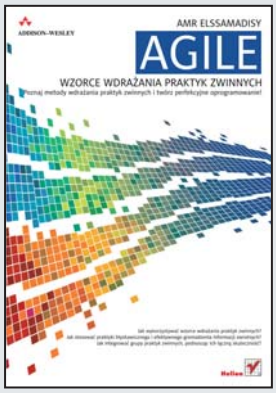
Agile. Wzorce wdrażania praktyk zwinnych



Autor: Amr Elssamadisy
Tłumaczenie: Mikołaj Szczepaniak
ISBN: 978-83-246-2318-1
Tytuł oryginału: Agile Adoption Patterns: A Roadmap to Organizational Success
Format: 168x237, stron: 408

Poznaj metody wdrażania praktyk zwinnych i twórz perfekcyjne oprogramowanie!

- Jak wykorzystywać wzorce wdrażania praktyk zwinnych?
- Jak stosować praktyki błyskawicznego i efektywnego gromadzenia informacji zwrotnych?
- Jak integrować grupy praktyk zwinnych, podnosząc ich łączną skuteczność?

Metody zwinne mają pomóc Ci w tworzeniu oprogramowania dostarczającego więcej walorów biznesowych – dzięki nim powinieneś robić to nie tylko szybciej i taniej, ale też bezpiecznie i bezstresowo. Okazuje się jednak, że wiele organizacji ma problemy z implementowaniem i pełnym wykorzystaniem tych metod. Jeśli nie chcesz dołączyć do tego grona, powinieneś skorzystać z tej książki – zaprezentowano w niej najlepsze praktyki doskonalące proces wytwarzania oprogramowania, a poza tym wskazano konkretne powody wyboru zalecanych praktyk.

Książka „Agile Adoption Patterns A Roadmap to Organizational Success" w sposób wyczerpujący, a jednocześnie zrozumiały prezentuje proces definiowania optymalnej strategii wdrażania praktyk zwinnych. W podręczniku zanalizowane zostały także najważniejsze przeszkody na drodze do implementacji metod zwinnych, obok których zaprezentowano sprawdzone rozwiązania tych problemów. Z tego przewodnika dowiesz się, jak wybrać praktyki najlepsze dla Twojej firmy i Twojego środowiska technicznego oraz jak przyrostowo wdrażać metody zwinne. Nauczysz się efektywnego tworzenia oprogramowania niezależnie od Twojej roli w projekcie – lidera, programisty, architekta lub klienta.

- Wzorce wdrażania praktyk zwinnych
- Praktyki sprzężenia zwrotnego
- Praktyki techniczne i pomocnicze
- Zautomatyzowane testy programisty
- Programowanie w parach
- Angażowanie społeczności
- Projekty ewolucyjne
- Wdrażanie praktyk zwinnych
- Wymagania sterowane testami
- Iteracja zwinna
- Grupa praktyk komunikacyjnych

Oto podręcznik efektywnego wdrażania praktyk zwinnych, które bez trudu zaimplementujesz do swojego projektu!

Helion SA
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl

SPIS TREŚCI

Rozdział 5

WDRAŻANIE PRAKTYK ZWINNYCH

Do tej pory koncentrowaliśmy się na walorach i problemach biznesowych. Na końcu obu dotychczasowych rozdziałów tej części wykonałeś ćwiczenia, w ramach których opracowałeś listę walorów biznesowych (z przypisanymi priorytetami) oraz listę problemów wymagających rozwiązania (także z priorytetami). Jeśli nie sporządziłeś tych list, wróć do odpowiednich podrozdziałów i wykonaj te ćwiczenia teraz. Bogatszy o znajomość priorytetów swoich klientów i najważniejszych utrudnień napotykanych przez Twoją firmę, będziesz gotowy do identyfikacji praktyk, których wdrożenie powinieneś rozważyć właśnie z myślą o eliminacji tych przeszkód i uzyskiwaniu możliwie wysokiej wartości w wyniku swoich działań.

W tym rozdziale spróbuję wskazać Ci kierunek, jak właściwie wybierać praktyki pod kątem ewentualnego wdrażania we własnej organizacji. Postaram się też zasugerować Ci sposoby oceny Twojej dotychczasowej pracy, która — mimo że z natury rzeczy będzie subiektywna — pozwoli Ci osiągnąć należytą „zwinność" w podejściu do problemu wdrażania. Chciałbym, żebyś traktował ten materiał jako zbiór wskazówek, jak wyznaczać własne priorytety i jak sporządzać własną listę praktyk do wdrożenia. Jeśli szukasz gotowych recept — wymienionych praktyk A, B i C — z pewnością nie znajdziesz ich w tej książce. (A jeśli znajdziesz je gdzie indziej, zachowaj ostrożność, ponieważ tak reklamowanym materiałom z reguły nie można ufać).

PRAKTYKI

Istotą tej książki jest prezentacja wzorców wdrażania zwinnych praktyk wytwarzania oprogramowania, czyli praktyk zwinnych sporządzonych w formie wzorców koncentrujących się na problemie wdrażania. W tym rozdziale spróbujemy wspólnie wybrać praktyki najlepiej pasujące do kontekstu Twojej organizacji. Identyfikacja właściwych praktyk w dużej mierze zależy od pracy wykonanej w poprzednich rozdziałach, kiedy zachęcałem Cię do przypisania swoich walorom biznesowym i problemom priorytetów — na tej podstawie wybierzemy praktyki, których wdrożenie powinno podnieść walory biznesowe i ograniczyć obserwowane problemy.

WZORCE KOJARZENIA PRAKTYK ZWINNYCH Z WALORAMI BIZNESOWYMI

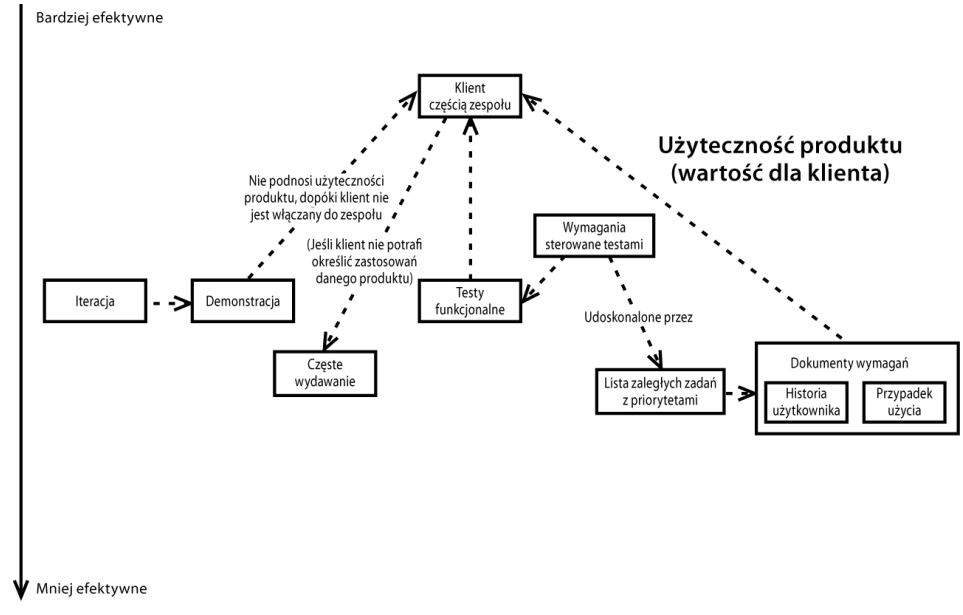
Zacznijmy od istoty tego rozdziału. Na rysunkach od 5.1 do 5.7 przedstawiono diagramy ilustrujące związki poszczególnych walorów biznesowych z praktykami, które powinny te walory rozszerzać. Prezentowane związki (odwzorowania), jak wszystkie wzorce opisywane w tej książce, powstały przez połączenie doświadczeń zgromadzonych podczas wielu prób wdrażania praktyk zwinnych. Każda z tych praktyk odpowiada wzorcowi, który zostanie szczegółowo udokumentowany w dalszej części tej książki. Nie powinieneś się więc przejmować, jeśli na tym etapie część spośród nich jest dla Ciebie niezrozumiała.

Przeanalizujmy teraz diagram z rysunku 5.1, abyś lepiej zrozumiał, jak czytać podobne ilustracje walorów biznesowych. Strzałki pomiędzy praktykami reprezentują zależności — oznacza to, że na przykład refaktoryzacja zależy od zautomatyzowanych testów programisty. Istotne znaczenie ma także układ pionowy — im wyżej znajduje się dana praktyka, tym większy jest jej wpływ na odpowiedni walor biznesowy. Oznacza to, że na przykład iteracje mają większy wpływ na walor krótkiego czasu dostarczania produktu na rynek (pozwala ten czas bardziej skrócić) niż zautomatyzowane testy programisty, a praktyka wytwarzania poprzedzonego testami jest bardziej efektywna od praktyki wytwarzania kończonego testami. Możesz więc wykorzystywać te diagramy do określania, które praktyki zasługują na szczególną uwagę jako potencjalne rozwiązania do wdrożenia. Nie należy jednak traktować tych diagramów jako czegoś więcej niż sugestie. Mimo że wszystkie proponowane praktyki mają pozytywny wpływ na odpowiednie walory biznesowe, bliższa analiza szczegółowych rozwiązań może wykazać, że niektóre z nich nie znajdują zastosowania w przypadku Twojej organizacji.

Rysunek 5.1. Praktyki skracające czas dostarczania produktu na rynek

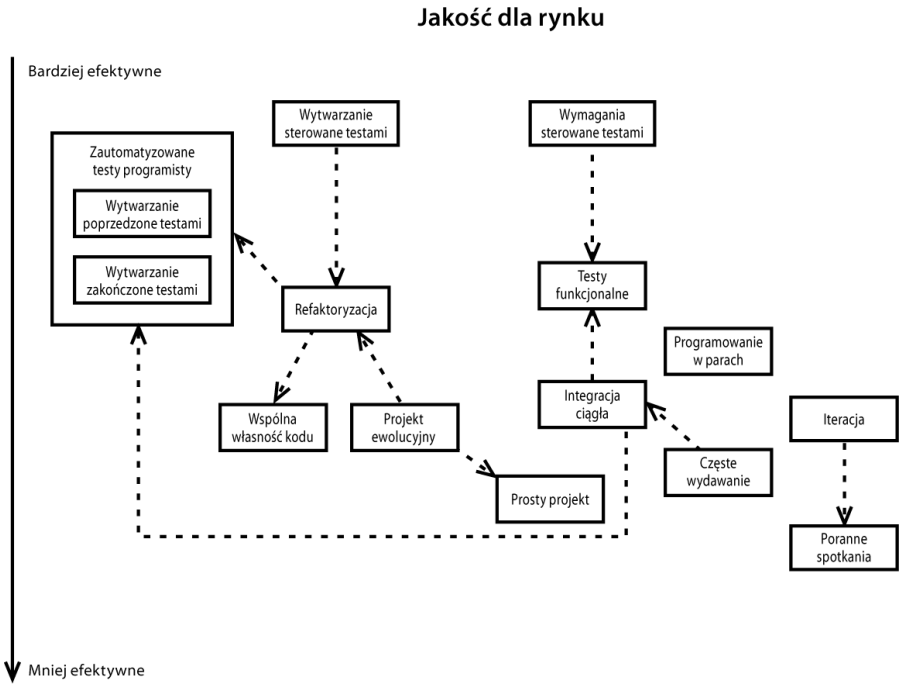
Wykonywanie niewielkich kroków i — tym samym — błyskawiczne odkrywanie niepowodzeń jest najbardziej efektywną metodą oceny skuteczności poszczególnych praktyk. Warto eliminować usterki możliwie wcześnie, ponieważ im wcześniej je odkrywamy, tym mniej nas kosztują i tym mniejsze jest ryzyko rozpoczęcia budowy na niepewnych fundamentach. Właśnie dlatego takie praktyki, jak iteracja czy integracja ciągła (ang. *continuous integration*), mają tak dobry wpływ na czas wprowadzania produktu na rynek. Obie te praktyki zależą jednak od innych praktyk. Wyobraźmy sobie, że chcemy skrócić czas przekazywania produktu na rynek, wdrażając praktykę zautomatyzowanych testów i trio iteracyjnego, czyli iteracji, stanu wykonania (ang. *done state*) oraz listy zaległych zadań iteracji (ang. *iteration backlog*).

Na rysunku 5.2 pokazano praktyki zwiększające użyteczność produktu. Jak dotąd najbardziej efektywną spośród znanych praktyk w tym obszarze jest metoda włączania klientów do zespołu (ang. *customers part of team*). Warto więc sprawdzić jej skuteczność w pierwszej kolejności. Jeśli już teraz stosujesz zautomatyzowane testy programistów lub metodę kończenia iteracji prezentacjami wersji demonstracyjnych, powinieneś rozważyć wdrożenie praktyki testów funkcjonalnych.



Rysunek 5.2. Praktyki podnoszące użyteczność produktu

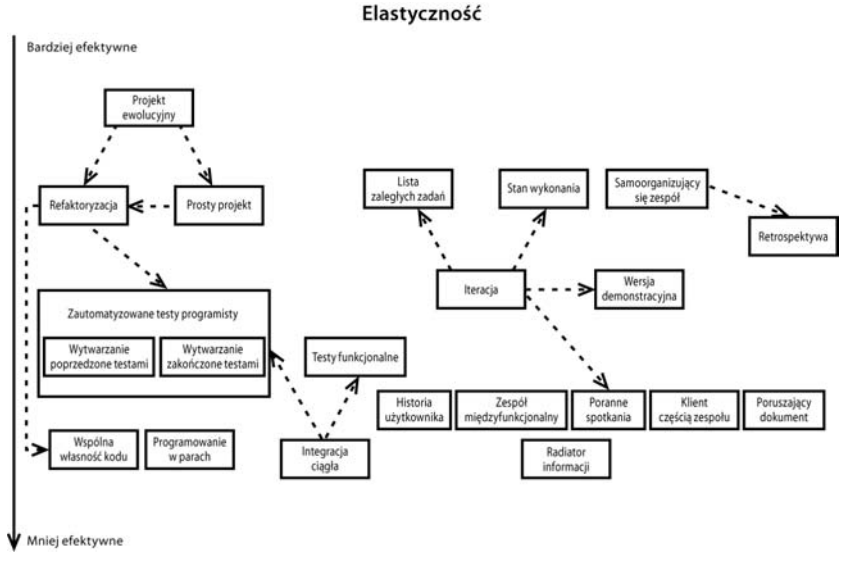
Mimo że w obszarze jakości produktów niepodzielnie rządzą praktyki wytwarzania sterowanego testami i wymagań sterowanych testami, obie te metody zależą od innych praktyk (patrz rysunek 5.3). Powinieneś więc rozważyć przyjęcie za punkt wyjścia jednej z praktyk zaliczanych do grupy zautomatyzowanych testów programisty (najlepiej wytwarzanie poprzedzone testami)



Rysunek 5.3. Praktyki podnoszące jakość produktu

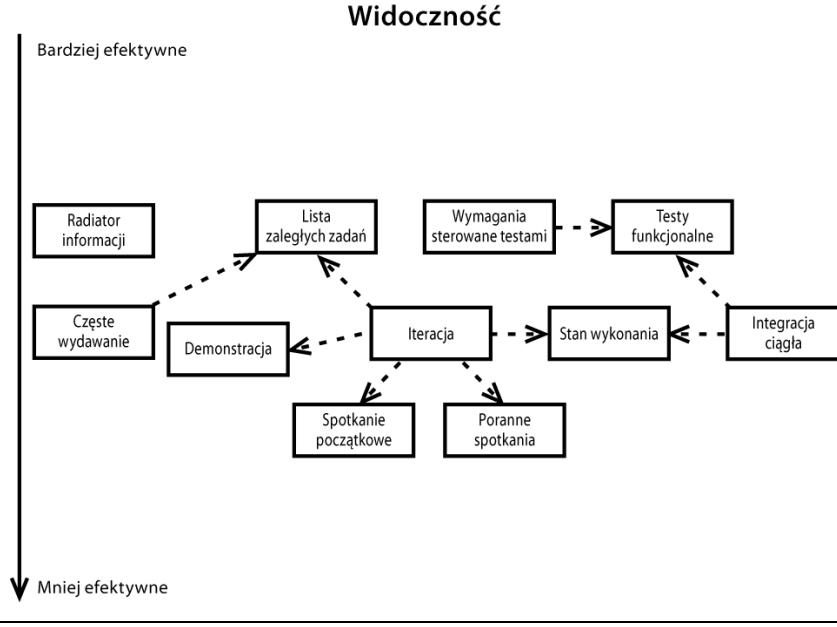
oraz praktyki programowania w parach. Bezpośrednio po nich warto wprowadzić technikę refaktoryzacji. Programowanie w parach powinno Ci pomóc w szybkim i bezbolesnym wdrażaniu tych wyjątkowo trudnych praktyk. Kiedy już nabierzesz wprawy w korzystaniu ze zautomatyzowanych testów programisty, skoncentruj swoją uwagę na pełnowartościowym wytwarzaniu sterowanym testami i rozważ implementację testów funkcjonalnych.

Istnieją dwa ogólne rodzaje elastyczności, o których można mówić w kontekście wytwarzania oprogramowania — elastyczność zespołu oraz elastyczność techniczna (patrz rysunek 5.4). Elastyczność zespołu decyduje o jego zdolności do rozpoznawania i reagowania na zachodzące zmiany. Warunkiem właściwego reagowania na zmiany, przez dobór odpowiedniego oprogramowania, jest elastyczność techniczna. Oznacza to, że musimy zadbać zarówno o elastyczność zespołu, jak i o elastyczność techniczną. W pierwszej kolejności powinieneś wdrożyć praktyki zautomatyzowanych testów programisty, samoorganizującego się zespołu oraz trio iteracyjnego, czyli iteracji, stanu wykonania i listy zaległych zadań. O ile testowanie zapewni Ci osiągnięcie właściwej elastyczności technicznej, pozostałe wymienione praktyki podniosą elastyczność Twojego zespołu.



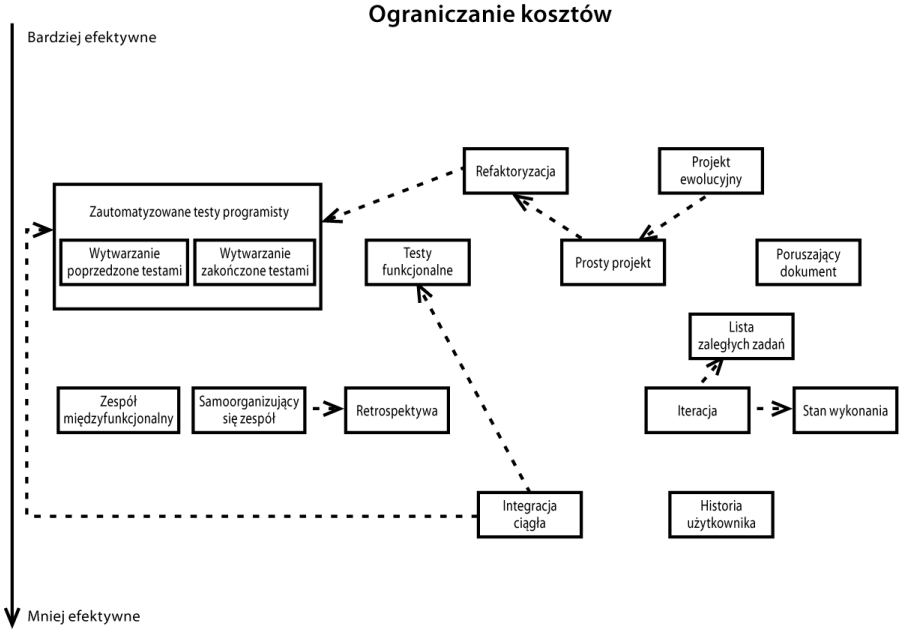
Rysunek 5.4. Praktyki zwiększające elastyczność

Lista zaległych zadań i radiatory informacji to pierwsze, jednocześnie stosunkowo łatwe działania na rzecz poprawy widoczności (patrz rysunek 5.5). W zależności od potrzeb w zakresie zwiększania widoczności możesz obrać albo stosunkowo prostą drogę z praktykami iteracji, stanu wykonania i wersji demonstracyjnych, albo trudniejszy, ale też bardziej efektywny model z testami funkcjonalnymi i wymaganiami sterowanymi testami.



Rysunek 5.5. Praktyki poprawiające widoczność

Koszty można ograniczać na dwa sposoby — możesz albo zadbać o uproszczenie konserwacji kodu, albo po prostu pisać mniej kodu (ograniczając się do implementowania najważniejszych funkcji). Wdrażanie kolejno praktyk zautomatyzowanych testów, refaktoryzacji, uproszczonych projektów i projektowania ewolucyjnego jest właściwym sposobem ograniczania kosztów utrzymania (konserwacji). Praktyki listy zaległych zadań, iteracji i stanu wykonania skutecznie zmniejszają ilość pisanego kodu (patrz rysunek 5.6).

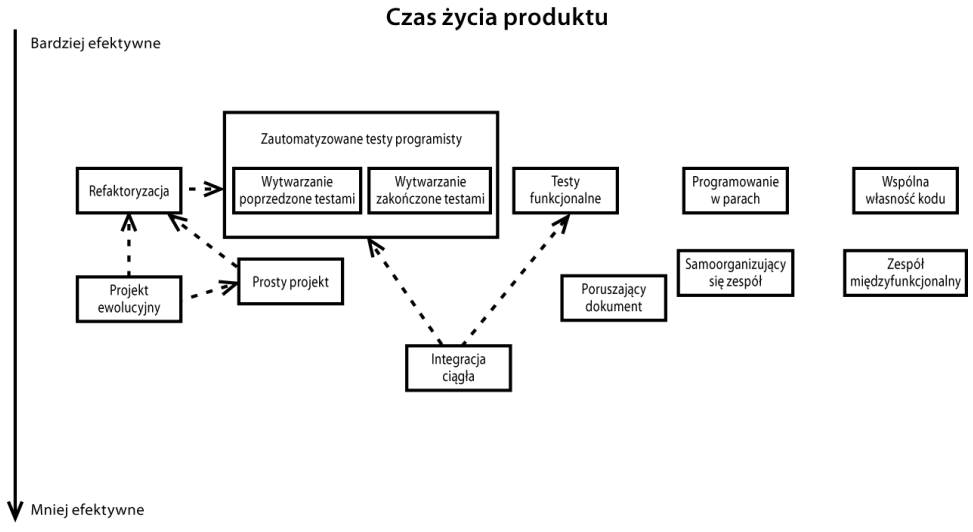


Rysunek 5.6. Praktyki redukujące koszty

Czas życia produktu jest odwrotnie proporcjonalny do kosztów jego konserwacji oprogramowania. Istnieją dwa sprawdzone sposoby ograniczania kosztów utrzymania: (1) sporządzenie siatki bezpieczeństwa złożonej z testów i efektywne wprowadzanie zmian w systemie z jednoczesnym ograniczaniem ich kosztów oraz (2) szerzenie wśród członków zespołu wiedzy o projekcie danego systemu. Kluczem do skutecznej realizacji pierwszego z tych zadań jest wdrożenie praktyki zautomatyzowanych testów programisty; drugie zadanie można zrealizować przez programowanie w parach i wspólną własność kodu.

WZORCE KOJARZENIA PRAKTYK ZWINNYCH Z PROBLEMAMI

Jak już wspomniano, istnieją dwa główne typy problemów: biznesowe oraz związane z procesem. Problemy biznesowe są w istocie przeciwieństwem walorów biznesowych, stąd identyczne wzorce kojarzenia odpowiednich praktyk zwinnych:



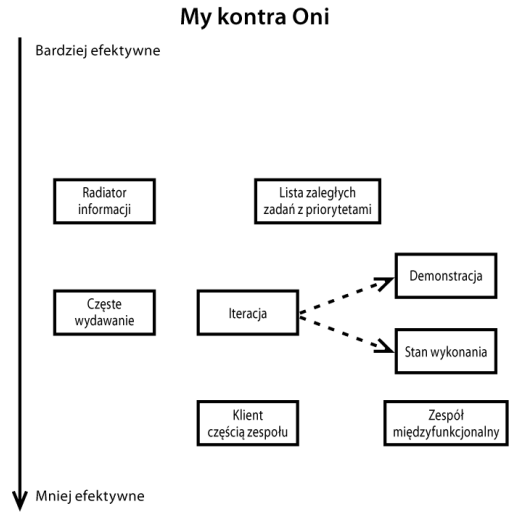
Rysunek 5.7. Praktyki wydłużające czas życia produktu

- jakość produktu przekazanego klientowi jest nie do przyjęcia — jakość produktu trafiającego na rynek;
- dostarczanie klientowi nowych funkcji trwa zbyt długo — czas wprowadzania produktu na rynek;
- zaimplementowane funkcje nie są wykorzystywane przez klienta — użyteczność produktu;
- oprogramowanie okazało się nieprzydatne dla klienta — użyteczność produktu;
- budowa oprogramowania jest zbyt droga — ograniczenie kosztów.

Dla pozostałych problemów istnieją odrębne odwzorowania we wzorce wdrażania praktyk zwinnych (pokazane na rysunkach od 5.8 do 5.15).

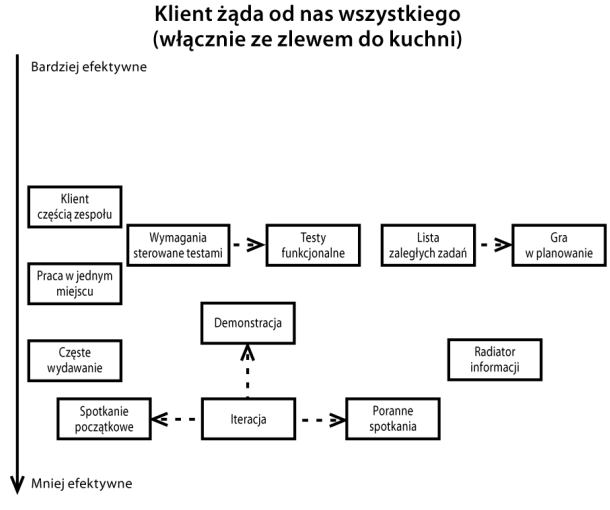
Najlepszym rozwiązaniem łagodzenia skutków występowania problemu określanego jako my kontra oni jest prowadzenie możliwie częstych konwersacji na temat rzeczywistej natury realizowanego projektu (patrz rysunek 5.8). Warto zacząć od poprawy widoczności przez tworzenie radiatorów informacji prezentujących najważniejsze aspekty wytwarzania. Powinieneś też (we współpracy z całym zespołem i klientami) sporządzić listę zaległych zadań i przypisać im odpowiednie priorytety. Spróbuj wykorzystać te praktyki do poprawy widoczności i budowy zaufania. Kiedy będziesz gotowy, rozważ wykonanie dalszych kroków na rzecz poprawy zaufania — dbaj o częste dostarczanie produktów przez wdrożenie praktyk iteracji, wersji demonstracyjnych i stanu wykonania.

Musisz zrozumieć, że kiedy klient żąda od Ciebie niemal wszystkiego, nie chodzi o jego rzeczywiste oczekiwania, tylko o brak wiary w możliwość punktualnego dostarczenia właściwych funkcji i obawę przed kosztownymi procesami renegocjacji raz podpisanych umów (co jest typowe dla



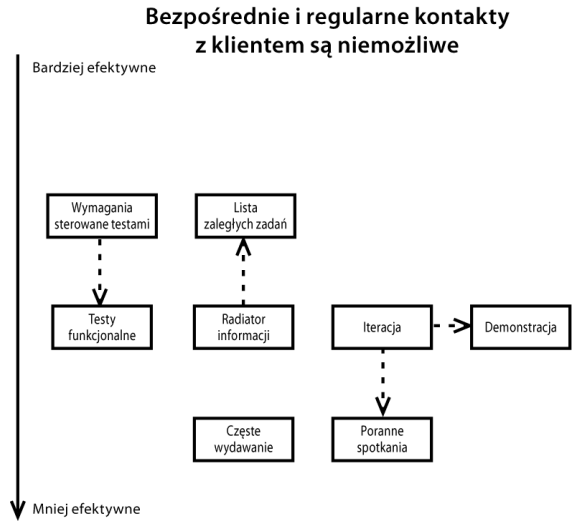
Rysunek 5.8. Praktyki rozwiązujące problem „my kontra oni"

tradycyjnych projektów informatycznych). Najlepszym sposobem rozwiązania tego problemu jest włączenie klientów do prac zespołu projektowego (patrz rysunek 5.9). Spróbuj zaangażować ich w proces sporządzania listy zaległych zadań i przypisywania tym zadaniom priorytetów.



Rysunek 5.9. Praktyki rozwiązujące problem żądania wszystkiego przez klienta

Jeśli bezpośrednie uzyskiwanie od klienta jego opinii i sugestii jest niemożliwe, możesz ograniczyć negatywne skutki tego stanu rzeczy, zmniejszając liczbę błędów komunikacyjnych (patrz rysunek 5.10). Można to zrobić przez przygotowanie testów funkcjonalnych i stopniowe dochodzenie

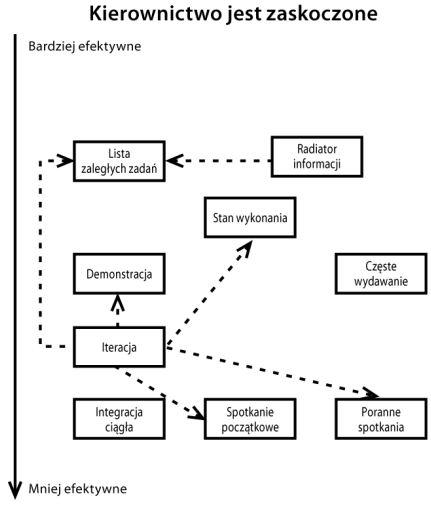


Rysunek 5.10. Praktyki rozwiązujące problem niemożności bezpośredniego i regularnego uzyskiwania informacji od klienta

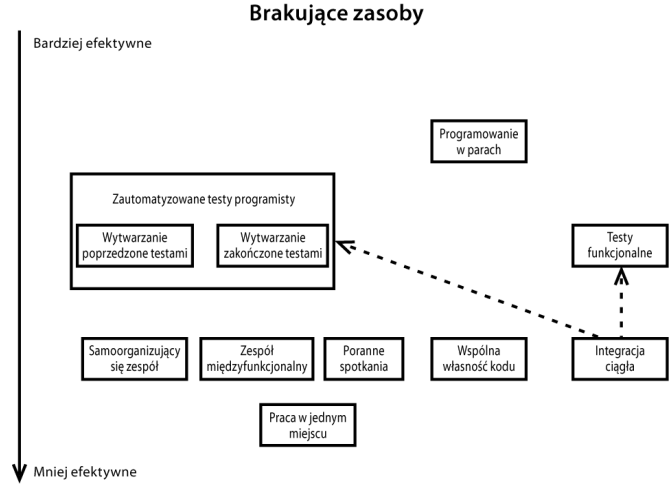
do praktyki wymagań sterowanych testami, gdzie sporządzony przez klienta dokument opisujący wymagania jednocześnie pełni funkcję wykonywalnej specyfikacji. Prawidłowe wdrażanie tej konkretnej praktyki jest wyjątkowo czasochłonne, zatem warto przystąpić do tego procesu możliwie wcześnie i wykazywać należytą cierpliwość. W międzyczasie warto sporządzić listę zaległych zadań i podejmować próby pracy przyrostowej z iteracjami kończonymi tworzeniem wersji demonstracyjnych.

Aby uniknąć niepotrzebnego zaskakiwania kierownictwa organizacji, powinieneś zrobić dwie rzeczy: (1) starać się budować swoje aplikacje w sposób przyrostowy oraz (2) informować swoich przełożonych o rzeczywistych postępach prac. Pierwszy cel można osiągnąć dzięki definiowaniu stanu wykonania możliwie bliskiego stanowi produktu gotowego do wdrożenia, po czym przystąpieniu do realizacji iteracji. Do komunikowania prawdziwych postępów prac można wykorzystywać radiatory informacji oraz wersje demonstracyjne prezentowane na końcu każdej iteracji.

Zjawisko brakujących zasobów ludzkich (stanowiących wąskie gardło) wynika z dążenia do specjalizacji. Programowanie w parach jest zdecydowanie najbardziej efektywną metodą dzielenia się wiedzy i — tym samym — zdobywania cennych specjalizacji przez pozostałych członków zespołu (patrz rysunek 5.12). Z czasem stosowanie tej praktyki doprowadzi do sytuacji, w której wielu członków zespołu będzie potrafiło rozwiązywać problemy, które wcześniej były domeną zaledwie jednego eksperta. Innym krokiem w dobrym kierunku jest wdrożenie praktyki zautomatyzowanych testów programistów, które umożliwiają członkom zespołu pracę na nieznanym sobie kodzie bez ryzyka uszkodzenia wcześniej zaimplementowanych i sprawdzonych rozwiązań (dzięki istnieniu zabezpieczającej sieci testów). Oznacza to, że istnienie tych testów i możliwość bezpiecznej pracy na cudzym kodzie może skutecznie niwelować skutki niedoboru specjalistów.

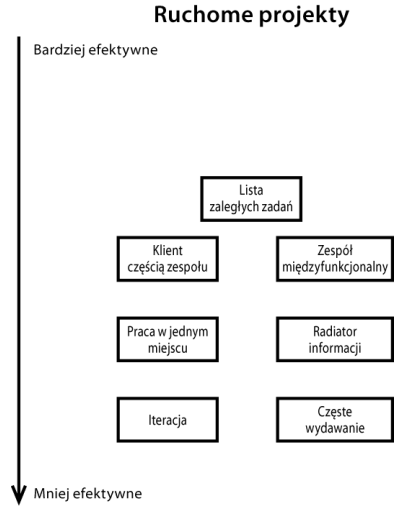


Rysunek 5.11. Praktyki rozwiązujące problem zaskoczenia zarządu (braku widoczności)



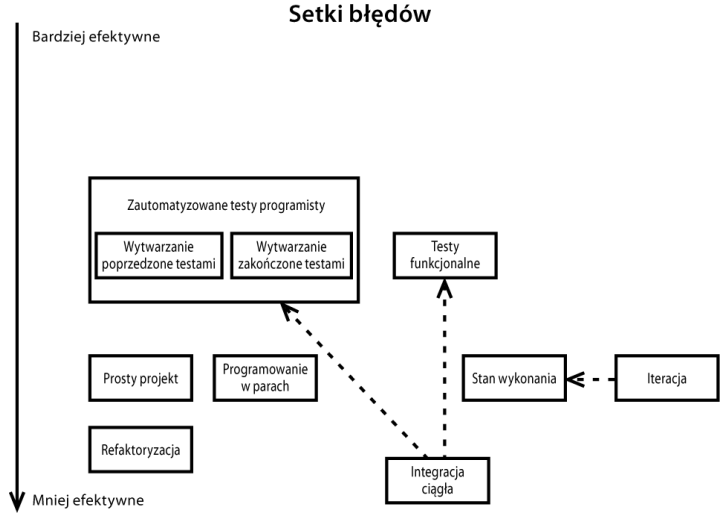
Rysunek 5.12. Brakujące zasoby — specjaliści od budowy oprogramowania są członkami wielu zespołów jednocześnie stosujących różne praktyki

Projekty sprawiają wrażenie ruchomych, niestałych w wyniku braku jasno zdefiniowanych priorytetów. Priorytety należy przypisywać wymaganiom, sporządzając i stale utrzymując listę zadań do wykonania. Aby mieć pewność, że taka lista precyzyjnie odpowiada potrzebom klienta, warto włączyć przedstawiciela tego klienta do prac zespołu i utworzyć zespół międzyfunkcjonalny zdolny do kompleksowej realizacji tych wymagań. W ten sposób umożliwisz zarówno sobie, jak i swoim klientom lepsze zrozumienie wymagań, ich priorytetów i pętli zwrotnej decydującej o możliwie szybkim wprowadzaniu korekt.



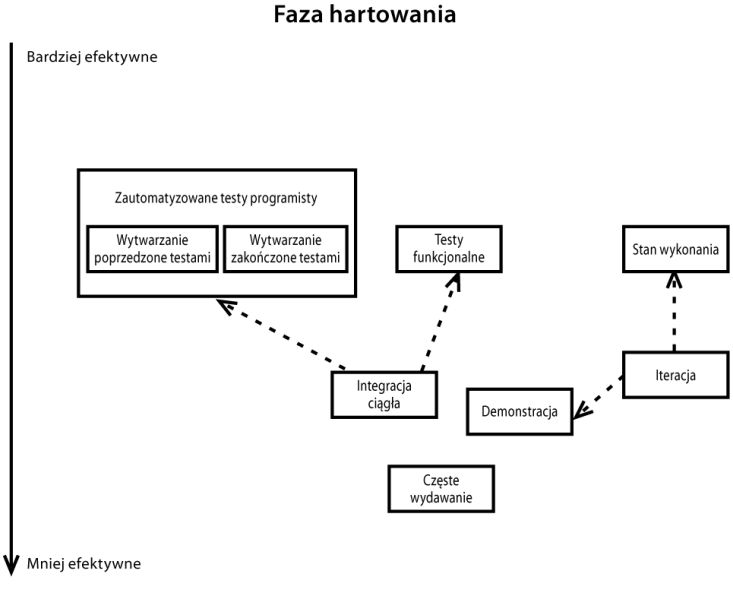
Rysunek 5.13. Praktyki rozwiązujące problem ruchomych projektów

Musisz sobie poradzić z setkami błędów. W pierwszej kolejności powinieneś wdrożyć praktykę zautomatyzowanych testów programisty, wspartą praktyką programowania w parach, aby ograniczyć liczbę nowych błędów wprowadzanych do systemu i przystąpić do stopniowej budowy sieci bezpieczeństwa (złożonej z testów). Warto następnie wdrożyć iteracje ze stanem wykonania, aby odnaleźć jak najwięcej błędów. Nigdy nie odkładaj na później tak kłopotliwych zadań jak integracja — staraj się rozwiązywać problemy możliwie szybko.



Rysunek 5.14. Praktyki rozwiązujące problem setek (lub tysięcy) usterek zarejestrowanych w systemie śledzenia błędów

Konieczność stosowania fazy „hartowania", „utwardzania" jest symptomem kumulacji kilku ważnych defektów. Wstrzymaj wszystkie bieżące prace i skoncentruj swoje wysiłki na dodaniu niezbędnych testów, aby zyskać możliwość wytwarzania kodu z wykorzystaniem zautomatyzowanych testów programisty. W ramach każdej iteracji sformułuj dobry stan wykonania (możliwie bliski stanu gotowego do wdrożenia), aby możliwie wcześnie eliminować trudne do odnajdowania błędy.



Rysunek 5.15. Praktyki rozwiązujące problem konieczności stosowania fazy „utwardzania" na końcu każdego cyklu wydawania

WYPRACOWYWANIE WŁASNEJ STRATEGII WDRAŻANIA PRAKTYK ZWINNYCH

Zgromadzone do tej pory informacje o wartościach i problemach biznesowych możesz wykorzystać do identyfikacji praktyk, które powinieneś wdrożyć we własnej organizacji.

- **Wybierz właściwe praktyki, kierując się wyłącznie walorami biznesowymi swoich produktów.** W tym scenariuszu przyjmujemy, że Twoja organizacja nie napotyka żadnych poważnych utrudnień. Chcesz po prostu udoskonalić swój obecny proces wytwarzania oprogramowania przez zwiększenie walorów biznesowych produktów dostarczanych przez Twój zespół. Wykorzystaj odwzorowania pokazane na rysunkach od 5.1 do 5.7, aby wybrać praktyki, które będą miały największy wpływ na walory biznesowe Twojej organizacji.

- **Wybierz praktyki z myślą o ograniczeniu negatywnych skutków problemów, których waga zależy od priorytetów odpowiednich walorów biznesowych.** Ten model ma na celu eliminację przeszkód z jednoczesnym dążeniem do maksymalizacji walorów biznesowych. Priorytety przypisywane tym problemom powinny zależeć od walorów biznesowych szczególnie ważnych dla Twoich klientów. Oznacza to, że w pierwszej kolejności powinieneś sporządzić listę problemów (z priorytetami), po czym wybrać odpowiednie praktyki do wdrożenia — możesz posłużyć się odwzorowaniami pokazanymi na rysunkach od 5.8 do 5.15.
- **Wybierz praktyki eliminujące najbardziej widoczne problemy.** Taka postawa jest dość popularna, choć sam jestem daleki od jej propagowania. Opisany model można by porównać do gaszenia pożarów — zamiast koncentrować się na walorach biznesowych, próbujemy bezrefleksyjnie eliminować największe przeszkody. To nieskuteczne podejście jest udziałem zbyt wielu zespołów technicznych, które decydują się na przypisywanie priorytetów swoim zadaniom bez uwzględniania opinii i sugestii klienta. (Sam popełniałem w przeszłości podobne błędy).

Informacje (praktyki) zawarte na rysunkach w poprzednich podrozdziałach uporządkowano według efektywności. Oznacza to, że pierwsza praktyka na każdym rysunku jest najbardziej efektywnym sposobem podnoszenia określonego waloru biznesowego lub ograniczania skutków występowania danego problemu. Warto więc zacząć właśnie od pierwszej praktyki, a po jej skutecznym wdrożeniu wrócić do pozostałych praktyk związanych z interesującym nas walorem lub problemem biznesowym.

Niezależnie od tego, jak dobierzesz priorytety w ramach swojej listy praktyk do wdrożenia, powinieneś przeprowadzać sam proces ich wdrażania w sposób iteracyjny. Wyposażony w listę wybranych wcześniej praktyk zwinnych możesz przystąpić do wdrażania ich według następującej procedury.

1. Zacznij od oceny stanu obecnego. Zapoznaj się z istniejącymi materiałami (nawet jeśli mają subiektywny charakter) poświęconymi bieżącym walorom biznesowym, które chcesz rozszerzyć, oraz problemom, których skutki chcesz ograniczyć.
2. Wyznacz cele, które chcesz osiągnąć. O ile chcesz podnieść wybrany walor biznesowy? Na ile chcesz zredukować skutki wybranego problemu? Jakie ramy czasowe przyjąłeś? Spróbuj sformułować początkowe odpowiedzi na te pytania, by następnie modyfikować je wraz z zyskiwaniem konkretnych doświadczeń.
3. Wybierz pierwszą praktykę (lub grupę praktyk) ze sporządzonej wcześniej listy.
4. Zapoznaj się ze wzorcem wdrażania danej praktyki lub grupy praktyk. Spróbuj określić, czy interesujący Cię wzorzec znajduje zastosowanie w Twoim środowisku roboczym (więcej informacji na temat tych wzorców znajdziesz w części 3, zatytułowanej „Katalog wzorców"). Jeśli dana praktyka z jakiegoś powodu nie może być zastosowana w Twoim środowisku, wróć do swojej listy i wybierz następny element podnoszący jakiś walor biznesowy lub ograniczający skutki jakiegoś problemu.

5. Kiedy ostatecznie rozstrzygniesz, że dany wzorzec można zastosować w Twoim środowisku, zapoznaj się z jego szczegółowym opisem. Początkowo powinieneś postępować według wskazówek zawartych w podrozdziale „Wdrażanie" odpowiedniego rozdziału.
6. W stałych odstępach czasu staraj się oceniać interesujący Cię walor biznesowy pod kątem ewentualnej poprawy lub problem pod kątem redukcji jego negatywnych skutków. Jeśli nie zanotowałeś poprawy od ostatniej oceny (sprzed wdrożenia danej praktyki w Twoim środowisku), zastosuj wskazówki zawarte w podrozdziałach „Odmiany" i „Ale" rozdziału poświęconego tej praktyce. (Być może powinieneś wcześniej przeczytać rozdział 6., zatytułowany „Wzorce wdrażania praktyk zwinnych", aby lepiej zrozumieć przebieg procedury wdrażania zwinnych praktyk wytwarzania oprogramowania).
7. Wróć do kroku 1. i ponownie oceń swój walor lub problem biznesowy. Jeśli dany obszar wymaga dodatkowych udoskonaleń (jeżeli nadal nie osiągnięto celu wyznaczonego w kroku 2.), powinieneś rozważyć dodanie innej praktyki lub całej grupy praktyk, aby zbliżyć się do tego celu. Jeśli udało Ci się osiągnąć sformułowane cele, przejdź do następnego elementu listy.

Gdzie w opisanej procedurze jest miejsce na działania sterowane testami? Twoje testy powinny się składać na cel sformułowany w kroku 2. W kroku 6. powinieneś ocenić swój proces już po wdrożeniu nowej praktyki. Istotą kroku 6. jest przetestowanie efektywności wdrożonej praktyki lub praktyk w zakresie osiągnięcia wyznaczonego celu. Opisana pętla (wyznaczenie celu, wdrożenie praktyki i weryfikacja tej praktyki pod kątem skuteczności w dochodzeniu do oczekiwanego celu) jest typową strategią wdrażania sterowanego testami[1].

CO DALEJ?

Po lekturze tego rozdziału jesteś gotowy do opracowania własnej strategii wdrażania zwinnych praktyk wytwarzania oprogramowania z uwzględnieniem konkretnych walorów biznesowych i problemów Twojej organizacji. Tworzenie tej strategii nie jest czynnością jednorazową — powinieneś wykazywać zwinność także w trakcie wdrażania nowych praktyk. Oceniaj obserwowane postępy pod kątem spełnienia wyznaczonych celów i stale modyfikuj stosowane procedury. Wraz z zyskiwaniem nowej wiedzy o poszczególnych praktykach i własnym środowisku modyfikuj swoją strategię. Popełnianie błędów jest czymś zupełnie naturalnym — kluczem do sukcesu jest odkrywanie niepowodzeń możliwie szybko i korygowanie błędnych założeń.

[1] W świecie praktyk zarządzania opisany model bywa określany mianem cyklu PDCA (od ang. *Plan, Do, Check, Act* — planuj, wykonaj, sprawdź, działaj). Oryginalna koncepcja PDCA z lat trzydziestych ubiegłego wieku jest co prawda dziełem Waltera Shewharta z Bell Laboratories, jednak prawdziwą popularność zyskała dopiero w latach pięćdziesiątych dzięki pozytywnej opinii guru zarządzania jakością, W. Edwarda Deminga.

W dalszej części tej książki szczegółowo omówimy każdą ze wspomnianych w tym rozdziale praktyk. Dla każdej z nich istnieje specjalny wzorzec, który nie tylko opisuje tę praktykę, ale też rozwiązywane przez nią problemy, udokumentowane scenariusze jej skutecznego zastosowania w przeszłości (z uwzględnieniem specyficznych cech środowisk i popełnionych błędów, abyśmy wiedzieli, na co powinniśmy zwracać szczególną uwagę).

OD TEORII DO PRAKTYKI — BUDOWA WŁASNEJ STRATEGII WDRAŻANIA PRAKTYK ZWINNYCH

Spróbuj odpowiedzieć na następujące pytania związane z przygotowywaniem strategii wdrażania. (Wykorzystaj odpowiedzi na pytania postawione w rozdziałach 3. i 4.). Rzeczywiste przykłady tworzenia tego rodzaju strategii można znaleźć w rozdziałach 45. i 46.

1. Jakie cele chcesz osiągnąć przez wdrażanie praktyk zwinnych? Czy chcesz ograniczyć negatywne skutki występowania jakiś problemów, czy raczej podnieść walory biznesowe swojego produktu? Odpowiedz konkretnie — jeśli stawiasz sobie więcej niż jeden cel, przypisz im priorytety.
2. Zapoznaj się z dostępnymi materiałami na temat bieżących walorów biznesowych i problemów wymagających rozwiązania. Nie przejmuj się, jeśli istniejące raporty sprawiają wrażenie subiektywnych lub nieścisłych. Spróbuj uzyskać możliwie pełną wiedzę o obecnym stanie Twojej organizacji, zarówno w wymiarze walorów biznesowych, jak i problemów.
3. Wybierz strategię wdrażania nowych praktyk. Wykorzystaj tę strategię do identyfikacji właściwych praktyk.
4. Zapoznaj się z treścią następnego rozdziału, w którym wprowadzono zagadnienia związane ze wzorcami. Po jego lekturze możesz przystąpić do przeprowadzania opisanej powyżej procedury wdrażania pierwszej z wybranych praktyk. Nie zapominaj o konieczności okresowej weryfikacji walorów i problemów biznesowych, aby mieć pewność, że stosowana praktyka przynosi zamierzony efekt.
5. Gratulacje i powodzenia! Właśnie wstąpiłeś na ścieżkę wdrażania zwinnych praktyk programowania!