

O'REILLY®

Koncepcja Domain-Driven Design

Dostosowywanie architektury aplikacji
do strategii biznesowej



Helion 

Vlad Khononov

Tytuł oryginału: Learning Domain-Driven Design: Aligning Software Architecture and Business Strategy

Tłumaczenie: Radosław Meryk

ISBN: 978-83-283-9262-5

© 2022 Helion S.A.

Authorized Polish translation of the English *Learning Domain-Driven Design*
ISBN 9781098100131 © 2022 Vladislav Khononov.

This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to publish and sell the same.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz wydawca dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz wydawca nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<https://helion.pl/user/opinie/kododr>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:

<https://ftp.helion.pl/przyklady/kododr.zip>

Helion S.A.

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 231 22 19, 32 230 98 63

e-mail: helion@helion.pl

WWW: <https://helion.pl> (księgarnia internetowa, katalog książek)

Printed in Poland.

- Kup książkę
- Poleć książkę
- Oceń książkę

- Księgarnia internetowa
- Lubię to! » Nasza społeczność

Spis treści

Słowo wstępne	13
Przedmowa	15
Wprowadzenie	23
Część I. Projektowanie strategiczne	25
1. Analiza Dziedzin Biznesowych	27
Co to jest Dziedzina Biznesowa?	27
Co to jest Poddziedzina?	28
Typy Poddziedzin	28
Porównywanie Poddziedzin	31
Identyfikowanie granic Poddziedzin	34
Przykłady analizy Dziedziny	37
Gigmaster	37
BusVNext	39
Kim są eksperci dziedzinowi?	40
Podsumowanie	41
Ćwiczenia	42
2. Odkrywanie wiedzy dziedzinowej	43
Problemy biznesowe	43
Odkrywanie wiedzy	44
Komunikacja	44
Co to jest Język Wszechobecny?	46
Język biznesu	46
Scenariusze	47
Spójność	47

Model Dziedziny Biznesowej	48
Co to jest „model”?	48
Skuteczne modelowanie	49
Modelowanie Dziedziny Biznesowej	49
Ciągły wysiłek	50
Narzędzia	50
Wyzwania	51
Podsumowanie	52
Ćwiczenia	52
3. Zarządzanie złożonością Dziedziny	54
Niespójne modele	54
Co to jest Kontekst Ograniczony?	56
Granice modelu	56
Język Wszegobećny dokładniej	57
Zakres Kontekstu Ograniczonego	57
Konteksty Ograniczone a Poddziedziny	59
Poddziedziny	59
Konteksty Ograniczone	59
Wzajemne oddziaływanie między Poddziedzinami a Kontekstami Ograniczonymi	59
Granice	61
Granice fizyczne	61
Granice własności	62
Konteksty Ograniczone w prawdziwym życiu	62
Dziedziny semantyczne	63
Nauka	63
Kupowanie lodówki	64
Podsumowanie	66
Ćwiczenia	66
4. Integracja Kontekstów Ograniczonych	68
Kooperacja	68
Partnerstwo	69
Wspólne Jądro	69
Klient-Dostawca	71
Konformista	72
Warstwa Antykorupcyjna	72
Usługa Otwartego Hosta	73
Różne Drogi	74
Problemy z komunikacją	75
Poddziedziny Ogólne	75
Różnice w modelach	75

Mapa Kontekstu	75
Utrzymanie	76
Ograniczenia	76
Podsumowanie	77
Ćwiczenia	77

Część II. Projektowanie taktyczne **79**

5. Implementacja prostej Logiki Biznesowej	81
Skrypt Transakcji	81
Implementacja	82
To nie jest takie proste!	82
Kiedy używać Skryptu Transakcji	86
Aktywny Rekord	86
Implementacja	87
Kiedy używać wzorca Aktywny Rekord	88
Bądź pragmatyczny	89
Podsumowanie	89
Ćwiczenia	89
 6. Rozwiązywanie problemów ze złożoną Logiką Biznesową	 91
Historia	91
Model Dziedziny	92
Implementacja	92
Bloki konstrukcyjne	93
Zarządzanie złożonością	108
Podsumowanie	109
Ćwiczenia	110
 7. Modelowanie wymiaru czasu	 112
Event Sourcing	112
Wyszukiwanie	117
Analiza	118
Źródło prawdy	119
Magazyn zdarzeń	119
Model Dziedziny ze źródłem w postaci zdarzeń	120
Zalety	122
Wady	123
Często zadawane pytania	124
Wydajność	124
Usuwanie danych	126
Dlaczego nie mogą po prostu...?	126

Podsumowanie	127
Ćwiczenia	127
8. Wzorce architektoniczne	129
Logika Biznesowa a wzorce architektoniczne	129
Architektura Warstwowa	130
Warstwa Prezentacji	130
Warstwa Logiki Biznesowej	131
Warstwa Dostępu Do Danych	131
Komunikacja między warstwami	132
Odmiany	132
Kiedy używać Architektury Warstwowej?	136
Porty i Adaptery	137
Terminologia	137
Zasada inwersji zależności	137
Integracja komponentów infrastrukturalnych	138
Odmiany	139
Kiedy używać wzorca Porty i Adaptery?	139
Segregacja Odpowiedzialności za Polecenia i Zapytania	140
Modelowanie poliglotyczne	140
Implementacja	140
Projekcje Modeli Odczytu	141
Wyzwania	143
Segregacja modeli	144
Kiedy używać CQRS?	144
Zakres	145
Podsumowanie	146
Ćwiczenia	146
9. Wzorce komunikacji	148
Tłumaczenie modelu	148
Bezstanowe tłumaczenie modelu	149
Stanowe tłumaczenie modelu	151
Integracja Agregatów	153
Skrzynka Nadawcza	155
Saga	157
Menedżer Procesu	160
Podsumowanie	163
Ćwiczenia	163

10. Heurystyki projektowe	167
Heurystyka	167
Konteksty Ograniczone	167
Wzorce implementacji Logiki Biznesowej	169
Wzorce architektoniczne	171
Strategia testowania	171
Piramida Testowania	172
Romb Testowania	173
Odwrócona Piramida Testowania	173
Drzewo decyzyjne projektu taktycznego	173
Podsumowanie	175
Ćwiczenia	175
11. Zmieniające się decyzje projektowe	176
Zmiany w Dziedzinach	176
Podstawowa na Ogólną	177
Ogólna na Podstawową	177
Pomocnicza na Ogólną	178
Pomocnicza na Podstawową	178
Podstawowa na Pomocniczą	178
Ogólna na Pomocniczą	178
Problemy projektu strategicznego	179
Problemy projektu taktycznego	180
Skrypt Transakcji na Aktywny Rekord	180
Aktywny Rekord na Model Dziedziny	180
Model Dziedziny na Model Dziedziny ze źródłem w postaci zdarzeń	182
Generowanie przeszłych przejęć	182
Modelowanie zdarzeń migracji	183
Zmiany organizacyjne	184
Partnerstwo na Klient-Dostawca	185
Klient-Dostawca na Różne Drogi	185
Wiedza dziedzinowa	185
Rozwój	186
Poddziedziny	187
Konteksty Ograniczone	187
Agregaty	188
Podsumowanie	188
Ćwiczenia	189

12. EventStorming	191
Co to jest EventStorming?	191
Kto powinien wziąć udział w warsztatach EventStorming?	191
Czego potrzebujesz do zorganizowania sesji EventStorming?	192
Przebieg warsztatów EventStorming	193
Krok 1. Nieustrukturyzowana eksploracja	193
Krok 2. Osie czasu	194
Krok 3. Możliwe problemy	195
Krok 4. Kluczowe zdarzenia	195
Krok 5. Polecenia	196
Krok 6. Reguły	197
Krok 7. Modele Odczytu	197
Krok 8. Systemy zewnętrzne	197
Krok 9. Agregaty	199
Krok 10. Konteksty Ograniczone	199
Odmiany	200
Kiedy korzystać z warsztatów EventStorming?	200
Wskazówki dotyczące facylitacji	201
Obserwuj dynamikę	202
Zdalne sesje EventStorming	202
Podsumowanie	202
Ćwiczenia	203
13. Projekt oparty na Dziedzinie w praktyce	204
Analiza strategiczna	205
Opis Dziedziny Biznesowej	205
Poznaj aktualny projekt	206
Strategia modernizacji	207
Modernizacja strategiczna	208
Modernizacja taktyczna	209
Pielęgnuj Język Wszechobecny	210
Praktyczny projekt DDD	213
Promowanie projektu DDD	213
Ujawnij metodologię DDD	213
Podsumowanie	215
Ćwiczenia	216

14. Mikrouslugi	219
Co to jest usługa?	219
Co to jest „mikrousluga”?	220
Metoda jako usługa: czy to doskonale mikrouslugi?	221
Cel projektu	222
Złożoność systemu	223
Mikrouslugi jako usługi głębokie	224
Mikrouslugi jako moduły głębokie	225
Projektowanie DDD a granice mikrouslug	226
Konteksty Ograniczone	226
Agregaty	229
Poddziedziny	229
Kompresowanie publicznych interfejsów mikrouslug	230
Usługa Otwartego Hosta	230
Warstwa Antykorupcyjna	231
Podsumowanie	232
Ćwiczenia	232
15. Architektura sterowana zdarzeniami	234
Architektura sterowana zdarzeniami	234
Zdarzenia	235
Zdarzenia, Polecenia i Komunikaty	235
Struktura	236
Rodzaje zdarzeń	236
Projektowanie integracji sterowanej zdarzeniami	241
Rozproszona Wielka Kula Błota	241
Sprzężenie czasowe	242
Sprzężenie funkcjonalne	243
Sprzężenie implementacji	243
Refaktoryzacja kodu integracji sterowanej zdarzeniami	243
Heurystyka projektu sterowanego zdarzeniami	244
Podsumowanie	246
Ćwiczenia	246
16. Siatka danych	248
Analityczny model danych a transakcyjny model danych	248
Tabela faktów	249
Tabela wymiarów	250
Modele analityczne	251

Platformy zarządzania danymi analitycznymi	253
Hurtownia danych	253
Jezioro danych	255
Wyzwania związane z architekturami hurtowni danych i jeziora danych	257
Siatka danych	257
Dekompozycja danych wokół Dziedzin	257
Dane jako produkt	258
Zadbaj o autonomię	260
Zbuduj ekosystem	260
Łączenie siatki danych z projektowaniem DDD	260
Podsumowanie	262
Ćwiczenia	262
Posłowie	264
A. Zastosowanie DDD: studium przypadku	269
B. Odpowiedzi na pytania	283
Bibliografia	289

Integracja Kontekstów Ograniczonych

Wzorzec Kontekstów Ograniczonych nie tylko chroni spójność Języka Wszechobecnego, ale także umożliwia modelowanie. Nie można zbudować modelu bez określenia jego celu — jego granic. Granica dzieli odpowiedzialność języków. Język w jednym Kontekście Ograniczonym może modelować Dziedzinę Biznesową w celu rozwiązania określonego problemu. Inny Kontekst Ograniczony może reprezentować te same encje biznesowe, ale modelować je w celu rozwiązania innego problemu.

Co więcej, modele w różnych Kontekstach Ograniczonych mogą być rozwijane i implementowane niezależnie. Trzeba jednak zapamiętać, że Konteksty Ograniczone same w sobie nie są niezależne. Tak jak nie można zbudować systemu z niezależnych komponentów — komponenty muszą ze sobą współdziałać, aby można było osiągnąć nadrzędne cele systemu — tak samo nie da się stworzyć niezależnych implementacji w Kontekstach Ograniczonych. Chociaż Konteksty Ograniczone mogą niezależnie ewoluować, muszą się ze sobą integrować. W efekcie zawsze będą istniały punkty styku pomiędzy Kontekstami Ograniczonymi. Są to tak zwane *kontrakty*.

Zapotrzebowanie na kontrakty wynika z różnic w modelach i językach Kontekstów Ograniczonych. Ponieważ każdy kontrakt dotyczy więcej niż jednej strony, należy je zdefiniować i skoordynować. Ponadto z definicji wynika, że dwa Konteksty Ograniczone korzystają z różnych Języków Wszechobecných. Który język będzie wykorzystany do celów integracji? W projekcie rozwiązania należy oszacować te problemy i wziąć je pod uwagę.

W tym rozdziale zaprezentuję wzorce DDD dotyczące definiowania relacji i integracji między Kontekstami Ograniczonymi. Wzorce te wynikają z charakteru współpracy pomiędzy zespołami pracującymi nad Kontekstami Ograniczonymi. Wzorce te zostaną podzielone na trzy grupy, z których każda reprezentuje rodzaj współpracy zespołowej: Kooperacja, Klient-Dostawca oraz Różne Drogi.

Kooperacja

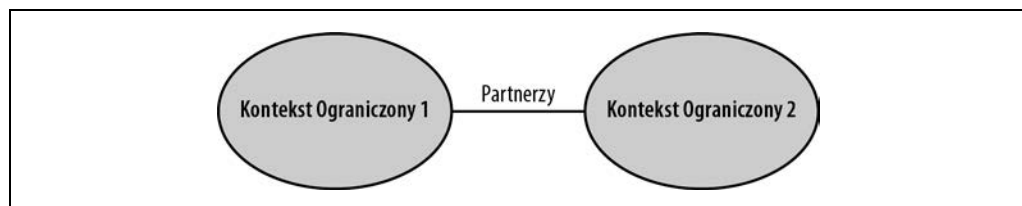
Wzorce Kooperacji dotyczą Kontekstów Ograniczonych implementowanych przez zespoły o dobrze ugruntowanej komunikacji.

W najprostszym przypadku są to Konteksty Ograniczone zaimplementowane przez jeden zespół. Dotyczy to również zespołów z celami zależnymi od siebie, w których sukces jednego zespołu zależy od sukcesu drugiego i odwrotnie. Głównym kryterium w tym przypadku jest jakość komunikacji i współpracy pomiędzy zespołami.

Przyjrzyjmy się dwóm wzorcom DDD, odpowiednim dla współpracujących ze sobą zespołów: Partnerstwo i Wspólne Jądro.

Partnerstwo

W modelu Partnerstwa integracja między Kontekstami Ograniczonymi jest koordynowana w sposób doraźny. Gdy jeden zespół powiadomi inny zespół o zmianach w interfejsie API, drugi dostosuje się do tych zmian — bez dramatów i konfliktów (patrz rysunek 4.1).



Rysunek 4.1. Model Partnerstwa

Koordynacja integracji jest w tym przypadku dwukierunkowa. Żaden zespół nie dyktuje języka, który ma być używany do zdefiniowania kontraktu. Zespoły mogą pracować nad różnicami i wybierać najbardziej odpowiednie rozwiązanie. Ponadto obie strony współpracują przy rozwiązywaniu wszelkich możliwych problemów z integracją. Żaden zespół nie jest zainteresowany zablokowaniem drugiego.

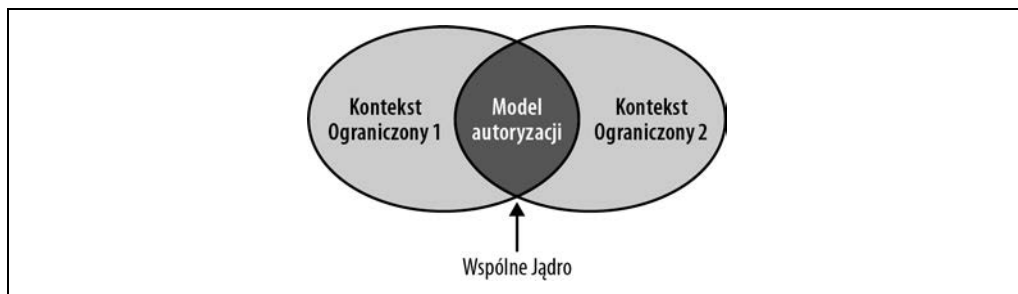
Do skutecznej integracji w takim przypadku potrzebne są ugruntowane praktyki współpracy, wysoki poziom zaangażowania i częste synchronizacje między zespołami. Z technicznego punktu widzenia potrzebna jest ciągła integracja zmian stosowanych przez oba zespoły, aby jeszcze bardziej zminimalizować pętle sprzężeń zwrotnych integracji.

Wzorzec Partnerstwa nie wydaje się odpowiedni dla zespołów rozproszonych geograficznie, ponieważ może stanowić wyzwanie dla synchronizacji i komunikacji.

Wspólne Jądro

Pomimo tego, że Konteksty Ograniczone są granicami modelu, nadal mogą wystąpić przypadki, w których ten sam model Poddziedziny lub jej część zostaną zaimplementowane w wielu Kontekstach Ograniczonych. Należy podkreślić, że wspólny model jest zaprojektowany zgodnie z potrzebami wszystkich Kontekstów Ograniczonych. Ponadto współdzielony model musi być spójny we wszystkich Kontekstach Ograniczonych, które go stosują.

Jako przykład rozważmy system korporacyjny, który używa dostosowanego do potrzeb modelu zarządzania uprawnieniami użytkowników. Uprawnienia każdemu użytkownikowi mogą zostać przyznane bezpośrednio lub mogą być odziedziczone po jednej z jednostek organizacyjnych, do których należy. Co więcej, każdy Kontekst Ograniczony może modyfikować model autoryzacji, a zmiany stosowane w każdym Kontekście Ograniczonym muszą wpływać na wszystkie inne Konteksty Ograniczone korzystające z modelu (patrz rysunek 4.2).



Rysunek 4.2. Wspólne Jądro

Wspólny zakres

Model nakładających się na siebie Kontekstów Ograniczonych łączy cykle życia występujących w nim Kontekstów. Zmiana wprowadzona w modelu współdzielonym ma natychmiastowy wpływ na wszystkie Konteksty Ograniczone należące do modelu. W związku z tym, aby zminimalizować kaskadowe skutki zmian, nakładanie w tym modelu powinno być ograniczone tylko do tej części modelu, która musi zostać zaimplementowana w obu Kontekstach Ograniczonych. Byłoby idealnie, gdyby Wspólne Jądro składało się tylko z kontraktów integracyjnych i struktur danych, które podczas przekazywania przekraczają granice Kontekstów Ograniczonych.

Implementacja

Wspólne Jądro jest zaimplementowane w taki sposób, że każda modyfikacja kodu źródłowego ma natychmiastowe odzwierciedlenie we wszystkich Kontekstach Ograniczonych, które z niego korzystają.

Jeśli w organizacji stosowane jest podejście mono-repozytorium, Wspólne Jądro mogą tworzyć pliki źródłowe, do których odwołuje się wiele Kontekstów Ograniczonych. Jeśli użycie współdzielonego repozytorium nie jest możliwe, Wspólne Jądro może zostać wyodrębnione do dedykowanego projektu i wywoływane w Kontekstach Ograniczonych jako współdzielona biblioteka. Tak czy inaczej, każda zmiana we Wspólnym Jądrze musi zainicjować przeprowadzenie testów integracyjnych we wszystkich Kontekstach Ograniczonych, które z niego korzystają.

Wymagana jest ciągła integracja zmian, ponieważ Wspólne Jądro należy do wielu Kontekstów Ograniczonych. Nierozpowszechnienie zmian Wspólnego Jądra we wszystkich powiązanych z nim Kontekstach Ograniczonych prowadzi do niespójności w modelu: Konteksty Ograniczone mogą polegać na przestarzałych implementacjach Wspólnego Jądra, co prowadzi do uszkodzenia danych i (lub) problemów ze środowiskiem wykonawczym.

Kiedy należy korzystać ze Wspólnego Jądra?

Nadrzędnym kryterium zastosowania wzorca Wspólnego Jądra są koszty powielenia w zestawieniu z kosztami koordynacji. Ponieważ ten wzorzec wprowadza silną zależność między wykorzystującymi go Kontekstami Ograniczonymi, powinien być stosowany tylko wtedy, gdy koszt powielania jest wyższy niż koszt koordynacji — innymi słowy, tylko wtedy, gdy integracja zmian wprowadzonych w modelu współdzielonym przez oba Konteksty Ograniczone będzie wymagała więcej wysiłku niż koordynowanie zmian we wspólnej bazie kodu.

Różnice pomiędzy kosztami integracji i duplikacji zależą od zmienności modelu. Im częściej model się zmienia, tym wyższe koszty integracji. Dlatego wzorec Wspólnego Jądra w naturalny sposób ma zastosowanie do Poddziedzin, które zmieniają się najczęściej, czyli Poddziedzin Podstawowych.

W pewnym sensie wzorec Wspólnego Jądra jest sprzeczny z zasadami Kontekstów Ograniczonych wprowadzonych w poprzednim rozdziale. Jeśli korzystające ze wzorca Konteksty Ograniczone nie są implementowane przez ten sam zespół, wprowadzenie Wspólnego Jądra jest sprzeczne z zasadą, która mówi, że właścicielem Kontekstu Ograniczonego powinien być jeden zespół. W modelu nakładających się Kontekstów Ograniczonych Wspólne Jądro jest w gruncie rzeczy rozwijane przez wiele zespołów.

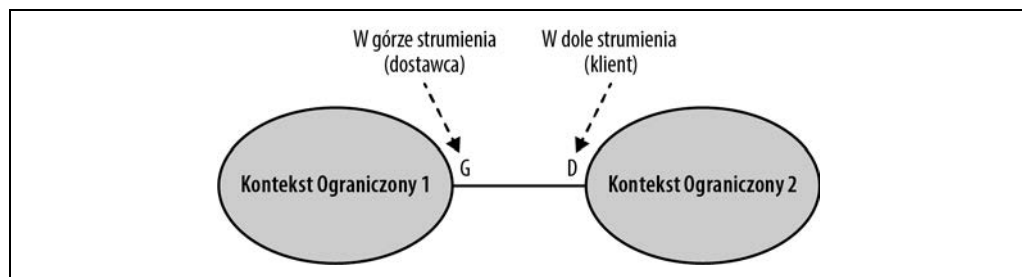
Z tego powodu użycie Wspólnego Jądra powinno być uzasadnione. Jest to pragmatyczny wyjątek, który należy dokładnie rozważyć. Typowym przypadkiem użycia przemawiającym za implementacją Wspólnego Jądra jest sytuacja, w której problemy z komunikacją lub współpracą uniemożliwiają wdrożenie wzorca Partnerstwo — na przykład z powodu ograniczeń geograficznych lub polityki organizacyjnej. Implementacja ściśle powiązanej funkcjonalności bez odpowiedniej koordynacji spowoduje problemy z integracją. Skutkiem będą rozsynchronizowane modele oraz spory o to, który model jest lepiej zaprojektowany. Minimalizacja zakresu Wspólnego Jądra decyduje o zakresie kaskadowych zmian, a inicjowanie testów integracyjnych dla każdej zmiany jest sposobem na wymuszenie wczesnego wykrywania problemów z integracją.

Innym powszechnym przypadkiem użycia wzorca Wspólnego Jądra, chociaż w takim kontekście jest zwykle wykorzystywany tymczasowo — jest stopniowa modernizacja starego systemu. W takim scenariuszu współdzielona baza kodu może być praktycznym rozwiązaniem pośrednim do stopniowej dekompozycji systemu na Konteksty Ograniczone.

Wreszcie, Wspólne Jądro może być dobrym rozwiązaniem integracji Kontekstów Ograniczonych, będących własnością jednego zespołu i zaimplementowanych przez ten sam zespół. W takim przypadku doraźna integracja Kontekstów Ograniczonych — Partnerstwo — może z czasem „wypłukać” granice Kontekstów. Wspólne Jądro może być użyte do jawnego definiowania kontraktów integracji Kontekstów Ograniczonych.

Klient-Dostawca

Drugą grupą wzorców współpracy, które omówię, są wzorce Klient-Dostawca. Jak widać na rysunku 4.3, jeden z Kontekstów Ograniczonych — dostawca — świadczy usługi swoim klientom. Usługodawca działa w „górze strumienia przetwarzania”, a klient lub konsument działa w „dole strumienia”.



Rysunek 4.3. Relacja Klient-Dostawca

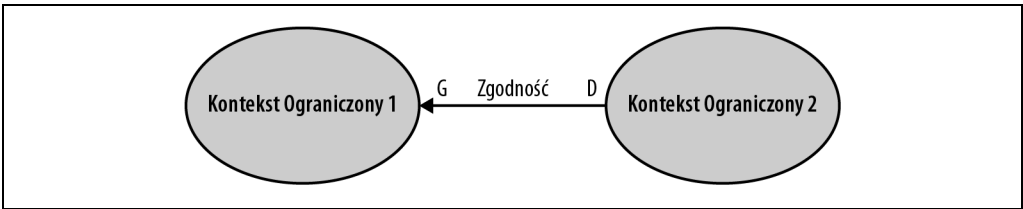
W przeciwieństwie do przypadku współpracy oba zespoły (w górze i w dole strumienia przetwarzania) mogą być skuteczne niezależnie od siebie. W związku z tym w większości przypadków mamy do czynienia z nierównowagą sił: kontrakt integracji może dyktować albo zespół w górze strumienia, albo zespół w dole strumienia.

W tym podrozdziale omówię trzy wzorce dotyczące następujących różnic możliwości: Konformista, Warstwa Antykorupcyjna i Usługa Otwartego Hosta.

Konformista

W niektórych przypadkach równowaga sił sprzyja zespołowi działającemu w górze strumienia przetwarzania, który nie ma realnej motywacji do wspierania potrzeb swoich klientów. Zamiast tego po prostu dostarcza kontrakt integracji zdefiniowany zgodnie z własnym modelem — skorzystaj z niego lub zrezygnuj. Taka nierównowaga sił może być spowodowana integracją z dostawcami usług, którzy pochodzą z zewnątrz organizacji, lub po prostu wynikać z zasad obowiązujących w organizacji.

Jeśli zespół działający w dole strumienia przetwarzania może zaakceptować model zespołu działającego w górze strumienia, relacja Kontekstów Ograniczonych nazywana jest *Konformistą*. Model Kontekstu Ograniczonego zespołu w dole strumienia przetwarzania jest zgodny z modelem Kontekstu Ograniczonego zespołu w górze strumienia przetwarzania (patrz rysunek 4.4).



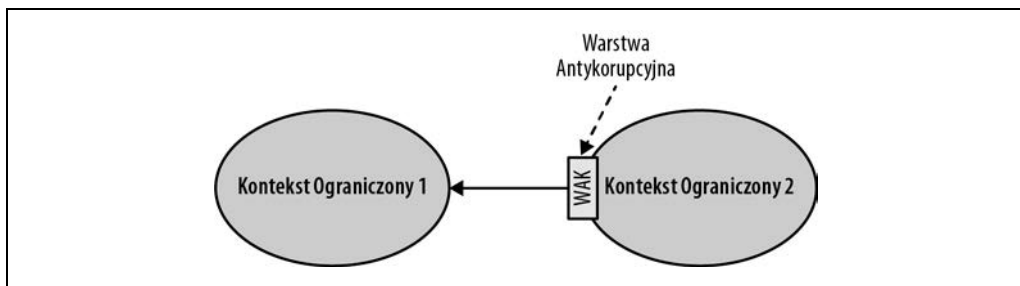
Rysunek 4.4. Relacja Konformisty

Decyzja zespołu z dołu strumienia przetwarzania o rezygnacji z części autonomii może być uzasadniona na wiele sposobów. Na przykład kontrakt udostępniony przez zespół działający w górze strumienia przetwarzania może być standardowym, dobrze ugruntowanym modelem lub może być po prostu wystarczająco dobry dla potrzeb zespołu działającego w dole strumienia.

Następny wzorzec odnosi się do przypadku, w którym Konsument nie chce zaakceptować modelu dostawcy.

Warstwa Antykorupcyjna

Podobnie jak we wzorcu Konformista, równowaga sił w tej relacji nadal jest wypaczona w kierunku usługi działającej w górze strumienia przetwarzania. Jednak w tym przypadku Kontekst Ograniczony nie jest skłonny do dostosowania się do wymogów usługi z góry strumienia. Zamiast tego może przetłumaczyć model Kontekstu Ograniczonego na model dostosowany do własnych potrzeb za pomocą Warstwy Antykorupcyjnej, tak jak pokazano na rysunku 4.5.



Rysunek 4.5. Integracja z wykorzystaniem Warstwy Antykorupcyjnej

Wzorzec Warstwy Antykorupcyjnej odnosi się do scenariuszy, w których dostosowanie się do modelu dostawcy nie jest pożądane lub warte wysiłku. Mogą to być następujące sytuacje:

Gdy Kontekst Ograniczony usługi w dole strumienia zawiera Poddziedzinę Podstawową

Model Poddziedziny Podstawowej wymaga dodatkowej uwagi, a przestrzeganie modelu dostawcy może utrudnić modelowanie Dziedziny Problemu.

Gdy model usługi w górze strumienia jest z perspektywy potrzeb konsumenta nieskuteczny lub niewygodny

Jeśli Kontekst Ograniczony dostosowuje się do bałaganu, istnieje ryzyko, że sam stanie się bałaganem. Dzieje się tak często w przypadku integracji ze starszymi systemami.

Gdy kontrakt dostawcy często się zmienia

Konsument chce chronić swój model przed częstymi zmianami. W przypadku Warstwy Antykorupcyjnej zmiany w modelu dostawcy wpływają jedynie na mechanizmy translacji.

Warto też pamiętać, że podczas modelowania tłumaczenie modelu dostawcy izoluje konsumenta działającego w dole strumienia od obcych pojęć, które nie są istotne dla jego Kontekstu Ograniczonego. W ten sposób upraszcza Język Wszechobecny i model konsumenta.

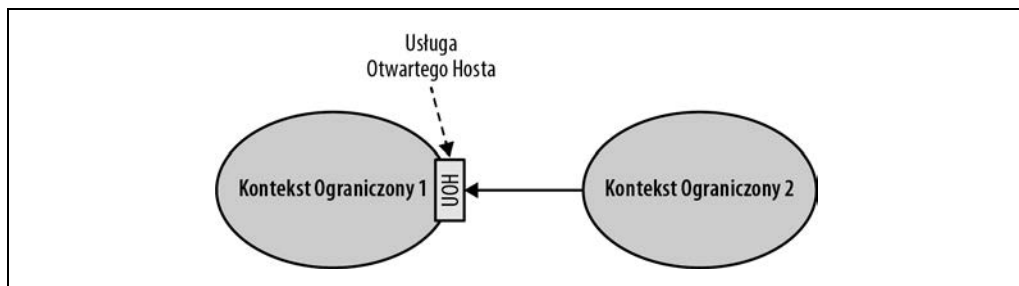
Różne sposoby implementacji Warstwy Antykorupcyjnej omówię w rozdziale 9.

Usługa Otwartego Hosta

Ten wzorzec odnosi się do przypadków, w których większe możliwości są po stronie konsumentów. Dostawca jest zainteresowany ochroną swoich konsumentów i zapewnieniem najlepszej możliwej obsługi.

Aby chronić konsumentów przed zmianami w modelu implementacji, działający w górze strumienia dostawca oddziela model implementacji od publicznego interfejsu. To rozdzielenie umożliwia dostawcy rozwijanie swoich modeli implementacji i interfejsów publicznych w różnym tempie (patrz rysunek 4.6).

Publiczny interfejs dostawcy nie powstał po to, by dostosować się do swojego Języka Wszechobecnego. Powstał, by ujawnić protokół wygodny dla konsumentów, wyrażony w języku zorientowanym na integrację. W związku z tym protokół publiczny nazywany jest *Językiem Opublikowanym*.

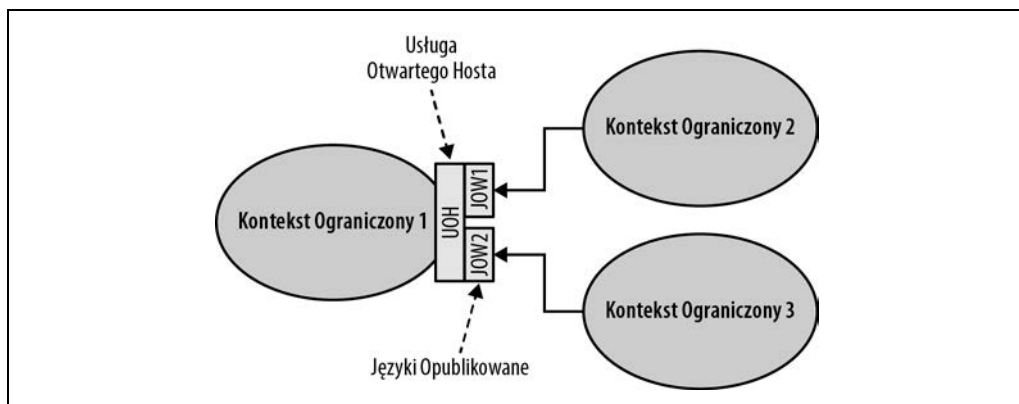


Rysunek 4.6. Integracja za pośrednictwem Usługi Otwartego Hosta

W pewnym sensie wzorzec Usługi Otwartego Hosta jest odwróceniem wzorca Warstwy Antykorupcyjnej: to nie konsument, a dostawca implementuje tłumaczenie swojego wewnętrznego modelu.

Oddzielenie modeli implementacji i integracji Kontekstu Ograniczonego daje Kontekstowi Ograniczonemu w górze strumienia przetwarzania swobodę rozwijania swojej implementacji bez wpływu na Konteksty w dole strumienia. Oczywiście jest to możliwe tylko wtedy, gdy zmodyfikowany model implementacji można przetłumaczyć na Język Opublikowany, z którego już korzystają konsumenci.

Ponadto oddzielenie modelu integracji umożliwia Kontekstowi w górze strumienia przetwarzania udostępnienie wielu wersji Języka Opublikowanego jednocześnie, co pozwala konsumentom na przeprowadzenie migracji do nowej wersji stopniowo (patrz rysunek 4.7).



Rysunek 4.7. Usługa Otwartego Hosta udostępniająca wiele wersji Języka Opublikowanego

Różne Drogi

Ostatnią opcją współpracy jest całkowity jej brak. Wzorzec ten może być stosowany z różnych powodów w przypadkach, gdy zespoły nie chcą lub nie mogą ze sobą współpracować. Przyjrzymy się kilku takim przypadkom.

Problemy z komunikacją

Częstym powodem unikania współpracy są trudności komunikacyjne spowodowane wielkością organizacji lub polityką wewnętrzną. Gdy zespoły mają trudności ze współpracą i uzgodnieniami, więcej korzyści może pojawić się wtedy, gdy każdy z zespołów pójdzie we własnym kierunku i powieli funkcjonalności w wielu Kontekstach Ograniczonych.

Poddziedziny Ogólne

Powodem, dla którego zespoły decydują się na pójście własną drogą, może być również charakter zdublowanej Poddziedziny. Gdy określona Poddziedzina jest Ogólna i jeśli rozwiązanie ogólne jest łatwe do zintegrowania, bardziej opłacalne może być zintegrowanie go lokalnie w każdym Kontekście Ograniczonym z osobna. Przykładem może być framework logowania. Nie miałoby sensu, aby jeden z Kontekstów Ograniczonych udostępniał go jako usługę. Uciążliwości związane z dodatkową złożonością integracji takiego rozwiązania przeważałyby nad korzyściami wynikającymi z niepowielania funkcjonalności w wielu Kontekstach. Powielanie funkcjonalności byłoby tańsze niż współpraca.

Różnice w modelach

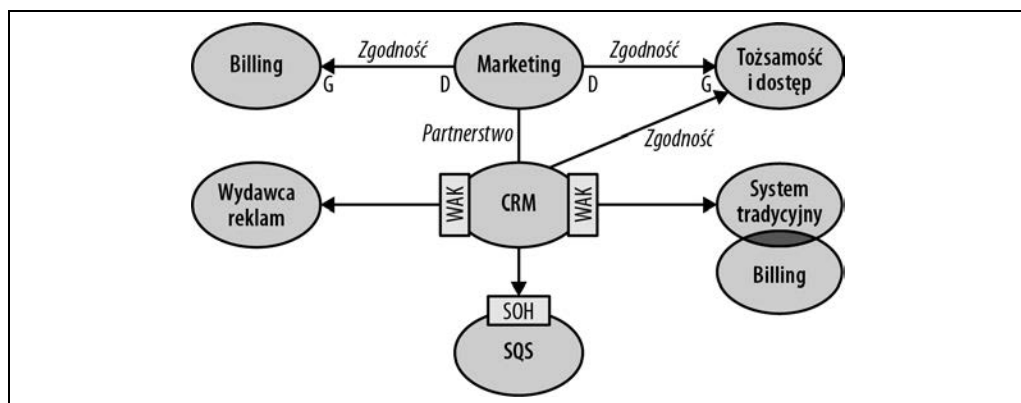
Powodem zastosowania Różnych Dróg mogą być również różnice w modelach Kontekstów Ograniczonych. Modele mogą różnić się tak bardzo, że relacja konformistyczna stanie się niemożliwa, a implementacja Warstwy Antykorupcyjnej byłaby droższa niż powielanie funkcjonalności. W takim przypadku również bardziej opłaca się, aby zespoły poszły oddzielnymi drogami.



Należy unikać wzorca Różne Drogi w przypadku integracji Poddziedzin Podstawowych. Powielanie implementacji takich Poddziedzin byłoby sprzeczne ze strategią firmy, polegającą na ich implementacji w sposób najbardziej efektywny i najbardziej zoptymalizowany.

Mapa Kontekstu

Po przeanalizowaniu wzorców integracji między Kontekstami Ograniczonymi systemu możemy je wykreślić na Mapie Kontekstu, podobnej do pokazanej na rysunku 4.8.



Rysunek 4.8. Mapa Kontekstu

Mapa Kontekstu jest wizualną reprezentacją Kontekstów Ograniczonych systemu i integracji między nimi. Ta wizualna notacja pozwala na uzyskanie cennych strategicznych informacji na wielu poziomach:

Wysokopoziomowy projekt

Mapa Kontekstu zawiera przegląd komponentów systemu i modeli, które one implementują.

Wzorce komunikacji

Mapa Kontekstu prezentuje wzorce komunikacji między zespołami — na przykład pokazuje, które zespoły ze sobą współpracują, a które preferują „mniej intymne” wzorce integracji, takie jak Warstwa Antykorupcyjna lub Różne Drogi.

Problemy organizacyjne

Mapa Kontekstu może przekazywać informacje organizacyjne. Na przykład, może odpowiadać na pytanie o to, co to znaczy, że wszyscy konsumenci usług zespołu z górnej części strumienia przetwarzania decydują się na implementację Warstwy Antykorupcyjnej lub jeśli wszystkie implementacje wzorca Różne Drogi koncentrują się wokół tego samego zespołu?

Utrzymanie

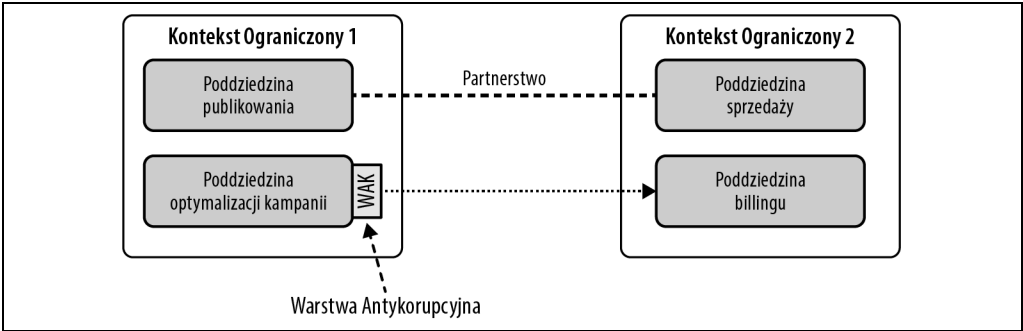
W idealnej sytuacji Mapa Kontekstu powinna zostać wprowadzona do projektu od samego jego początku i być aktualizowana w miarę dodawania nowych Kontekstów Ograniczonych i modyfikacji istniejących.

Ponieważ Mapa Kontekstu potencjalnie zawiera informacje pochodzące z pracy wielu Zespołów, najlepiej zdefiniować utrzymanie Mapy Kontekstu jako wspólny wysiłek: każdy Zespół jest odpowiedzialny za aktualizację własnych sposobów integracji z innymi Kontekstami Ograniczonymi.

Mapą Kontekstu można zarządzać i utrzymywać ją w postaci kodu, za pomocą takiego narzędzia jak Context Mapper (<https://contextmapper.org>).

Ograniczenia

Należy zapamiętać, że tworzenie wykresów Mapy Kontekstu może być trudne. Gdy Konteksty Ograniczone systemu obejmują wiele Poddziedzin, może istnieć wiele wzorców integracji. Na przykład na rysunku 4.9 widać dwa Konteksty Ograniczone z dwoma wzorcami integracji: Partnerstwo i Warstwa Antykorupcyjna.



Rysunek 4.9. Złożona Mapa Kontekstu

Co więcej, nawet jeśli Konteksty Ograniczone obejmują tylko jedną Poddziedzinę, nadal może istnieć wiele wzorców integracji — na przykład jeśli moduły Poddziedzin wymagają różnych strategii integracji.

Podsumowanie

Konteksty Ograniczone nie są niezależne. Muszą wchodzić ze sobą w interakcje. Różne sposoby integracji Kontekstów Ograniczonych definiują poniższe wzorce:

Partnerstwo

Konteksty Ograniczone są integrowane w sposób doraźny.

Wspólne Jądro

Dwa lub więcej Kontekstów Ograniczonych jest zintegrowanych przez współdzielenie ograniczonego, nakładającego się modelu, należącego do wszystkich korzystających z tego modelu Kontekstów Ograniczonych.

Konformista

Konsument dostosowuje się do modelu usługodawcy.

Warstwa Antykorupcyjna

Konsument tłumaczy model usługodawcy na model odpowiadający potrzebom konsumenta.

Usługa Otwartego Hosta

Usługodawca implementuje Język Opublikowany — model zoptymalizowany pod kątem potrzeb konsumentów.

Różne Drogi

Powielanie określonych funkcji jest tańsze niż współpraca i integracja.

Integracje pomiędzy Kontekstami Ograniczonymi można wykreślić na Mapie Kontekstu. Narzędzie to daje wysokopoziomowy wgląd w projekt systemu, istniejące wzorce komunikacji i problemy organizacyjne.

Po zapoznaniu się z narzędziami i technikami DDD do analizy i modelowania Dziedzin Biznesowych, zmienimy naszą perspektywę ze strategii na taktykę. W części II poznasz różne sposoby implementacji logiki Dziedziny, organizowania architektury wysokiego poziomu i koordynowania komunikacji między komponentami systemu.

Ćwiczenia

1. Który wzorzec integracji nigdy nie powinien być używany dla Poddziedzin Podstawowych?
 - a. Wspólne Jądro.
 - b. Usługa Otwartego Hosta.
 - c. Warstwa Antykorupcyjna.
 - d. Różne Drogi.

2. Do jakiej Poddziedziny z dolnej części strumienia przetwarzania lepiej pasuje implementacja Warstwy Antykorupcyjnej?
- a. Poddziedziny Podstawowej.
 - b. Poddziedziny Pomocniczej.
 - c. Poddziedziny Ogólnej.
 - d. B i C.
3. Do jakiej Poddziedziny z górnej części strumienia przetwarzania lepiej pasuje implementacja Usługi Otwartego Hosta?
- a. Poddziedziny Podstawowej.
 - b. Poddziedziny Pomocniczej.
 - c. Poddziedziny Ogólnej.
 - d. A i B.
4. Który wzorzec integracji w pewnym sensie narusza granice własności Kontekstów Ograniczonych?
- a. Partnerstwo.
 - b. Wspólne Jądro.
 - c. Różne Drogi.
 - d. Żaden wzorzec integracji nigdy nie powinien przekraczać granic własności Kontekstów Ograniczonych.

PROGRAM PARTNERSKI

— GRUPY HELION —



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA
Helion 

DDD: praktyczny sposób na rozwiązanie problemów biznesowych!

Domain-driven design, czyli projektowanie dziedziny, w skrócie DDD, jest zbiorem praktyk tworzenia architektury oprogramowania z uwzględnieniem dziedziny biznesu i jej problemów. W ten sposób logika biznesowa staje się sercem aplikacji. Wielu architektów jednak nie radzi sobie ze złożonością tworzonego oprogramowania. Nauczenie się zasad i wzorców DDD, przyswojenie technik dekompozycji dużego systemu, modelowania i projektowania integracji komponentów jest sposobem na zbudowanie systemu w pełni zgodnego z potrzebami i celami biznesowymi.

Ta książka zawiera opis podstawowych wzorców, zasad i praktyk przydatnych podczas analizy dziedzin biznesowych, ułatwiających zrozumienie ich strategii i dostosowanie architektury do potrzeb biznesu, aby umożliwić zbudowanie solidnej implementacji logiki biznesowej. Omówiono tu narzędzia i techniki podejmowania decyzji projektowych, a także istotniejsze wzorce projektowe. Dużo uwagi poświęcono kodowi i różnym sposobom implementacji logiki biznesowej systemu. Opisano również techniki i strategie stosowania DDD w rzeczywistych projektach. Ciekawym elementem jest zaprezentowanie związków projektowania dziedziny z innymi ważnymi metodologiami i wzorcami.

W książce między innymi:

- analiza dziedziny biznesowej firmy w kontekście architektury systemu
- strategiczne i taktyczne narzędzia DDD
- budowa wspólnego rozumienia dziedzin biznesowych
- dekompozycja systemu na konteksty ograniczone
- koordynacja pracy wielu zespołów
- stopniowe wdrażanie technik DDD do projektów typu brownfield

Vladik (Vlad) Khononov jest inżynierem oprogramowania od ponad 15 lat. Specjalizuje się w projektowaniu dziedziny dla firm z wielu branż i chętnie dzieli się swoim bogatym doświadczeniem. Często występuje podczas branżowych konferencji, takich jak O'Reilly Software Architecture, DDD Europe i NDC, a także działa na rzecz grup Domain-Driven Design Israel i Tel Aviv Software Architecture. Mieszka w północnym Izraelu z żoną i (prawie) rozsądną liczbą kotów.

Helion

helion.pl

HELION SA
ul. Kościuski 1c
44-100 Gliwice
tel. 32 250 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej!



ISBN 978-83-283-9262-5



Cena: 79,00 zł