

Program szkolenia:

Refaktoring legacy do Domain Driven Design

Informacje:

Nazwa:	Refaktoring legacy do Domain Driven Design
Kod:	DDD-refaktoring
Kategoria:	Domain Driven Design
Grupa docelowa:	developerzy architekci analitycy
Czas trwania:	3 dni
Forma:	40% wykłady / 60% warsztaty

Podczas szkolenia uczestnicy poznają techniki umożliwiające refaktoryzację istniejących systemów w celu zmniejszenia nakładu pracy związanego z ich utrzymaniem.

Równoległym celem refaktoryzacji jest **wsteczna inżynieria modelu**, który nie był utrzymywany i został rozmyty.

Dodatkowym celem będzie poznawanie zachowania i struktury w celu naprawiania błędów i wprowadzania nowych funkcjonalności oraz refaktoryzacja designu.

Podczas warsztatów praktycznych uczestnicy będą refaktoryzować istniejący kod i pisać testy regresyjne. Uczestnicy zrefaktoryzują istniejący system pod kątem wprowadzenia nowych funkcjonalności.

Szkolenia powinny zostać poprzedzone szkoleniami z zakresu: [Technik modelowanie DDD](#) oraz [Technik implementacji DDD](#).

Zalety szkolenia:

- Wzrost morale przez odzyskanie wiary w to, że jest jeszcze szansa...

Szczegółowy program:

1. Czytanie kodu

- 1.1. Zbieranie i interpretowanie metryk
- 1.2. Wykrywanie punktów krytycznych systemu
- 1.3. Przegląd kodu i identyfikowanie "zapachów"

2. Podstawy refaktoryzacji

- 2.1. Podstawowe techniki rafaktoryzacji
- 2.2. Wsparcie ze strony IDE
- 2.3. Identyfikowanie szwów i rozdzielanie zależności
- 2.4. Zaawansowane refaktoringi w wielu krokach
- 2.5. Tworzenie planu refaktoringu
- 2.6. Refaktoryzacja designu
- 2.7. Refaktoryzacja do wzorca

3. Wykorzystanie technik modelowania w procesie refaktoringu

- 3.1. Zmiana podejścia w celu redukcji ilości reguł jakie należy brać pod uwagę
 - 3.1.1. podejście "od domeny" zamiast podejścia "od procesu"
- 3.2. Techniki lingwistyczne
 - 3.2.1. Wyłanianie Domain Story
 - 3.2.2. Techniki pełnych zdań zamiast zbierania rzeczowników
 - 3.2.3. Eksploracja domeny przy pomocy zdań podmiot.orzeczenie(dopełnienie, przydawka)
 - 3.2.4. Gibberish Game - usuwanie dwuznaczności i odkrywanie nowych koncepcji domenowych
 - 3.2.5. Słowo-Znaczenie(Kontekst)-Reguły
 - 3.2.5.1. Odwrócenie kolejności w celu odkrywania ukrytych koncepcji domenowych

3.3. Techniki wizualizacji

3.3.1. Grupowanie operacji wokół niezmienników

3.3.2. Metafory wizualne realnych Agregatów

3.3.3. Poziomy modelu

3.3.4. Separacja modelu pod kątem podatności na zmiany i niestabilności

4. Rozwarstwienie logiki jako główna strategia refaktoryzacji

4.1. Wyłanianie Use Case/User Story w warstwie aplikacji

4.1.1. Projektowanie API systemu

4.1.2. Hermetyzacja tego CO system powinien robić w serwisie aplikacyjne

4.2. Wyłanianie modelu biznesowego z building blocks warstwy domenowej

4.2.1. Hermetyzacja tego JAK i DLACZEGO system się tak zachowuje w Building Blocks DDD

4.2.2. Początek pracy - wyłanianie Value Objects

4.2.2.1. Praca nad słownictwem domenowym

4.2.2.2. Zarządzanie wartością jest łatwiejsze niż encją

4.2.3. Wyłanianie Agregatów

4.2.3.1. Grupowanie atrybutów ze względu na spójną zmianę w Use Case

4.2.3.2. Grupowanie atrybutów ze względu na ochronę niezmienników

4.2.4. Serwisy Domenowe - wyłanianie pod-procedur biznesowych

4.2.5. Polityki - hermetyzacja zmienności poza stabilnym interfejsem

4.2.6. Fabryki - hermetyzacja kreacji obiektów w jednym miejscu

4.2.7. Cztery poziomy modelu ze względu na podatność na zmiany

4.2.7.1. Capability

4.2.7.2. Operations

4.2.7.3. Policy

4.2.7.4. Decision Support

4.2.8. Szukanie stabilności

4.2.8.1. Open-close Principle w praktyce

4.2.8.2. Model stabilny - Agregaty i Serwisy Domenowe

4.2.8.3. Domknięcia modelu - polityki

4.2.8.4. Podejście funkcyjne

4.2.8.5. Wybór domknięć - fabryki

4.2.8.6. Hermetyzacja złożoności w jednym miejscu

5. Testowanie regresyjne

5.1. Sposoby testowania systemu

5.2. Rodzaje testów i przykłady ich wykorzystania

5.3. Automatyzacja procesu testowania

5.4. Wybór strategii testowania w projekcie

5.5. Pisanie testów automatycznych w projekcie, który ich nie posiada