 poziomach modelu

3.9. Zdarzenia biznesowe

3.9.1. Decoupling Bounded Context

3.9.2. Anticorruption Layer

3.10. Specyfikacje

3.10.1. Modelowanie złożonych warunków biznesowych

3.11. Dodatkowe wzorce - rozszerzenia Building Blocks

3.11.1. Sagi - modelowanie złożonych procesów biznesowych

3.11.2. Dekoratory Polityk - Supple Design

3.11.3. Agregat jako maszyna stanów

3.11.4. Łańcuch odpowiedzialności

3.11.5. Budowniczy

3.12. Praktyczne przykłady modelowania biznesowego z wykorzystaniem Building Blocks

## 4. Wzorce Strategiczne

4.1. Domain distillation - techniki wyłaniania

4.1.1. Core Domain

4.1.2. Supporting Domain

4.1.3. Generic Domain

4.2. Bounded Context - separacja i integracja

4.3. Shared Kernel - najlepsze praktyki

4.4. Conformist - kiedy warto stosować

4.5. Cztery poziomy modelu dużej skali

4.5.1. Capability

4.5.2. Operations

4.5.3. Policy

4.5.4. Decision Support

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

5.1. Zwiększanie poziomu czytelności kodu poprzez Domain Specific Language

5.2. Podział logiki na aplikacyjną i domenową

5.3. Podejście warstwowe - rozmieszczenie building blocks na warstwach

5.3.1. Warstwa interfejsów (prezentacji)

5.3.2. Warstwa logiki - rozwarstwienie na dwie warstwy logiki

5.3.2.1. Logika aplikacyjna (API)

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

5.3.2.3. Zapis Domain Story w formie czytelnej prozy

5.3.2.4. Logika domenowa (Building Blocks DDD)

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

5.3.3. Warstwa infrastruktury

5.3.3.1. Dostęp do danych

5.3.3.2. Infrastruktura techniczna

5.4. Przełożenie Use Case/User Story na warstwę aplikacji

5.5. Przełożenie modelu biznesowego na building blocks warstwy domenowej

5.6. Modelowanie Bounded Context

5.6.1. Decoupling

5.6.2. Integracja

## 6. Architektura systemu

6.1. Strategiczne projektowanie

6.1.1. Określenie Core Domain

6.1.2. Integracja z domenami Generic i Supporting

6.2. Zarys architektury integracji modułów

6.2.1. Event Driven Architecture

6.2.2. SOA