TEAM TOPOLOGIES

ORGANIZOWANIE BIZNESU I ZESPOŁÓW TECHNOLOGICZNYCH DLA SZYBKIEGO PRZEPŁYWU PRACY

Przedmowa
RUTH MALAN

MATTHEW SKELTON
MANUEL PAIS

Helion

Tytuł oryginału: Team Topologies: Organizing Business and Technology Teams for Fast Flow

Tłumaczenie: Tomasz Walczak

ISBN: 978-83-289-3149-7

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
helion.pl/user/opinie/teamto
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *helion.pl* (księgarnia internetowa, katalog książek)

Printed in Poland.

SPIS TREŚCI

SPIS ILUSTRACJI I TABEL

ILUSTRACJE

TABELE

STUDIA PRZYPADKÓW I PRZYKŁADY Z BRANŻY

2. Prawo Conwaya i jego znaczenie

> [Prawo Conwaya] wymaga ciągłego zadawania sobie pytania: „Czy istnieje lepsze rozwiązanie projektowe, które jest dla nas niedostępne ze względu na strukturę naszej organizacji?”.
>
> — **Mel Conway**, *Toward Simplifying Application Development, in a Dozen Lessons*

W rozdziale 1. omówiliśmy, dlaczego organizacje powinny traktować strukturę zespołów jako ważny czynnik wpływający na sukces. Przedstawiliśmy także podstawowe idee i zasady, które pomagają zrozumieć, jak zespoły funkcjonują w ramach organizacji. Wprowadziliśmy kluczowe pojęcia, z których będziemy korzystać dalej w książce. W pozostałych rozdziałach części I szczegółowo wyjaśniamy, co prawo Conwaya mówi o zespołach, strukturze organizacyjnej i architekturze oprogramowania. Następnie omawiamy znaczenie podejścia stawiającego zespół na pierwszym miejscu. Celem części I jest przedstawienie podstawowych zasad projektowania organizacji i zespołów, które będą niezbędne do zrozumienia topologii zespołów. Zaczynamy od prawa Conwaya.

Jak zrozumieć i zastosować prawo Conwaya

Prawo Conwaya ma kluczowe znaczenie dla zrozumienia sił działających przy określonej strukturze zespołów, ponieważ mają one długotrwały, a przy tym nieuświadamiany wpływ na rozwijane systemy oprogramowania, dziś większe i z większą liczbą połączeń niż kiedykolwiek wcześniej. Można się jednak zastanawiać, czy prawo z 1968 roku dotyczące architektury oprogramowania nadal jest aktualne.

Przeszliśmy długą drogę: od mikrousług, przez chmurę i kontenery, po architektury bezserwerowe. Z naszego doświadczenia wynika, że takie nowinki mogą pomóc zespołom w lokalnym usprawnieniu pracy, ale im większa jest organizacja, tym trudniej w pełni

wykorzystać zalety takich podejść. Sposób, w jaki zespoły są zorganizowane i współpracują, często bazuje na wcześniejszych projektach i/lub starszych technologiach, a także odzwierciedla najnowszy projekt struktury organizacyjnej, który może mieć kilka, a nawet kilkadziesiąt lat.

Jeśli kiedykolwiek zdarzyło Ci się pracować dla dużej organizacji, prawdopodobnie znasz monolityczne, współdzielone bazy danych obsługujące całą firmę. Oczywiście istniały uzasadnione historyczne powody dominacji tego modelu, takie jak wzrost specjalizacji ludzi i zespołów w warstwach stosu technologicznego. Było tak aż do czasu, gdy podejście DevOps i mikrousługi zyskały na popularności. Czynniki takie jak orientacja na projekt, cięcie kosztów za pomocą outsourcingu czy brak doświadczenia zespołów przyczyniły się do utrwalenia tego (obecnie rozpoznawalnego) antywzorca. Monolityczne bazy danych łączą zależne od nich aplikacje i prowokują do wprowadzania drobnych zmian w logice biznesowej na poziomie bazy danych (więcej na ten temat piszemy w rozdziale 6.). Jednak aby uniknąć takich baz, organizacje potrzebują nie tylko dobrych praktyk architektonicznych, ale także odpowiednich struktur i składu zespołów, które są zgodne z tym nowym sposobem myślenia.

Firma odzieżowa Adidas przeszła interesującą transformację, w której bezpośrednio uwzględniono prawo Conwaya w trakcie projektowania organizacji. Jak wyjaśnili Fernando Cornago, starszy dyrektor ds. inżynierii platformy, i Markus Rautert, wiceprezes ds. inżynierii i architektury platformy, ich dział informatyczny przestał być postrzegany jako centrum kosztów, z jednym dostawcą dostarczającym większość oprogramowania (co wymagało częstego przekazywania projektów) i tylko kilkoma wewnętrznymi inżynierami (zajmującymi się bardziej zarządzaniem niż inżynierią). Zamiast tego zaczęto go traktować jak zespół zorientowany na produkt. Adidas zainwestował 80% zasobów inżynieryjnych w rozwój wewnętrznych kompetencji dostarczania oprogramowania przez interdyscyplinarne zespoły dostosowane do potrzeb biznesowych. Pozostałe 20% przeznaczono na centralny zespół odpowiedzialny za platformę, który zajmował się platformami inżynieryjnymi i zmianami technicznymi, a także doradztwem i wdrażaniem nowych specjalistów. Adidas był w stanie sześćdziesięciokrotnie zwiększyć częstotliwość udostępniania produktów cyfrowych, a jednocześnie wzrosła jakość rozwijanego oprogramowania[1].

Oprócz doświadczeń empirycznych istnieje również rosnąca liczba badań, które w większości potwierdzają twierdzenia Conwaya. Alan MacCormack i jego współpracownicy z Harvard Business School przeprowadzili badania nad różnymi otwartoźródłowymi i zamkniętymi produktami. Na tej podstawie znaleźli „mocne dowody potwierdzające hipotezę, że architektura produktu zwykle odzwierciedla struktury organizacji, w której jest rozwijana”[2].

Badania z innych branż, takich jak produkcja pojazdów i projektowanie silników lotniczych, również potwierdzają tę tezę[3]. Przeprowadzono wystarczająco dużo badań w przemyśle, aby wykazać, że siła homomorficzna opisana w prawie Conwaya ma szerokie zastosowanie.

Następujący cytat Ruth Malan można uznać za współczesną wersję prawa Conwaya: „Jeśli architektura systemu i architektura organizacji są ze sobą sprzeczne, przeważa architektura organizacji”[4]. Malan przypomina, że organizacja jest ograniczona do tworzenia projektów, które odzwierciedlają rzeczywistą, praktyczną strukturę komunikacji w organizacji. Ma to istotne implikacje strategiczne dla każdej organizacji projektującej i budującej systemy oprogramowania, czy to wewnętrznie, czy to z wykorzystaniem zewnętrznych dostawców.

Przede wszystkim organizacja podzielona na funkcjonalne silosy (gdzie zespoły specjalizują się w określonych funkcjach, takich jak kontrola jakości, administracja bazami danych lub bezpieczeństwo) prawdopodobnie nigdy nie stworzy systemów oprogramowania dobrze zaprojektowanych pod kątem kompletnego przepływu pracy. Podobnie organizacja zorganizowana głównie wokół kanałów sprzedaży dla różnych regionów geograficznych prawdopodobnie nie opracuje efektywnej architektury oprogramowania, która zapewni wiele różnych usług oprogramowania dla wszystkich regionów globalnych.

Dlaczego organizacje rzadko odkrywają lub utrzymują pewne architektury? Conway daje kilka wskazówek w artykule z 1968 roku: „Jeśli wziąć pod uwagę dowolną [określoną] strukturę zespołu, istnieje klasa projektów, których taka organizacja nie może skutecznie realizować, ponieważ brakuje w niej niezbędnych ścieżek komunikacji”[5].

Ścieżki komunikacji (zgodne z formalnymi liniami raportowania lub nie) w organizacji skutecznie ograniczają rodzaje rozwiązań, jakie dana organizacja może opracować. Można jednak strategicznie wykorzystać to zjawisko. Aby zniechęcić pracowników do rozwoju pewnych rodzajów projektów, być może tych nadmiernie skupionych na technicznych szczegółach, można odpowiednio przekształcić organizację. Podobnie jeśli chcesz, aby organizacja odkrywała i wprowadzała określone projekty, na przykład bardziej sprzyjające przepływowi, możesz tak ją przekształcić, aby to ułatwić. Oczywiście nie ma gwarancji, że organizacja znajdzie i zastosuje pożądane projekty, ale dzięki właściwemu ukształtowaniu ścieżek komunikacji przynajmniej zwiększasz prawdopodobieństwo uzyskania oczekiwanych efektów.

Projektowanie organizacji z uwzględnieniem prawa Conwaya staje się ważnym strategicznym procesem, który może znacznie przyspieszyć odkrywanie efektywnych projektów oprogramowania i pomóc uniknąć tych mniej przydatnych. W rozdziale 8. szczegółowo omawiamy, jak strategicznie rozwijać organizację zgodnie z prawem Conwaya.

Odwrócony manewr Conwaya

Aby zwiększyć szanse organizacji na zbudowanie efektywnych systemów informatycznych zoptymalizowanych pod kątem przepływu, można zastosować odwrócony manewr Conwaya w celu rekonfiguracji komunikacji między zespołami przed ukończeniem oprogramowania. Choć początkowo może to wywołać opór, przy wystarczającej determinacji ze strony kierownictwa i świadomości zespołów takie podejście może przynieść oczekiwane rezultaty.

Odwrócony manewr Conwaya zyskał popularność w świecie technologii około 2015 roku i od tego czasu został zastosowany w wielu organizacjach. Książka *Accelerate* (*Przyspieszenie*) Nicole Forsgren, Jeza Humble'a i Gene'a Kima potwierdza znaczenie tej strategii dla wysokowydajnych organizacji:

> Nasze badania potwierdzają skuteczność tzw. „odwróconego manewru Conwaya", zgodnie z którym organizacje powinny rozwijać strukturę zespołów i organizacji w taki sposób, aby osiągnąć pożądaną architekturę. Celem jest stworzenie architektury wspierającej zdolność zespołów do wykonywania pracy — od etapu projektowania po wdrożenie — bez konieczności intensywnej komunikacji między zespołami[6].

Pamiętasz antywzorzec monolitycznej bazy danych, o którym wspominaliśmy wcześniej? Widzieliśmy skrajne przypadki, gdzie z powodu braku stałych zespołów i realizacji wszystkich zmian w ramach tymczasowych projektów (głównie zlecanych na zewnątrz) aplikacje stawały się głęboko powiązane na poziomie bazy danych (w wyniku współdzielonych danych i procedur). W przeszłości utrudniało to wdrażanie gotowych systemów dla niektórych części firmy, ponieważ nie dało się ich używać niezależnie od reszty logiki biznesowej. Przez narastający dług techniczny wewnętrzni inżynierowie nie mają wtedy czasu na pracę nad najważniejszymi funkcjami, które zaspokajają zmieniające się potrzeby klientów, co ogranicza zdolność organizacji do szybszego działania i uzyskania przewagi nad konkurencją.

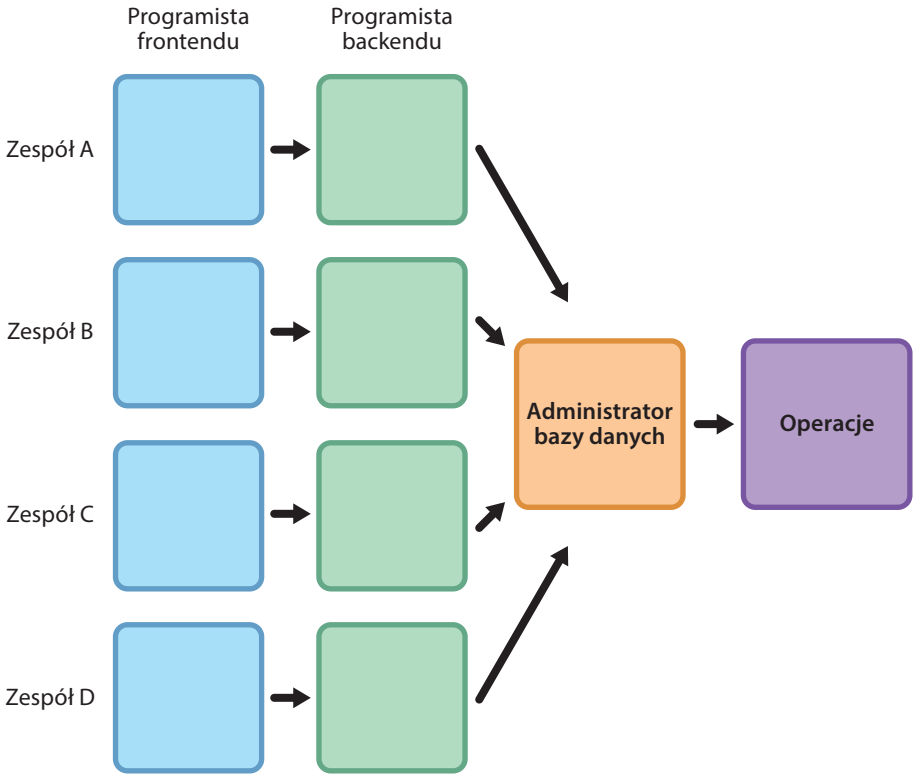
Jak zatem odwrócony manewr Conwaya może pomóc w strukturyzowaniu zespołów w celu uzyskania pożądanej architektury oprogramowania?

Przyjrzyj się celowo uproszczonej wersji prawa Conwaya dla organizacji tworzącej oprogramowanie. Pomoże to zilustrować istotne koncepcje i siły. Załóżmy, że nad różnymi częściami systemu pracują cztery niezależne zespoły, każdy składający się z programistów frontendu i backendu. Zespoły te przekazują zmiany administratorowi bazy danych. Rysunek 2.1 przedstawia przepływ zmian na poziomie koncepcyjnym.

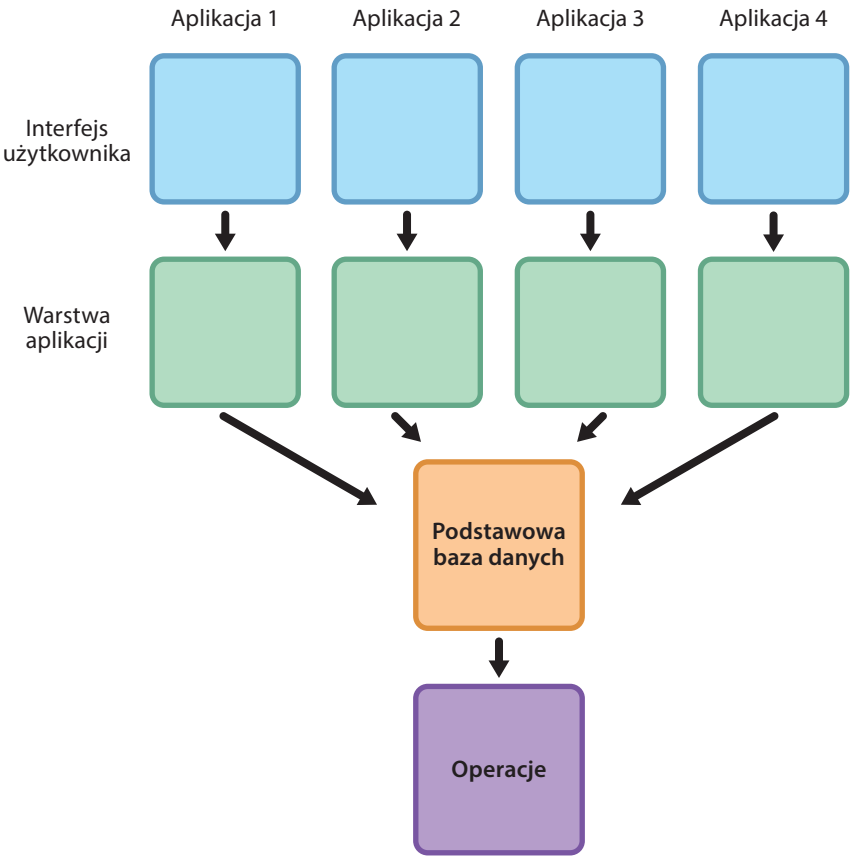
Zgodnie z prawem Conwaya architektura oprogramowania, która naturalnie wyłania się z takiego projektu zespołów, będzie miała oddzielne komponenty frontendowe i backendowe dla każdego zespołu oraz jedną wspólną bazę danych (rysunek 2.2).

Innymi słowy, jeden wspólny zespół administratorów baz danych prawdopodobnie doprowadzi do powstania jednej wspólnej bazy, a oddzielni programiści frontendu i backendu prawdopodobnie spowodują rozdział między warstwą interfejsu użytkownika a warstwą aplikacji. Wynika to z charakteru interakcji między pracownikami. Jeśli architektura oprogramowania z pojedynczą wspólną bazą danych i czterema dwuwarstwowymi aplikacjami jest pożądana, to wszystko jest w porządku.

Jednak jeśli *nie* chcesz używać jednej wspólnej bazy danych, pojawia się problem. Siła homomorficzna zidentyfikowana przez prawo Conwaya wywiera mocny wpływ na „naturalną" architekturę oprogramowania wynikającą z obecnej struktury organizacyjnej i ścieżek komunikacji.



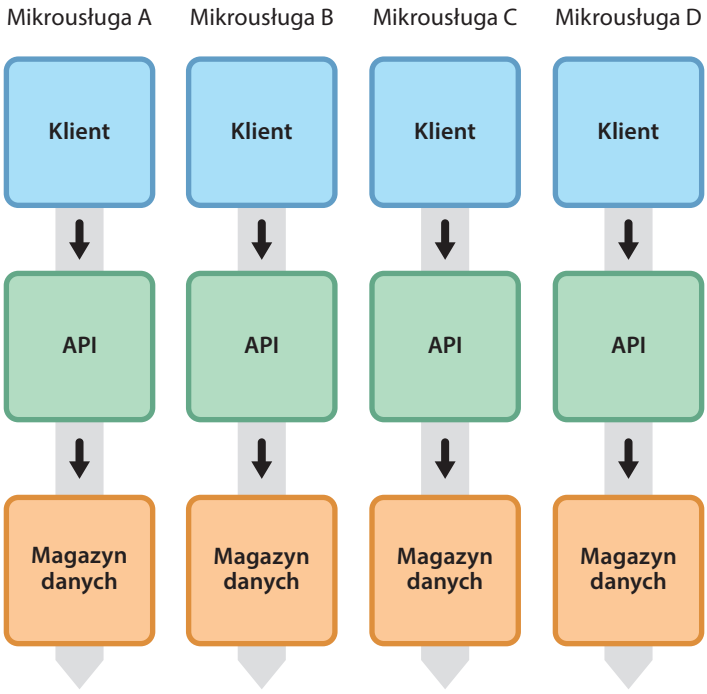
Rysunek 2.1. Cztery zespoły pracujące nad systemem informatycznym
W systemie informatycznym pracują cztery oddzielne zespoły składające się z programistów frontendu i backendu. Programiści frontendu komunikują się tylko z programistami frontendu, którzy z kolei kontaktują się z jednym administratorem bazy danych w sprawie zmian w bazie.



Rysunek 2.2. Architektura oprogramowania w organizacji z czterema zespołami

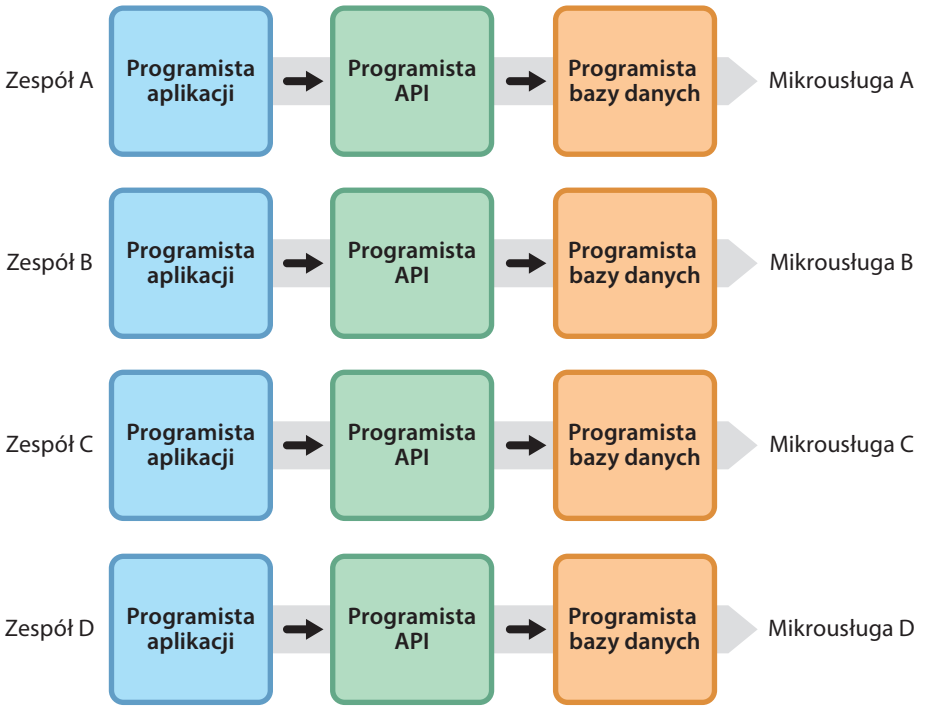
Cztery oddzielne aplikacje, każda z własnym interfejsem użytkownika i warstwą aplikacji, komunikujące się z jedną wspólną bazą danych. Odzwierciedla to architekturę komunikacji w zespołach, przedstawioną na rysunku 2.1 (rysunek został obrócony o dziewięćdziesiąt stopni).

Załóżmy na przykład, że dla nowych systemów opartych na chmurze chcesz zastosować architekturę mikrousług, w której każda usługa jest niezależna i posiada własne repozytorium danych (rysunek 2.3).



Rysunek 2.3. Architektura mikrousług z niezależnymi usługami i magazynami danych
Architektura oparta na mikrousługach z czterema odrębnymi usługami, z których każda ma własny magazyn danych, warstwę API i frontend.

Dzięki odwróconemu manewrowi Conwaya można zaprojektować zespoły w taki sposób, aby „pasowały" do wymaganej architektury oprogramowania. W tym celu należy utworzyć oddzielne zespoły dla aplikacji klienckich i API, a także włączyć do zespołu programistę baz danych, zamiast traktować go jako odrębną jednostkę (rysunek 2.4).



Rysunek 2.4. Projekt zespołów dostosowany do architektury mikrousług z niezależnymi usługami i magazynami danych
Projekt organizacji uwzględniający siłę homomorficzną związaną z prawem Conwaya. Ten projekt pomaga uzyskać architekturę oprogramowania z czterema niezależnymi mikrousługami (jest to schemat z rysunku 2.3 obrócony o dziewięćdziesiąt stopni).

Zgodnie z prawem Conwaya taka struktura zespołu w naturalny sposób doprowadzi do powstania pożądanej architektury oprogramowania. Jeśli chcesz, aby magazyny danych były zgodne z domeną biznesową, należy unikać tworzenia pojedynczego punktu dostępu w postaci jednej osoby lub zespołu odpowiedzialnego za bazę danych. Można to osiągnąć na przykład przez dodanie kompetencji w zakresie danych do zespołu zajmującego się rozwojem aplikacji.

Architektury oprogramowania sprzyjające przepływowi pracy w obrębie zespołu

Zgodnie z prawem Conwaya najpierw należy ustalić, jaka architektura oprogramowania jest potrzebna, a dopiero *potem* zająć się organizowaniem pracy zespołów. W przeciwnym razie ścieżki komunikacji i motywacje obecne w organizacji ostatecznie zdeterminują architekturę oprogramowania. Jak mówi Michael Nygard: „Przydział pracowników do zespołów jest pierwszym szkicem architektury”[7].

Aby zapewnić szybki i bezpieczny przepływ zmian, trzeba wziąć pod uwagę przepływ w obrębie zespołu i zaprojektować architekturę oprogramowania tak, aby do niego pasowała. Podstawowym środkiem dostarczania jest zespół (więcej na ten temat piszemy w rozdziale 3.), dlatego architektura systemu musi umożliwiać i ułatwiać szybki przepływ w obrębie każdego zespołu. Na szczęście w praktyce oznacza to, że można stosować sprawdzone dobre praktyki z dziedziny architektury oprogramowania:

- Luźne powiązanie — brak wysokiej zależności jednych komponentów od innych.
- Wysoka spójność — komponenty mają jasno określone zadania, a ich wewnętrzne elementy są ze sobą ściśle powiązane.
- Jasno określona i odpowiednia kompatybilność wersji.
- Jasno określone i odpowiednie testy z udziałem wielu zespołów.

Na poziomie koncepcyjnym architektura oprogramowania powinna odzwierciedlać możliwy dzięki niej przepływ zmian. Zamiast określać serie połączonych ze sobą komponentów, należy projektować przepływy na bazie używanej platformy (o platformach piszemy w rozdziale 5.).

Dzięki dostosowaniu zadań do wielkości zespołu łatwiej jest osiągnąć to, co MacCormack i współpracownicy nazywają „»architekturą uczestnictwa«, która ułatwia zrozumienie rozwiązań przez ograniczenie rozmiaru modułów oraz sprzyja wnoszeniu wkładu dzięki minimalizacji zakresu przekazywania zmian w projekcie”[8]. Innymi słowy, potrzebna jest architektura oprogramowania zorientowana na zespół, która maksymalnie ułatwia ludziom pracę.

Organizacja powinna stale kontrolować, czy elementy nie są ze sobą powiązane i czy pozostają dostosowane do wielkości zespołu, ponieważ, jak pisze John Roberts w *The Modern Firm* (*Nowoczesna firma. Strategie organizacyjne na rzecz wydajności i rozwoju*), „istotny wzrost wydajności często można osiągnąć dzięki zastosowaniu projektu zgodnego z modelem zdecentralizowanym”[9]. Ten wzrost wydajności wynika częściowo ze zwiększenia tempa przepływu zmian, a częściowo z możliwości dostosowywania architektury przez organizację do nowych kontekstów.

Don Reinertsen, autor *The Principles of Product Development Flow*, pisze tak: „można również wykorzystać architekturę jako czynnik umożliwiający szybkie wprowadzanie zmian. W tym celu należy zadbać o taką strukturę architektury, aby płynnie absorbowała

zmiany”[10]. Architektura staje się więc czynnikiem wspomagającym, a nie przeszkodą. Jednak dzieje się tak tylko wtedy, gdy przyjmiesz podejście zorientowane na zespół i uwzględnisz prawo Conwaya.

Projektowanie organizacji wymaga wiedzy technicznej

Jeśli przyjąć, że opisana przez Conwaya siła prowadząca do podobieństwa między architekturą a organizacją zespołu jest rzeczywista, trzeba również zaakceptować, że każdy, kto decyduje o kształcie i układzie zespołów inżynieryjnych, ma mocny wpływ na architekturę systemów oprogramowania. Jest to logiczny skutek wynikający z prawa Conwaya. Ruth Malan ujmuje to tak: „jeśli to menedżerowie decydują [...], które usługi będą budowane przez poszczególne zespoły, to w praktyce menedżerowie decydują o architekturze systemu”[11].

Jak dobrze dział kadr zna się na systemach oprogramowania? Czy grupa liderów działów decydująca o przydziale budżetu dla zespołów zdaje sobie sprawę z możliwego wpływu dokonanych wyborów na przydatność architektury oprogramowania?

Jeśli wziąć pod uwagę coraz liczniejsze dowody na homomorfizm związany z prawem Conwaya, bardzo nieefektywne (być może nawet nieodpowiedzialne) jest określanie w organizacjach tworzących systemy oprogramowania kształtu, zakresu odpowiedzialności i granic zespołów bez udziału liderów technicznych.

Projektowanie organizacji i projektowanie oprogramowania są w praktyce dwiema stronami tej samej monety. Dlatego oba te zadania powinny być wykonywane przez tę samą wysoce kompetentną grupę osób. Rozwinięciem tej idei są słowa Allana Kelly’ego dotyczące roli architekta oprogramowania:

> „Jestem coraz bardziej przekonany, że osoba twierdząca, że jest Architektem, potrzebuje zarówno umiejętności technicznych, jak i społecznych. Musi rozumieć ludzi i umieć pracować w ramach struktury społecznej. Potrzebuje też zakresu obowiązków, który nie ogranicza się wyłącznie do kwestii technologicznych. Musi mieć również wpływ na struktury organizacyjne i kwestie personalne, czyli powinna być także menedżerem”[12].

Należy więc angażować w projektowanie organizacji osoby o umiejętnościach technicznych, ponieważ rozumieją one kluczowe koncepcje związane z projektowaniem oprogramowania, takie jak API, interfejsy, abstrakcja, hermetyzacja i tym podobne. Naomi Stanford ujmuje to tak: „działy i oddziały, systemy i procesy biznesowe [...] można projektować niezależnie, o ile w projekcie uwzględnione są interfejsy i granice z innymi jednostkami organizacji”[13].

Ogranicz zbędną komunikację

Jedną z ważnych implikacji prawa Conwaya jest to, że nie wszystkie przejawy komunikacji i współpracy są korzystne. Dlatego ważne jest zdefiniowanie „interfejsów zespołów", aby określić oczekiwania dotyczące tego, jaki rodzaj pracy wymaga ścisłej współpracy, a jaki nie. Wiele organizacji zakłada, że bardziej intensywna komunikacja zawsze jest korzystna. Jednak nie jest to do końca prawdą.

Potrzebna jest *ukierunkowana* komunikacja między konkretnymi zespołami. Należy zwracać uwagę na nieoczekiwaną komunikację i analizować jej przyczyny. Jak wykazali Manuel Sosa i jego współpracownicy w badaniach nad produkcją samolotów z 2004 roku, „menedżerowie powinni się skupić na zrozumieniu przyczyn powstawania nieuwzględnionych interfejsów projektowych [...] i nieprzewidzianych interakcji między zespołami [...] w systemach modułowych"[14].

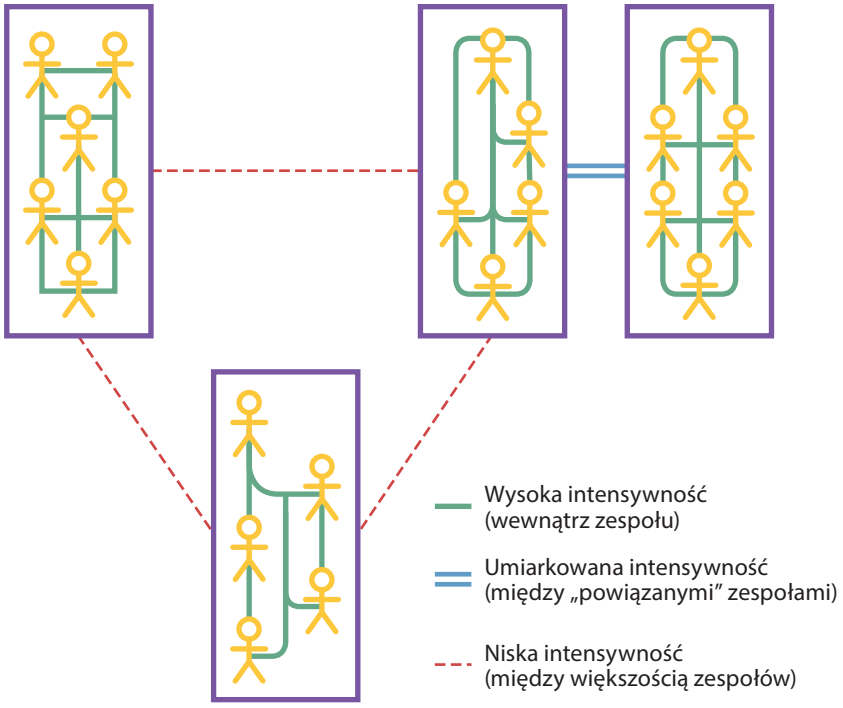
Mike Cohn, jeden z twórców podejścia Scrum w rozwoju produktów, w celu oceny jakości komunikacji między zespołami w organizacji zadaje następujące pytania: „Czy struktura minimalizuje liczbę ścieżek komunikacji między zespołami? [...] Czy struktura zachęca do komunikacji zespoły, które w przeciwnym razie by tego nie robiły?"[15].

Cohn zwraca tu uwagę na to, że jeśli zgodnie z projektem architektury oprogramowania dwa zespoły nie potrzebują się ze sobą komunikować, to musi występować jakiś problem, jeśli taka komunikacja *ma miejsce*. Czy API nie jest wystarczająco dobre? Czy platforma nie jest odpowiednia? Czy brakuje jakiegoś komponentu? Jeśli można ograniczyć komunikację między zespołami (lub nawet całkowicie z niej zrezygnować), a jednocześnie nadal budować i udostępniać oprogramowanie w bezpieczny, efektywny i szybki sposób, to należy tak postąpić. Ilustruje to rysunek 2.5, inspirowany prezentacją „Real Life Agile Scaling" Henrika Kniberga[16].

Prostym sposobem na ograniczenie komunikacji jest przeniesienie dwóch zespołów do różnych części biura, na różne piętra, a nawet do różnych budynków. Jeśli zespoły są wirtualne lub rozmawiają głównie za pomocą komunikatorów, natężenie i wzorce komunikacji między zespołami mogą pomóc zidentyfikować interakcje niezgodne z oczekiwaniami wynikającymi z danej architektury oprogramowania.

Podobnie jeśli duży zespół regularnie zajmuje się dwoma odrębnymi obszarami systemu, korzystne może być podzielenie go na dwie mniejsze grupy odpowiedzialne za te obszary. Należy to jednak zrobić tylko wtedy, gdy konkretni członkowie zespołu pracują nad różnymi systemami. Z kolei jeżeli zgodnie z planem cały zespół ma pracować nad więcej niż jedną częścią systemu (na przykład nad nowszą usługą i starszym komponentem), lepiej zachować go w całości. Więcej o wzorcach długoterminowej „ciągłej opieki" nad starszymi systemami dowiesz się z rozdziału 9.

Czasami dwa zespoły (lub większa ich liczba) mogą chcieć komunikować się w kwestii oprogramowania wyłącznie dlatego, że kod ich części systemu znajduje się w tym samym repozytorium w systemie kontroli wersji lub nawet jest częścią tej samej aplikacji bądź



Rysunek 2.5. Komunikacja między zespołami
Komunikacja wewnątrz zespołów powinna być bardzo intensywna. Komunikacja między dwoma „powiązanymi" zespołami może mieć umiarkowane natężenie. Natomiast komunikacja między większością zespołów powinna być ograniczona.

usługi, choć logicznie komponenty powinny być odrębne. W takich sytuacjach należy zastosować wzorce „płaszczyzn podziału" (omówione w rozdziale 6.), aby rozbić oprogramowanie na mniejsze fragmenty, które można przechowywać w osobnych repozytoriach.

Nie każdy musi się komunikować ze wszystkimi

W biurach typu open space, a szczególnie przy powszechnej, natychmiastowej komunikacji za pomocą komunikatorów, każdy może kontaktować się z każdym. W takiej sytuacji łatwo rozwija się schemat interakcji, gdzie wszyscy *muszą* komunikować się ze wszystkimi (co przerzuca ciężar filtrowania istotnych informacji na odbiorcę), aby wykonać swoją pracę. W kontekście prawa Conwaya może to prowadzić do niezamierzonych konsekwencji w systemach informatycznych, a zwłaszcza do braku modularności na poziomie podsystemów.

Jeśli w organizacji panuje przekonanie, że „każdy powinien czytać wszystkie wiadomości na czacie", „wszyscy muszą uczestniczyć w dużych spotkaniach stand-up" lub „wszyscy muszą być obecni na zebraniach", aby zatwierdzać decyzje, oznacza to problem

w strukturze organizacyjnej. Z prawa Conwaya wynika, że tego rodzaju komunikacja „wielu z wieloma" będzie prowadzić do powstawania monolitycznych, splątanych, silnie powiązanych i współzależnych systemów, które nie sprzyjają szybkiemu przepływowi pracy. Bardziej intensywna komunikacja niekoniecznie jest korzystna.

Uwaga: pułapki bezkrytycznego stosowania prawa Conwaya

Istnieje ryzyko błędnej interpretacji prawa Conwaya i tworzenia zespołów, które pozornie są zgodne z wymaganą architekturą, ale w rzeczywistości utrudniają szybki przepływ pracy. Co więcej, często pomija się lub ignoruje związek między narzędziami używanymi przez różne zespoły a komunikacją, choć takie narzędzia mogą być istotnym czynnikiem wpływającym na spójność projektu. W tym podrozdziale omawiamy pułapki wynikające z naiwnego stosowania prawa Conwaya.

Wybór narzędzi kształtuje sposoby komunikacji

Sposób, w jaki zespoły korzystają z narzędzi komunikacyjnych, może mieć duży wpływ na wzorce kontaktów między nimi. Częstym problemem w organizacjach, które starają się budować i utrzymywać nowoczesne systemy informatyczne, jest niedopasowanie granic odpowiedzialności zespołów lub działów do granic narzędzi. Czasami organizacja korzysta z wielu narzędzi, gdy może wystarczyć jedno (zapewniające wspólny, ujednolicony obraz). Z kolei w innych scenariuszach używane jest jedno narzędzie, co prowadzi do problemów, ponieważ zespoły potrzebują oddzielnych rozwiązań.

Wyjaśniliśmy, że zgodnie z prawem Conwaya organizacja jest ograniczona do tworzenia projektów, które odzwierciedlają jej struktury komunikacyjne. Dlatego trzeba mieć świadomość wpływu wspólnych narzędzi na sposób interakcji zespołów. Jeśli chcesz, aby zespoły współpracowały ze sobą, używanie wspólnych narzędzi ma sens. Jeżeli jednak potrzebujesz wyraźnej granicy odpowiedzialności między zespołami, lepsze mogą być oddzielne narzędzia (lub różne instancje tego samego rozwiązania).

Załóżmy, że chcesz, aby zespół programistów ściśle współpracował z informatycznym zespołem operacyjnym. Używanie oddzielnych narzędzi do zarządzania zgłoszeniami lub incydentami w tych dwóch zespołach prawdopodobnie ograniczy komunikację między nimi. Aby pomóc tym zespołom we współpracy i komunikacji, należy wybrać narzędzie, które spełnia potrzeby obu grup. Podobnie warto unikać specjalnego narzędzia „tylko dla środowiska produkcyjnego", z którego mogą korzystać wyłącznie zespoły z dostępem do tego środowiska. Jeśli takie narzędzie wchodzi w interakcję z oprogramowaniem lub przeprowadza jego pomiary, ograniczony dostęp do niego prawdopodobnie spowoduje lukę komunikacyjną między zespołami z dostępem i bez dostępu do tego produktu. Narzędzie może pozytywnie lub negatywnie wpływać na przepływ komunikacji, a tym samym na skuteczność interakcji między zespołami.

WSKAZÓWKA

Zapewnij widoczność informacji, a jednocześnie pamiętaj o bezpieczeństwie. Narzędzia do agregacji logów zapewniają proste rozwiązanie dla zespołów deweloperskich, które potrzebują dzienników z systemu produkcyjnego (na przykład w celu debugowania), ale nie mają bezpośredniego dostępu do środowisk produkcyjnych. Takie narzędzia przesyłają wszystkie dzienniki do zewnętrznej lokalizacji, gdzie są one wspólnie przetwarzane i indeksowane (a w razie potrzeby anonimizowane), co umożliwia szybsze przeszukiwanie zdarzeń i szukanie korelacji niż w wyniku analizy pojedynczych logów. Zespoły uzyskują dostęp do potrzebnych informacji, a jednocześnie zachowane zostają zabezpieczenia środowiska produkcyjnego (przy czym trzeba zadbać o bezpieczne przesyłanie dzienników).

Jednakże gdy zakresy odpowiedzialności dwóch zespołów *nie* nakładają się na siebie (zespoły mają odrębne role, które nie wymagają intensywnej współpracy), narzucanie obu grupom tego samego narzędzia do śledzenia incydentów lub nawet tego samego produktu do monitorowania nie przyniesie wiele korzyści, zwłaszcza jeśli jeden z zespołów jest zewnętrzny i dostarcza usługi.

Nie wybieraj więc jednego narzędzia dla całej organizacji bez wcześniejszego przeanalizowania relacji między zespołami. Używaj oddzielnych narzędzi dla niezależnych zespołów, a wspólnych dla jednostek, które ze sobą współpracują.

Wiele zespołów odpowiedzialnych za różne komponenty

Niektóre organizacje naiwnie interpretują prawo Conwaya i tworzą wiele zespołów odpowiedzialnych za różne komponenty, skupiających się na budowaniu małych części systemów. Zespoły odpowiedzialne za komponenty — choć może lepiej nazwać je zespołami ds. skomplikowanych podsystemów, co omawiamy w rozdziale 5. — są czasami potrzebne, ale dzieje się tak tylko w wyjątkowych przypadkach, gdy niezbędna jest szczegółowa wiedza specjalistyczna. Zwykle należy optymalizować szybki przepływ pracy, dlatego preferowane są zespoły zorientowane na strumień wartości. Te aspekty omawiamy szerzej w rozdziale 5.

Ciągłe reorganizacje tworzące wewnętrzne „królestwa” lub prowadzące do redukcji zatrudnienia

W przeszłości ukrytym celem wielu „reorganizacji” było redukowanie liczby pracowników lub tworzenie obszarów wpływów dla menedżerów i liderów. Kiedy firma zmienia strukturę, aby dostosować ją do prawa Conwaya, powinna dążyć do ulepszania przestrzeni (kontekstu, ograniczeń itp.), w której można poszukiwać rozwiązań dotyczących systemów informatycznych. Te dwa podejścia wzajemnie się wykluczają. W firmach tworzących oprogramowanie i produkty cyfrowe struktura powinna wyprzedzać architekturę produktu.

Jeśli połączyć to nastawienie z podejściem stawiającym zespół na pierwszym miejscu, należy zrezygnować z regularnych reorganizacji dla celów odpowiadających kierownictwu.

Można ująć to bezpośrednio: regularne reorganizacje przeprowadzane dla wygody kierownictwa lub redukcji zatrudnienia utrudniają organizacji efektywne budowanie i obsługę systemów informatycznych. Reorganizacje ignorujące zagrożenia wynikające z prawa Conwaya, obciążenia poznawczego zespołu i powiązanej dynamiki są jak operacja na otwartym sercu wykonywana przez dziecko i mogą wyrządzić poważne szkody.

Podsumowanie: prawo Conwaya jest kluczowe dla efektywnego projektowania zespołów w branży informatycznej

Zgodnie z prawem Conwaya struktura organizacji i rzeczywiste ścieżki komunikacji między zespołami znajdują odzwierciedlenie w architekturze tworzonych systemów. Dlatego bezcelowe są próby projektowania oprogramowania niezależnie od projektowania samych zespołów.

To proste prawo ma dalekosiężne konsekwencje. Po pierwsze, struktura organizacji ogranicza liczbę możliwych architektur systemu. Po drugie, szybkość dostarczania oprogramowania jest silnie uzależniona od poziomu zależności między zespołami, narzucanych przez strukturę organizacji.

Szybki przepływ pracy wymaga ograniczenia komunikacji między zespołami. Współpraca między zespołami jest ważna na etapie rozwoju, gdzie niezbędne są odkrywanie możliwości i wiedza ekspercka. Jednak w obszarach, gdzie ważniejsze jest wykonywanie określonych zadań, a nie odkrywanie możliwości, komunikacja staje się zbędnym kosztem.

Ważną metodą uzyskiwania pożądanej architektury oprogramowania (i związanych z nią korzyści, takich jak szybkość dostarczania czy czas potrzebny na eliminowanie awarii) jest zastosowanie odwróconego manewru Conwaya, czyli projektowanie zespołów zgodnych z oczekiwaną architekturą. Przedstawiliśmy prosty przykład, w którym organizacja mogła uniknąć monolitycznej bazy danych przez włączenie kompetencji z zakresu baz danych do zespołu odpowiedzialnego za aplikację. Dzięki temu zespół zyskał wystarczającą autonomię, by utrzymywać oddzielne magazyny danych (przy czym niewykluczone, że polegał na centralnym zespole administratorów baz danych w kwestii zaleceń dotyczących projektowania baz lub synchronizacji z innymi bazami).

Podsumowując, dzięki uwzględnieniu wpływu prawa Conwaya w trakcie projektowania architektury oprogramowania i/lub reorganizacji struktury zespołów można wykorzystać siłę izomorficzną, która upodabnia architekturę oprogramowania do struktury zespołów.

W idealnym świecie zespoły są trwałe, samodzielne i zawsze w pełni zaangażowane. W rzeczywistości jednak muszą współpracować — a ich interakcje powinny się zmieniać wraz z cyklem życia produktu i technologii. Dlatego organizacje potrzebują modeli, które pomagają zespołom stale analizować własne działanie i ewoluować, aby sprostać wymaganiom dynamicznego środowiska biznesowego.

Oto praktyczny przewodnik, dzięki któremu ułatwisz swojemu zespołowi skuteczne dostarczanie wartości!

— **Ruth Malan**, Bredemeyer Consulting

Dzięki tej książce odkryjesz Team Topologies — praktyczny i elastyczny model projektowania organizacji. Zapewnia on jasne wzorce, proste do zastosowania i interpretacji w wielu różnych zespołach i kontekstach. Dowiesz się, z czym są związane ograniczenia pracy zespołowej, jak brzmi prawo Conwaya i w jaki sposób je zastosować. Opisano tu zasady wyboru topologii zespołów dla różnych kontekstów organizacyjnych i przypisywania zespołów do poszczególnych obszarów systemu. Przedstawiono także sposoby rozwijania struktury organizacyjnej, aby znacznie zwiększyć innowacyjność i przyśpieszyć dostarczanie rozwiązań. W efekcie zbudujesz organizację, która będzie skutecznie reagować na zmieniające się uwarunkowania.

Skoncentruj się na najważniejszych strategiach zespołów o wysokiej wydajności!

— **Barry O'Reilly**, założyciel ExecCamp i autor książek

Matthew Skelton i Manuel Pais są ekspertami w dziedzinie organizacji pracy zespołów IT. Skelton od lat doradza firmom z całego świata w zakresie DevOps, architektury systemów i skalowania zespołów. Pais specjalizuje się w usprawnianiu współpracy między zespołami technologicznymi i biznesowymi, koncentruje się na poprawie przepływu pracy. Razem stworzyli metodologię Team Topologies, która stała się globalnym punktem odniesienia dla skutecznych modeli organizacyjnych.

Helion
helion.pl
HELION S.A.
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej!

ISBN 978-83-289-3149-7

Cena: 79,00 zł

IT REVOLUTION