BO·TT·EGA
IT minds

Program szkolenia:

Pragmatyczna refaktoryzacja z użyciem technik Domain Driven Design

Informacje:

Nazwa:	**Pragmatyczna refaktoryzacja z użyciem technik Domain Driven Design**
Kod:	**Craft-refactor**
Kategoria:	Wzorce i Craftsmanship
Odbiorcy:	developerzy
Czas trwania:	3 dni
Forma:	50% wykłady / 50% warsztaty

Przekształcenia “Extract Method”, “Replace If with Guardian” oraz “Extract Object” poprawiają czytelność kodu, ale nie ratują projektów, których bazowy model jest zły. Jak w bezpieczny i pragmatyczny sposób dokonać refaktoryzacji, która przypomina wymianę silnika lecącego samolotu? Poniższy warsztat przedstawia szereg typowych problemów projektowych skategoryzowanych na bazie produkcyjnego, audytorskiego i konsultacyjnego doświadczenia z szeregu projektów z różnych dziedzin biznesowych.

Nie jest to kolejny warsztat, w którym krok po kroku przechodzimy po katalogu typowych “zapachów” kodu i sposobów ich eliminacji. Podczas zajęć analizujemy też architektoniczne problemy, których naprawa wykracza poza standardowe ramy refaktoryzacji, ale których bezpieczne wykonanie może owocować dostarczeniem wartości biznesowej bez 3-miesięcznego przestoju. Uczestnik zdobywa konkretną mapę nawigacyjną wraz z narzędziami pozwalającymi mu skutecznie identyfikować i eliminować wieloletnie problemy występujące w systemie. Wszystko to w sposób bezpieczny, poparty dobrymi praktykami inżynierskimi, Domain-Driven Design oraz możliwością odwrotu.

Na warsztacie uczestnik poznaje sposoby rozmowy z odbiorcami biznesowymi, którzy nie do końca są przekonani do podejścia refaktoryzacji. Zajęcia obfitują w konkretne metryki biznesowe, budujące wspólne zrozumienie pomiędzy zespołem deweloperskim, a interesariuszami biznesowymi.

Po warsztacie, uczestnik m. in. potrafi identyfikować i naprawiać problemy architektoniczne i implementacyjne wykorzystując podejście Domain-Driven Design, analizę historii repozytorium, analizę metryk i stabilności kodu, podejście Test-Driven Development, podejście obiektowe/funkcyjne, modularyzację i wiele innych.

Warsztat może się odbyć wersji od trzydniowej do pięciodniowej. Czwarty lub/i piąty dzień poświęcamy na analizę domenową nowego projektu, która służy jako klamra wiedzy stosowanej przez trzy pierwsze dni w systemie Legacy. Alternatywnie warsztat może odbyć się jako konsultacja na projekcie klienta.

BOTTEGA IT minds

Szczegółowy program:

1. Typowe przypadki

- 1.1. Co zrobić z duża i nieestetyczną klasą?
- 1.2. Co zrobić z dużą i estetyczną klasą?
- 1.3. Jak naprawić odwieczny problem - brak spójności danych?
- 1.4. Moja klasa/moduł ma bardzo dużo zależności i bardzo dużo logiki biznesowej, jak to podzielić?
- 1.5. Bazowy model mojego modułu jest niedopasowany do potrzeb biznesu, co zrobić?
- 1.6. Jak naprawić niewydajne i bardzo złożone odczyty danych?
- 1.7. W moim projekcie brak jasno wydzielonych granic, jak to naprawić?

2. Sprawdzona Ścieżka Refaktoryzacji

- 2.1. Odkrywanie i opisywanie problemu
  - 2.1.1. Drivery architektoniczne
- 2.2. “Sprzedawanie” i marketing rozwiązań
  - 2.2.1. Odbiorca techniczny
    - 2.2.1.1. Egoless programming
  - 2.2.2. Odbiorca biznesowy
    - 2.2.2.1. Wektor nowej funkcji biznesowej
    - 2.2.2.2. Wektor nowej funkcji dla nowych klientów
    - 2.2.2.3. Wektor nowej funkcji dla jednego z nowych klientów
    - 2.2.2.4. Odwracanie zmian
- 2.3. Lokalizacja prawdziwych problemów
  - 2.3.1. Zbieranie i interpretowanie metryk
  - 2.3.2. Wykrywanie punktów krytycznych
  - 2.3.3. Identyfikacja “zapachów”

BO·TT·EGA
IT minds

2.3.4. Kod stabilny vs kod zmieniany

2.3.5. Analiza historii repozytorium

2.3.6. Podejście Domain-Driven Design

2.4. Planowanie zmian

2.4.1. Efekt "Brzydkiego Kaczątka"

2.4.2. Problem "You are never done"

2.5. Wprowadzanie zmian

2.5.1. Refaktoryzacja mechaniczna/taktyczna

2.5.2. Refaktoryzacja architektoniczna/strategiczna

2.6. Wdrażanie zmian

2.6.1. Cykle i mikrocykle

2.7. Obserwacja efektów

2.7.1. Bezpieczny odwrót

3. Refaktoring ciągły

3.1. Wsparcie ze strony IDE

3.2. Identyfikowanie szwów

3.2.1. Wspólne zrozumienie i Edukacja

3.2.1.1. Code Review

3.2.1.2. ArchUnit i inne narzędzia

3.3. Konwencje jako driver architektoniczny

3.4. Refaktoring bez znajomości problemu biznesowego

3.4.1. Jak to robić bezpiecznie?

4. Refaktoring przygotowujący

4.1. Nowe funkcje biznesowe

4.2. Naprawa obecnych funkcji biznesowych

BOTTEGA IT minds

- 4.3. Naprawa atrybutów jakościowych
- 4.4. Budowanie zaufania
- 4.5. Metryki biznesowe jako element marketingu refaktoryzacji

5. Wdrażanie zmian

- 5.1. Testy
  - 5.1.1. Odkrywanie szwów
  - 5.1.2. Technika ułatwiania testów poprzez obniżanie szwów
  - 5.1.3. Testy charakterystyki systemu
  - 5.1.4. Testy i szwy jako odkrywanie granic modułów i uwalnianie potencjałów biznesowych
- 5.2. Problem Big Bang Release
- 5.3. Technika Step by Step
- 5.4. Parallel Models jako jedyny sposób bezpiecznej strategicznej refaktoryzacji
  - 5.4.1. Bezpieczny odwrót
  - 5.4.2. Logi
  - 5.4.3. Zaawansowane metryki
  - 5.4.4. Samoleczący się system
  - 5.4.5. Możliwość codziennego budowanie zaufania do interesariusza projektowego

6. Wykorzystanie Domain-Driven Design

- 6.1. Hermetyzacja tego CO system powinien robić w serwisy aplikacyjne
- 6.2. Hermetyzacja tego JAK i DLACZEGO system się tak zachowuje w Building Blocks DDD
- 6.3. Praca nad słownictwem domenowym
- 6.4. Zarządzanie wartością jest łatwiejsze niż encją
- 6.5. Odkrywanie Value Objects, Agregatów oraz Polityk
- 6.6. Polityki - hermetyzacja zmienności poza stabilnym interfejsem

BO·TT·EGA
IT minds

6.7. Rozwijalność i czytelność systemu poprzez trójpodział logiki

6.7.1. Operations

6.7.2. Policy

6.7.3. Decision Support

6.8. Podejście funkcyjne

6.9. Hermetyzacja złożoności w jednym miejscu

6.10. Podejście "od domeny" zamiast podejścia "od procesu"

6.11. Separacja modelu pod kątem podatności na zmiany i niestabilności

7. Wprowadzanie zmian

7.1. Poziomy modelu

7.2. Zmiana API

7.2.1. API publiczne vs API prywatne

7.3. Coupling, Coupling Logiczny i Data Coupling

8. Modelowanie Domeny (tylko w wersji 4 dniowej)

8.1. Odkrywanie złożoności za pomocą Event Stormingu

8.1.1. “As Is” vs “To Be”

8.1.2. Analiza problematycznych miejsc

8.1.3. Identyfikacja problematycznych miejsc w kodzie

8.1.4. Odkrywanie słownictwa domenowego

8.1.5. Destylacja archetypowych problemów

8.1.6. Modularyzacja do Bounded Contextów

8.1.6.1. Mapowanie do obecnego systemu

8.1.7. Destylacja agregatów

8.1.7.1. Mapowanie do obecnego systemu