Bezpieczne oprogramowanie. Metodologia, techniki i narzędzia projektowania

Autor: Bijay K. Jayaswal, Peter C. Patton
ISBN: 978-83-246-1463-9
Tytuł oryginału: Design for Trustworthy Software: Tools, Techniques, and Methodology of Developing Robust Software
Format: 172x245, stron: 816

Poznaj narzędzia, techniki oraz metodologię tworzenia niezawodnego oprogramowania

- Jak przeprowadzić weryfikację, oceniać i konserwować oprogramowanie?
- Jakie są finansowe aspekty tworzenia oprogramowania godnego zaufania?
- Jak zarządzać portfelem technologii informatycznych?

Jakość oprogramowania to wielowarstwowe zagadnienie. Spojrzenie na nią z kilku perspektyw jest kluczowe dla procesu tworzenia nowego produktu. Należy przy tym wziąć pod uwagę nie tylko opłacalność jego wytwarzania i konkurencyjność, ale także jawne i ukryte potrzeby naszych klientów oraz partnerów biznesowych. Stąd wynika potrzeba używania zintegrowanej technologii, pomagającej spełniać wszystkie te wymagania. Odpowiada na nią technologia projektowania oprogramowania godnego zaufania (ang. Designing for Trustworthy Software). Służy ona wczesnemu rozwiązywaniu problemów związanych z jakością tworzonego produktu, dzięki czemu zapobiega powstawaniu błędów implementacji. Siłą tej technologii jest także fakt, że wszelkie działania związane z jakością są podejmowane przed napisaniem każdego wiersza kodu.

Ta książka pomoże w poprawie jakości wszystkim tym, którzy wdrażają rozwiązania wewnętrzne i zewnętrzne, prowadzą konsultacje i świadczą pomoc techniczną. Zawiera ona przełomowe rozwiązania dla specjalistów z dziedziny oprogramowania oraz jakości – od programistów po liderów projektu, od głównych architektów oprogramowania po klientów. Z tego podręcznika dowiesz się m.in., jak stosować najlepsze praktyki w dziedzinie kontrolowania jakości, organizacji szkoleń i zarządzania w wyjątkowym środowisku rozwoju oprogramowania.

- Metodologia rozwoju oprogramowania
- Miary jakości oprogramowania
- Koszty jakości oprogramowania
- Narzędzia i techniki projektowania oprogramowania godnego zaufania
- Analityczny proces hierarchiczny
- Złożoność, błędy i poka-yoke w procesach rozwoju oprogramowania
- Ocena ryzyka oraz analiza przyczyn i skutków błędów (FEMA)
- Technologie obiektowe i komponentowe
- Techniki AHP, TRIZ, CoSQ i metoda Taguchiego
- Integracja, wzbogacanie i konserwacja oprogramowania godnego zaufania
- Wdrażania technologii DFTS
- QFD dla projektów

Twórz niezawodne oprogramowanie wysokiej jakości!

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

Spis treści

ROZDZIAŁ 2

Wyzwania na drodze do oprogramowania godnego zaufania — solidny projekt w kontekście oprogramowania

Projektuj produkty tak, aby nie zawodziły w praktyce.
Tym samym zmniejszysz liczbę defektów w fabryce.
Genichi Taguchi

Poprawa wymaga zmian. Doskonałość wynika z częstego ich wprowadzania.
Winston Churchill

Omówienie

Wieloaspektowe spojrzenie na jakość jest kluczowe w identyfikowaniu licznych wymagań klientów i innych partnerów. Występują niezwykłe podobieństwa między zagadnieniami związanymi z jakością oprogramowania i wyrobów produkowanych, jednak należy uwzględnić także istotne różnice. Koszty powodowane przez oprogramowanie niskiej jakości są coraz bardziej kluczowe, ponieważ koszt cyklu życia typowego systemu przekracza cenę sprzętu. Oprogramowanie wyższej jakości pozwala istotnie zredukować te koszty, ponieważ 80 – 90% ich sumy pochłania konserwacja związana z poprawianiem, adaptowaniem i rozwijaniem udostępnionego oprogramowania. Obecnie około 40% wydatków na rozwój oprogramowania pochłaniają testy potrzebne w celu usunięcia błędów. Kluczowa jest także niezawodność oprogramowania, ponieważ stosunek błędów występujących w programach do usterek sprzętowych może wynosić 100:1, a nawet jeszcze więcej. W tym rozdziale opisujemy niepowodzenia tradycyjnych systemów zarządzania jakością w kontekście udostępniania oprogramowania godnego zaufania. Proponujemy zintegrowaną technologię rozwoju oprogramowania — projektowanie oprogramowania godnego zaufania (ang. *Design for Trusthworthy Software* — DFTS) — bazującą na trzech kluczowych elementach: iteracyjnym modelu rozwoju solidnego oprogramowania, inżynierii optymalizacji projektu oprogramowania i technologii projektowania obiektowego. W DFTS wysiłki nad zapewnieniem jakości koncentrują się na wczesnych etapach procesu rozwoju. Technologia ta

pozwala na ciągłą interakcję z użytkownikami i między osobami pracującymi nad produktem na różnych etapach, pomaga uwzględnić opinie klientów oraz umożliwia wczesną i ciągłą analizę ryzyka, optymalizację projektu i zastosowanie odpowiedniej technologii rozwoju oprogramowania. W tym rozdziale podkreślamy kluczową rolę prawdziwego zaangażowania ze strony menedżerów na drodze do tworzenia oprogramowania godnego zaufania.

Struktura rozdziału

- Niezawodność oprogramowania — fakty i mity
- Ograniczenia tradycyjnych systemów kontroli jakości
- Japońskie systemy zarządzania jakością i podejście Taguchiego
- Istota metod Taguchiego do tworzenia solidnych projektów
- Wyzwania na drodze do niezawodności oprogramowania — projektowanie oprogramowania godnego zaufania
- Model rozwoju solidnego oprogramowania — proces DFTS w praktyce
- Kluczowe zagadnienia
- Dodatkowe materiały
- Ćwiczenia internetowe
- Pytania przeglądowe
- Zagadnienia do dyskusji i projekty
- Przypisy

Niezawodność oprogramowania — fakty i mity

Jakość oprogramowania, podobnie jak jakość sprzętu, to wieloaspektowe zagadnienie. W rzeczywistości spojrzenie na jakość z kilku perspektyw jest kluczowe do zrozumienia i utworzenia wartościowego produktu oraz zaspokojenia zestawu jawnych i ukrytych potrzeb klienta. Producent musi także spełnić wymania wielu innych partnerów, takich jak audytorzy, dostawcy, związki branżowe, media i inne zainteresowane grupy. W końcu, co nie najmniej istotne, każdy wytwórca musi się upewnić, że jego produkty są opłacalne i konkurencyjne, aby mogły spełniać potrzeby właścicieli przedsiębiorstw. Jakość obejmuje wszystkie takie potrzeby i wymagania.

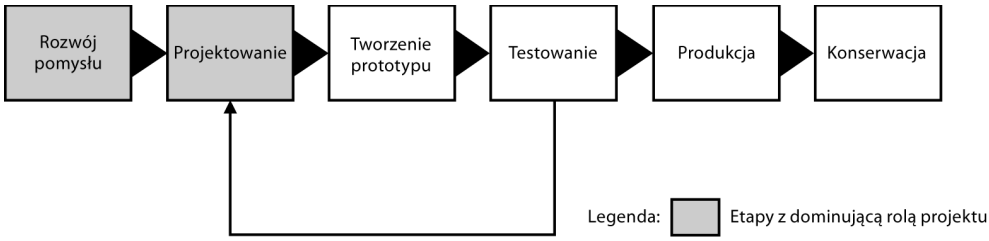
Często można usłyszeć, że zagadnienia związane z jakością w świecie oprogramowania są inne niż w przypadku innych produktów, jednak nie do końca jest to prawdą. Ogólnie mówiąc, zasady, systemy i metodologie jakościowe stosowane do produkcji wyrobów są równie prawidłowe w przypadku oprogramowania, sprzętu i innych dóbr czy usług. Jednak oprogramowanie ma specyficzne środowisko projektowania i rozwoju, które trzeba zrozumieć. Ponadto należy poznać specyficzne wyzwania wynikające z abstrakcyjnej natury systemów cyfrowych oraz poziomu złożoności wielu aplikacji, a także wziąć pod uwagę innowacyjność i trudność zadań, do których rozwiązywania zostały zaprojektowane.

Wierzymy, że w organizacjach zajmujących się rozwojem oprogramowania oraz takich, dla których tworzenie aplikacji jest istotną działalnością, zagadnienia związane z architekturą i projektowaniem są kluczowe dla długoterminowej opłacalności i powodzenia firmy. We wszystkich takich organizacjach te zadania są zbyt ważne, aby pozostawić je wyłącznie inżynierom oprogramowania. Problem produkcji oprogramowania godnego zaufania jest prawdziwym wyzwaniem i wymaga całkowitego zaangażowania kadry zarządzającej. W tej książce przedstawiamy podstawy filozoficzne, system zarządzania oraz technologię do zarządzania jakością oprogramowania zarówno w dużych, jak i małych firmach, ze szczególnym uwzględnieniem oprogramowania dla przedsiębiorstw.

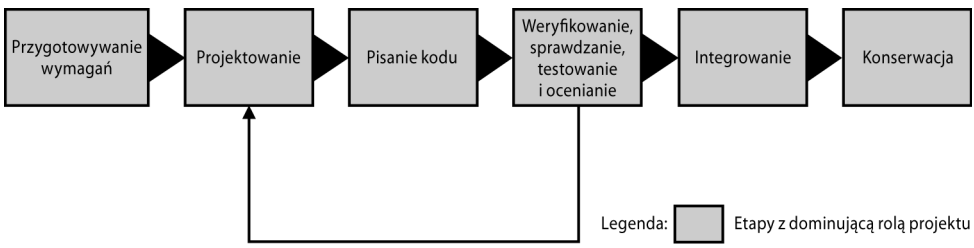
Podobieństwa i różnice między oprogramowaniem i wyrobami produkowanymi

Zrozumienie różnic między oprogramowaniem i produkowanymi wyrobami jest niezbędne do zaprojektowania solidnego systemu zarządzania jakością oprogramowania. Jedynie wtedy można zaadaptować systemy i metodologie, które przez ostatnie 50 lat miały tak znaczący wpływ na poprawę jakości wszelkich dóbr produkowanych, a także innych wyrobów i usług. Trzeba jednak uwzględnić także ważne różnice, aby prawidłowo rozwinąć system zarządzania jakością oprogramowania. Poniżej opisane są pewne związane z tym tematem zagadnienia:

- Oprogramowanie i wyroby produkowane różnią się w zakresie znaczenia różnych etapów w procesie ich tworzenia (zobacz rysunki 2.1 i 2.2). Rozwój oprogramowania charakteryzuje się centralizacją projektu. Oprogramowanie to przykład czystego



RYSUNEK 2.1.
Podstawowe etapy rozwoju wyrobów produkowanych



RYSUNEK 2.2.
Podstawowe etapy rozwoju oprogramowania

projektu, w którym wszystkie operacje są powiązane właśnie z projektem. Dlatego problemy z jakością i niezawodnością są zawsze wynikiem błędów projektowych, które są z kolei wynikają z pewnych braków poznawczych.

- W oprogramowaniu nie ma niczego, co można porównać do produkcji czy montażu, kiedy to można zrekompensować, a nawet naprawić błędy popełnione na etapie projektowania. W przypadku oprogramowania produkcja w zasadzie nie ma miejsca. Sam kod programu jest po prostu dalszym etapem projektu. Ponadto nie można tu powiedzieć, że produkt jest „wykonany i dostarczony". Zawsze można zaprojektować aplikację od nowa, zmodyfikować ją lub zaktualizować, a przez to zmienić. Ta charakterystyczna dla oprogramowania elastyczność często prowadzi do podejścia „dostarczmy to teraz — błędy zawsze będziemy mogli poprawić później".
- W odróżnieniu od wyrobów produkowanych, w przypadku oprogramowania nie ma odpowiednika etapu analizy projektu. Większość analiz (testów) trzeba przeprowadzać na kodzie programu. Dlatego problemy lub pomyłki projektowe mogą pozostać ukryte aż do późnych etapów procesu rozwoju lub nawet do czasu rozpoczęcia korzystania z aplikacji.

Można także zauważyć, że obecnie działalność wielu producentów sprzętu w coraz większym stopniu zależy od oprogramowania. Takie złożone systemy windują na następny poziom wyzwania w zakresie projektowania programów.

Porównywanie niezawodności oprogramowania i sprzętu

Zawodność sprzętu wynika głównie z fizycznych błędów pojedynczych komponentów, co jest często konsekwencją przewrotności natury. Z kolei w oprogramowaniu usterki charakteryzują się nieciągłym działaniem systemów cyfrowych. Wiedza o projektowaniu i inżynierii urządzeń jest łatwiejsza do udokumentowania w celu zapobieżenia błędom niż wiedza o oprogramowaniu. Ponadto teorie awarii sprzętu są rozwijane od lat i umożliwiają zapewnienie wysokiej niezawodności w wyrobach produkowanych[1]. Dla porównania, baza wiedzy o niezawodności oprogramowania jest bardzo mała — stąd nieodłączne problemy w zapewnianiu stabilności działania aplikacji.

Oprogramowaniu zawsze towarzyszą urządzenia. Jeśli jednak znana jest niezawodność komponentu sprzętowego, zawsze można zoptymalizować pewność działania systemu, dołączając jedynie komponenty programowe[2]. Każdy system składający się z oprogramowania i sprzętu może zawieść z powodu usterek aplikacji wywołanej za pomocą zewnętrznych poleceń. Awaria oprogramowania jest definiowana jako odchylenie od oczekiwanych wyników zewnętrznych lub niezgodność danych wyjściowych programu z określonymi wymaganiami. Oprogramowanie może zawieść, kiedy jest używane w nieoczekiwanej sytuacji. Zwykle awaria może wystąpić z winy oprogramowania lub nieprzewidzianych warunków jego wykorzystania. Inaczej mówiąc, aby wystąpił błąd, program trzeba uruchomić.

Bieżący trend kosztów systemów sprzętowo-programowych zmierza w kierunku nieproporcjonalnej dominacji oprogramowania. Koszt cyklu życia (LCC) typowego oprogramowania przekracza cenę sprzętu, a 80 – 90% tych wydatków wiąże się z konserwacją aplikacji w zakresie naprawiania, adaptowania i rozwijania udostępnionego programu w celu zaspokojenia zmieniających się i rosnących potrzeb użytkowników. Około 40% ceny rozwoju oprogramowania pochłaniają testy mające na celu usunięcie błędów i zapewnienie wysokiej jakości programu[3].

Stosunek zawodności oprogramowania do sprzętu może wynosić aż 100:1 w typowych komputerach bazujących na obwodach scalonych[4]. Dla bardziej skomplikowanych chipów ten stosunek może być jeszcze wyższy, co daje bardzo wyraźne wskazówki dotyczące kosztów i jakości. W tabeli 2.1 zamieszczono przegląd podstawowych różnic między niezawodnością sprzętu i oprogramowania.

Przyczyny zawodności oprogramowania

Zawodność oprogramowania to istotny problem społeczny. W rzeczywistości ma on globalne znaczenie i na jego rozwiązanie poświęca się znaczne zasoby. W poniższych punktach opisujemy podstawowe przyczyny zawodności oprogramowania.

- **Brak zaangażowania kadry zarządzającej**. Najczęstszą przyczyną problemów z jakością jest brak zaangażowania, poświęcenia i wsparcia kadry zarządzającej. Deming[5] oszacował kiedyś, że powody związane z zarządzaniem mogą odpowiadać

TABELA 2.1.
Różnice i podobieństwa w niezawodności sprzętu i oprogramowania[6]

Kategoria	Niezawodność sprzętu	Niezawodność oprogramowania
Podstawowa przyczyna	Z powodu efektów fizycznych	Z powodu błędów programisty (lub defektów albo awarii programu)
Przyczyny w cyklu życia		
Analizy	Nieprawidłowe zrozumienie klienta	Nieprawidłowe zrozumienie klienta
Wykonalność	Błędne wymagania użytkownika	Błędne wymagania użytkownika
Projekt	Błędy w fizycznym projekcie	Błędy w projekcie programu
Rozwój	Problemy z kontrolą jakości	Błędy w kodzie programu
Działanie	Usterka i awaria	Błędy programu (lub inne defekty albo awarie)
Efekty stosowania		
Funkcja projektu		
Dziedziny	Sprzęt zużywa się i zaczyna zawodzić	Oprogramowanie nie zużywa się, ale zawodzi w wyniku nieznanych defektów lub błędów
Relacje czasowe		Umiejętności programisty
Modele matematyczne	Fizyka błędu	Czas i dane
	Czas (t)	Funkcja malejąca
Obszar czasu	„Krzywa wannowa”	Dobrze rozwinięta teoretycznie, ale o niskiej akceptacji
	Dobrze rozwinięta teoretycznie i akceptowana	$R = f$(błędy [lub defekty albo awarie], t)
Funkcje	$R = f(\lambda, t)$, λ = intensywność błędów	Brak zgodności między różnymi proponowanymi modelami funkcji czasu
	Wykładnicza (stała λ)	Błędy = f(testy na danych)
	Weibulla (rosnąca λ)	Istnieje kilka modeli
Obszar danych	Bez znaczenia	Intensywność błędów, liczba wykrytych lub pozostałych defektów (lub awarii)
Modele wzrostu	Istnieje kilka modeli	
Miary	λ, MTBF (ang. *mean time between failures*, czyli średni czas pomiędzy awariami)	
	MTTF (ang. *mean time to failure*, czyli średni czas do wystąpienia awarii)	

[6] Przedruk za pozwoleniem z W. Kuo, V. Rajendra Prasad, F. A. Tillman i Ching-Lai Wang. *Optimal Reliability Design*. Cambridge University Press, Cambridge, 2001, punkt 13.5.1, s. 4.

TABELA 2.1.
Różnice i podobieństwa w niezawodności sprzętu i oprogramowania — ***ciąg dalszy***

Kategoria	Niezawodność sprzętu	Niezawodność oprogramowania
Techniki wzrostu	Projektowanie, przewidywanie	Przewidywanie
Techniki przewidywania	Diagramy blokowe, drzewa błędów	Analiza ścieżek (analiza wszystkich ścieżek to nierozwiązywalny problem, ponieważ liczba możliwości w nawet prostych programach może zmierzać do nieskończoności), złożoność, symulacje
Testy i ewaluacja	Akceptacja projektu i produkcji	Akceptacja projektu
Projekt	MIL-STD-781C (wykładnicze) Inne metody (niewykładnicze)	Testowanie ścieżek, symulacje, błędy, posiew Bayesa
Operacje	MIL-STD-781C	Brak
Zastosowanie nadmiaru		
Równoległość	Może poprawiać niezawodność	Trzeba rozważyć najczęstsze przyczyny
Rezerwa	Automatyczne wykrywanie i poprawianie błędów, automatyczne wykrywanie usterek i przełączanie	Automatyczne wykrywanie i poprawianie błędów, automatyczne kontrolowanie i ponowne inicjowanie oprogramowania
Logika większościowa	m elementów z n	Niepraktyczna

za około 85% ogólnych problemów z jakością w organizacji. Później Deming zrewidował te szacunki i podniósł je do 94%. Dotyczy to zarówno oprogramowania, jak i wyrobów produkowanych.

- **Niewystarczająca interakcja z użytkownikami**. Środowisko i wymagania użytkowników nie zostały prawidłowo zrozumiane. Powszechnie uważa się, że należy dążyć do poznania opinii klientów i dobrze zrozumieć oraz zinterpretować ich jawne i niejawne wymagania. Niestety, udział użytkowników w rozwoju oprogramowania po napisaniu i uzgodnieniu specyfikacji jest często znikomy. Ten brak ciągłej interakcji z użytkownikami oraz ich zmieniającymi się i ewoluującymi wymaganiami nie zawsze jest dostrzegany w procesie rozwoju oprogramowania. Jest to istotna przyczyna problemów z jakością oprogramowania i trzeba temu poświęcić należytą uwagę.
- **Rosnąca złożoność**. Systemy oprogramowania są tworzone do obsługi problemów o rosnącej złożoności. Często nie ma ręcznych rozwiązań, które mogłyby pomóc w zrozumieniu natury istniejących trudności. Oprogramowanie umożliwia

projektantowi zmierzenie się z ogromnymi trudnościami i udostępnienie dodatkowych funkcji oraz udogodnień, które nie zawsze zwiększają prawdziwą wartość programu. Złożone systemy oprogramowania używane do automatycznej kontroli lotu, w silnikach do masowego wyszukiwania, w handlu elektronicznym czy do zarządzania globalnymi przelewami gotówki w różnych walutach nie mają ręcznych odpowiedników, z którymi można je porównać. Trudność i innowacyjność prowadzą do złożoności oraz związanych z nią komplikacji projektowych i poznawczych, z czego wynikają wyzwania w rozwoju oprogramowania w obszarze niezawodności, bezpieczeństwa i zabezpieczeń.

- **Brak uzgodnionych kryteriów**. W nowych i złożonych systemach programista często tworzy kryteria niezawodności, które nie zawsze spełniają wymagania użytkowników.
- **Presja czasu związana z konkurencyjnością**. Konkurencyjna potrzeba skracania czasu cyklu rozwoju („możemy to później naprawić, ale dostarczyć musimy na czas") w niemal nieunikniony sposób prowadzi do błędów w projekcie i na innych etapach. Bardzo często po prostu brakuje czasu na wyczerpujące testy i usuwanie błędów.
- **Ograniczony zakres automatyzacji**. Rozwój i stosowanie oprogramowania to operacje w dużym stopniu bazujące na czynniku ludzkim. Narzędzia do automatyzacji, takie jak CASE i projektowanie obiektowe, są na pewno pomocne, ale zakres automatyzacji w oprogramowaniu jest ograniczony w porównaniu do wyrobów produkowanych.
- **Powiązanie z Internetem**. Coraz szersze zastosowania i integracja systemów oprogramowania z Internetem naraża je na zagrożenia wynikające z przypadku lub celowo szkodliwej działalności. Niebezpieczeństwo może być bardzo poważne: od kradzieży tożsamości, przez masowe oszustwa finansowe, po zagrożenie narodowego systemu bezpieczeństwa.
- **Abstrakcyjne działanie systemów cyfrowych**. Naturalnie abstrakcyjne działanie systemów cyfrowych sprawia, że trudno zapewnić jakość oprogramowania.
- **Brak odpowiednich zachęt**. Często bodźce rynkowe i regulacyjne w zakresie niezawodności oprogramowania są niewystarczające, podczas gdy istnieje wiele czynników promujących innowacyjne funkcje, wygodę stosowania i błyskawiczny cykl rozwoju.

Ograniczenia tradycyjnych systemów kontroli jakości

Praktyki zapewniania jakości zwykle koncentrują się na późnych operacjach, takich jak produkcja lub testy (zobacz rysunki 2.1 i 2.2). Takie podejście nie zawsze prowadzi do optymalnego projektu nawet w przypadku dużej liczby powtarzanych cykli projekt-prototyp-testy, które zasadniczo przebiegają przy użyciu metody prób i błędów. Ma to wpływ

na koszty, czas trwania cyklu i jakość produktu dostarczanego klientowi, jeśli udostępniane oprogramowanie nie ma odpowiednich właściwości w zakresie wydajności. Bardzo często dzieje się tak, ponieważ menedżer produktu nie ma czasu i środków, a musi udostępnić projekt w fazie produkcyjnej. Na dalszych etapach produkty są sprawdzane i przesiewane w celu wykrycia jednostek, które są niezgodne ze specyfikacją. Te jednostki są naprawiane, przetwarzane lub odrzucane. Takie systemy kontroli jakości bazują na dwóch podstawowych założeniach:

- Wymagania klientów są spełnione, jeśli produkt znajduje się w ramach uzgodnionych ograniczeń wyznaczanych przez specyfikację.
- Biznesowe skutki wydajności lub jakości produktów „ledwo spełniających wymagania specyfikacji" i „na poziomie docelowym" są takie same.

Jak się wkrótce okaże, żadne z tych założeń nie jest uzasadnione. W. Edwards Deming[7] był jednym z pierwszych analityków zarządzania, którzy zdali sobie sprawę z braków systemów jakości zależnych od kontroli. Jednak to japoński inżynier przemysłowy Genichi Taguchi jako pierwszy zaproponował alternatywny system zarządzania jakością, nazywany metodami Taguchiego. To podejście zwraca uwagę na wartość „poziomu docelowego" i potrzebę zarządzania jakością już na początku procesu — w dziale badań i rozwoju, na etapach projektu i inżynierii cyklu rozwoju oprogramowania — a nie polegania na kontroli mającej na celu wykrywanie i naprawianie błędów.

Japońskie systemy zarządzania jakością i podejście Taguchiego

Metody Taguchiego to zestaw zasad i metodologii projektowych mających na celu usprawnienie produktów i procesów. Te techniki składają się na dziedzinę wiedzy nazywaną inżynierią jakości, solidną inżynierią czy, zwłaszcza w Stanach Zjednoczonych, metodami Taguchiego od nazwiska autora tego podejścia, dr. Genichiego Taguchiego (zobacz ramki 2.1, 2.2 i 2.3).

Ramka 2.1. Życie i czasy doktora Genichiego Taguchiego[8, 9]

Doktor Genichi Taguchi miał znaczący wpływ na powstanie metodologii zarządzania jakością skoncentrowanej na projekcie. Taguchi urodził się w 1924 roku w Japonii. Po służbie w Wydziale Astronomicznym Instytutu Nawigacji Japońskiej Marynarki Wojennej w latach 1942 – 1945 pracował w Ministerstwie Zdrowia i Opieki oraz Instytucie Statystycznym Ministerstwa Edukacji. Zyskał bogatą wiedzę na temat technik projektowania eksperymentalnego i stosowania tablic ortogonalnych, ucząc się od nagradzanego japońskiego statystyka Matosaburo Masuyamy, z którym zetknął się w czasie pracy w Ministerstwie Zdrowia i Opieki. Doprowadziło to do jego zaangażowania jako konsultanta w firmie Morinaga Pharmaceuticals i jej organizacji nadrzędnej, Morinaga Seika.

W 1950 roku Taguchi dołączył do nowo powstałego Laboratorium Komunikacji Elektrycznej w Nippon Telephone and Telegraph Company z zadaniem zwiększenia wydajności działu badań i rozwoju poprzez szkolenie inżynierów w wydajnych technikach. Taguchi pracował tam przez ponad 12 lat i w tym czasie rozpoczął tworzenie swojej metodologii, którą dziś nazywa się metodami Taguchiego lub solidną inżynierią (zobacz ramka 2.2). Wtedy był także konsultantem japońskiego przemysłu. W wyniku tego japońskie firmy, między innymi Toyota i jej dostawcy, zaczęły, począwszy od wczesnych lat 50., szeroko stosować metody Taguchiego. Pierwsza książka Taguchiego, która przedstawiała tablice ortogonalne, została opublikowana w 1951 roku.

W latach 1954 – 55 Taguchi pracował jako profesor zaproszony w Indyjskim Instytucie Statystycznym w Kalkucie w Indiach. W czasie tej wizyty zetknął się ze sławnymi statystykami: Ronaldem A. Fisherem i Walterem A. Shewhartem. W latach 1957 – 58 Taguchi opublikował pierwszą wersję swej dwutomowej pracy *Design of Experiments*. Do Stanów Zjednoczonych przybył po raz pierwszy w 1962 roku jako zaproszony członek grupy badawczej na Uniwersytecie Princeton. W tym czasie odwiedził AT&T Bell Laboratories. W Princeton Taguchi był gościem poważanego statystyka Johna Tukeya, który ułatwił mu współpracę ze statystykami przemysłowymi z Bell Laboratories. W 1962 roku Taguchi otrzymał stopień doktora Uniwersytetu Kyushu.

W 1964 roku Taguchi został profesorem na tokijskim Uniwersytecie Aoyama Gakuin, którą to funkcję piastował do roku 1982. W 1966 Taguchi wraz z kilkoma innymi autorami napisał książkę *Management by Total Results*, która została przetłumaczona na język chiński przez Yuin Wu, współpracownika Taguchiego przy wielu późniejszych pracach. Na tym etapie metody Taguchiego były praktycznie nieznane na Zachodzie, choć stosowano je w Tajwanie i Indiach. W tym czasie i w latach 70. większość zastosowań metod Taguchiego dotyczyła procesów produkcji. Przejście do projektowania produktów miało miejsce później. We wczesnych latach 70. Taguchi rozwinął zagadnienie funkcji utraty jakości. Wtedy też opublikował dwie dalsze książki oraz trzecie wydanie pracy *Design for Experiments*. Taguchi został laureatem nagrody Deming Application Prize w 1960 roku oraz nagród Deminga za pozycje książkowe w latach 1951, 1953 i 1984, a do późnych lat 70. zyskał szerokie uznanie w Japonii.

W 1980 roku został zaproszony do Stanów Zjednoczonych, gdzie ponownie odwiedził AT&T Bell Laboratories. Udało mu się, mimo problemów z komunikacją, przeprowadzić udane eksperymenty, które pozwoliły zastosować metody Taguchiego w korporacji Bell Laboratories. Po wizycie Taguchiego w Stanach Zjednoczonych w 1980 roku coraz więcej amerykańskich fabryk zaczęło stosować jego metodologię. Mimo niechętnego przyjęcia tych metod przez grono amerykańskich statystyków, co prawdopodobnie wynikało ze sposobu ich rozpowszechniania, największe korporacje Stanów Zjednoczonych zaczęły stosować metodologię Taguchiego.

W 1982 roku Taguchi został doradcą japońskiej Organizacji do spraw Standardów. Za wkład w rozwój przemysłu na całym świecie Taguchi otrzymał wiele wyróżnień:

- trzykrotnie nagrodę Deming Prize za wkład w dziedzinę inżynierii jakości;
- medal Willarda F. Rockwella za połączenie inżynierii i metod statystycznych do osiągnięcia błyskawicznych korzyści w zakresie kosztów i jakości poprzez optymalizację procesu projektowania i produkcji wyrobów;

- medal imienia Shewharta Amerykańskiego Stowarzyszenia na rzecz Kontroli Jakości;
- Niebieską Wstęgę cesarza Japonii, przyznaną w 1990 roku za wkład w rozwój przemysłu;
- honorowe członkostwo w Amerykańskim Stowarzyszeniu na rzecz Kontroli Jakości.

Znalazł się też w motoryzacyjnej galerii sław oraz na światowym poziomie galerii sław z dziedziny inżynierii, nauki i technologii.

Ramka 2.2. Metodologia inżynierii jakości w zarysie[8, 9]

Metodologia Taguchiego dotyczy optymalizacji produktu i procesu na etapach projektowania oraz prac działu badań i rozwoju przed rozpoczęciem produkcji. Jest to metodologia zarządzania jakością stosowana we wczesnych fazach rozwoju produktu lub procesu, która w mniejszym stopniu koncentruje się na osiąganiu jakości poprzez kontrolę. Jest to wydajna technika, obejmująca testy projektu przed rozpoczęciem etapu produkcji, fabrykacji lub składania. Dzięki temu jakość staje się funkcją prawidłowego projektu, a nie nawet ścisłych testów i kontroli. Podejścia Taguchiego można używać także jako metodologii rozwiązywania problemów związanych z produkcją i procesami w obszarze fabrykowania. Jest też coraz częściej stosowane w innych dziedzinach przemysłu, na przykład przy rozwoju oprogramowania.

W odróżnieniu od zachodniego podejścia do jakości, w metodologii Taguchiego jakość jest postrzegana raczej w kategoriach **utraty jakości** niż samej jakości. **Utrata jakości** jest definiowana jako „straty powodowane przez produkt w społeczności od momentu jego udostępnienia". Ta utrata obejmuje straty ponoszone przez firmę w postaci kosztów przetwarzania i utylizacji, wydatki na konserwację oraz przestoje wynikające z awarii sprzętu i napraw gwarancyjnych. Należy uwzględnić także koszty klientów powodowane przez zawodność oraz niską wydajność produktu, prowadzące do dalszych strat po stronie producenta włącznie ze spadkiem jego udziałów w rynku. Określając docelowy poziom jakości jako najwyższy możliwy, Taguchi wiąże prostą kwadratową funkcję straty z odchyleniem od poziomu docelowego. Funkcja utraty jakości pokazuje, że zmniejszanie zmienności w zakresie jakości prowadzi do zmniejszania strat i związanej z tym poprawy jakości.

Gdy uwzględnimy takie podejście do utraty jakości, to strata wystąpi nawet w przypadku, kiedy produkt spełnia wymagania stawiane przez specyfikację, a jest minimalna, jeśli producent dąży do poziomu docelowego. W praktyce dla użytkowników nie jest ważna specyfikacja, prawda? Klienci chcą, aby produkt działał nawet wtedy, kiedy napięcie prądu jest niskie, droga wyboista, a operator terminala nieprzeszkolony. Produkt musi działać na docelowym poziomie w różnych warunkach. Mówiąc inaczej, należy **projektować z uwzględnieniem różnych warunków stosowania produktu przez użytkowników**. To w interesie producenta leży utrzymywanie wydajności produktu lub procesu tak blisko poziomu docelowego, jak to ekonomicznie możliwe. Funkcja straty może posłużyć do podjęcia decyzji projektowych w kategoriach finansowych. Pomaga zdecydować, czy dodatkowe koszty zwiększenia odporności i usprawnienia produkcji są usprawiedliwione oraz czy okażą się zasadne w warunkach rynkowych.

Metod Taguchiego można używać w trybie „offline" w czasie projektowania lub na bieżąco w produkcji.

Taguchi dzieli kontrolę jakości w trybie „offline" na trzy etapy.

1. **Projektowanie systemu** obejmuje tworzenie pomysłu na projekt przy użyciu burzy mózgów, badań lub innych technik. Projektowanie systemu jako całości obejmuje także inne narzędzia, techniki i metodologie, zwłaszcza AHP (ang. *Analytic Hierarchy Process*), QFD (ang. *Quality Function Deployment*), TRIZ (ang. *Theory of Inventive Problem Solving*) i FMEA (ang. *Failure Modes and Effects Analysis*), opisane odpowiednio w rozdziałach 8., 11., 12. i 13.
2. **Projektowanie parametrów** to istota metod Taguchiego. To pod tym względem Japończycy tradycyjnie się wyróżniali i osiągali wysokie poziomy jakości bez wzrostu kosztów, co jest główną przyczyną ich przewagi konkurencyjnej. Testowane są nominalne właściwości projektu lub wybrane poziomy czynników procesu. Ustalane są kombinacje poziomów parametrów produktu lub poziomów operacji wchodzących w skład procesu, które okazały się najmniej wrażliwe na zmiany w czynnikach środowiskowych i inne niekontrolowalne elementy (szum). To zagadnienie opisujemy w rozdziałach 16. i 17.
3. **Projektowanie odporności** należy zastosować do projektu produktu lub procesu, jeśli zmniejszenie wariancji uzyskane na etapie projektowania parametrów jest niewystarczające. Ta faza dodatkowo zwiększa odporność na czynniki mające duży wpływ na wariancję. Należy zastosować funkcję straty i poświęcić więcej środków tylko wtedy, **jeśli jest to konieczne**. Można zwiększyć odporność albo kupić lepsze materiały lub wyposażenie, jeśli jest to niezbędne, co podkreśla japońską filozofię inwestycji na końcu, a nie na początku, jak jest to praktykowane w firmach zachodnich.

Ramka 2.3. Taguchi o metodach Taguchiego[10]

- Utrata jakości wynika głównie z awarii produktu po jego sprzedaży. „Solidność" wyrobu jest bardziej funkcją jego projektu niż bieżącej kontroli, choćby najściślejszej, procesu produkcji.
- Projektuj produkty tak, aby nie zawodziły w praktyce. Tym samym zmniejszysz liczbę defektów w fabryce.
- Udostępniając produkt, który ledwie spełnia standardy korporacji, producent nie zyskuje prawie nic w porównaniu z rozpowszechnianiem wadliwego wyrobu. Należy dążyć do poziomu docelowego, a nie jedynie mieścić się w ramach specyfikacji.
- Inżynieria jakości (metody Taguchiego) to technologia przewidywania błędów jakościowych i zapobiegania im na wczesnych etapach rozwoju i projektowania produktu, włącznie z problemami związanymi z funkcjami produktu, zanieczyszczeniem środowiska i innymi kosztami, które pojawiają się w późnych fazach produkcji oraz po wypuszczeniu wyrobu na rynek.
- Nie używaj miar jakości związanych z klientem (takich jak procent defektów czy niezawodność) jako wczesnych miar jakości w dziale badań i rozwoju. W zamian

używaj dynamicznego stosunku SN jako wskaźnika wydajności do oceny solidności funkcji produktu.

- Solidne produkty zapewniają silny „sygnał" niezależnie od zewnętrznego „szumu" i z minimalną ilością „szumów" wewnętrznych. Usprawnienie projektu, czyli znaczący wzrost w stosunku sygnału do szumu komponentu, zwiększy solidność produktu jako całości.
- Należy wytrwale pracować w celu uzyskania projektów, które można produkować na nieustannie wysokim poziomie. Należy stale stawiać wysokie wymagania fabryce.
- Jest większe prawdopodobieństwo problemów z powodu dużej zmienności w obrębie specyfikacji niż stałego odchylenia poza nią. Jeśli odstępstwo od poziomu docelowego jest stałe, możliwe jest dostosowanie procesu do tego poziomu.
- Warunki panujące w fabryce rzadko są tak szkodliwe, jak zmienność w użytkowaniu produktów przez użytkowników.

Metody Taguchiego zostały uznane za jedno z najważniejszych inżynieryjnych osiągnięć XX wieku. Choć techniki statystyczne używane przez Taguchiego mają początki w eksperymentalnych praktykach projektowych rozwiniętych przez angielskiego statystyka sir Ronalda Fishera, ich filozoficzne podstawy są niezaprzeczalnie japońskie. Metody Taguchiego i inne japońskie systemy zarządzania jakością, takie jak *kaizen* (ciągłe usprawnianie), *kanban* (dokładnie na czas), **zarządzanie przez jakość** czy **szczupła produkcja**, zostały zainspirowane rozważaniami amerykańskiego guru z obszaru jakości, W. Edwardsa Deminga, przedstawionymi w jego książkach: *14 Points of Management*, *Seven Deadly Diseases* i *Obstacles to Quality Products*. Proponowana przez Richarda Zultnera adaptacja zasad Deminga do rozwoju oprogramowania jest opisana w rozdziale 5. Metody Taguchiego i inne systemy zarządzania jakością położyły podwaliny pod niezwykły rozwój Japonii jako potęgi przemysłowej po II wojnie światowej.

Trudno przecenić wpływ Deminga na prace Taguchiego. Podobnie jak inni japońscy specjaliści do spraw jakości, Taguchi był pod dużym wpływem Deminga. Aby zrozumieć specyficzne podejście Taguchiego, należy przeanalizować jego kontekst i korzenie. Metody Taguchiego i współczesne japońskie systemy zarządzania jakością miały swe początki po II wojnie światowej. Deming, którego często określa się mianem ojca współczesnego ruchu na rzecz jakości, po raz pierwszy odwiedził Japonię w 1946 roku. W następnych dziesięcioleciach kontynuował współpracę z rządem oraz przemysłem japońskim i wyszkolił tysiące japońskich menedżerów oraz inżynierów. Ci menedżerowie byli zainteresowani współczesnymi amerykańskimi zasadami zarządzania. Jednak Deming zaproponował im coś zupełnie nowego, co, jego zdaniem, miało pomóc w przekształceniu Japonii w dobrze prosperujące społeczeństwo i odbudować jej pozycję jako znaczącej potęgi przemysłowej.

Istota zasad zarządzania Deminga jest dobrze znana (zobacz ramkę 2.4), choć nie są one zbyt szeroko stosowane poza Japonią. Te zasady obejmują **głos klienta**, **zmniejszanie wariancji**, **stosowanie wskaźników statystycznych**, **zyskiwanie zaufania i szacunku**

Ramka 2.4. Istota 14 punktów Deminga[11]

1. Bądź wierny zamiarom usprawniania produktów i usług. Cel to osiągnięcie konkurencyjności, pozostanie w grze i zapewnienie miejsc pracy.
2. Zarząd musi zdać sobie sprawę z wyzwań związanych z jakością, poznać zakres odpowiedzialności i przejąć przywództwo na drodze zmian.
3. Należy przestać uważać kontrolę za najważniejszy element osiągania jakości. Trzeba wyeliminować potrzebę masowych inspekcji poprzez wbudowanie jakości w produkt.
4. Trzeba skończyć z nagradzaniem działań na podstawie ceny. W zamian należy minimalizować koszty łączne. Każdy produkt powinien być dostarczany przez tylko jednego dostawcę, co tworzy długotrwały związek bazujący na lojalności i zaufaniu.
5. Należy trwale i nieustannie usprawniać system produkcji i usług, aby poprawić jakość oraz wydajność, a tym samym ciągle zmniejszać koszty.
6. Trzeba prowadzić szkolenia zawodowe. Jeśli ludzie są nieodpowiednio wyszkoleni, nie będą pracować w taki sam sposób, a to wprowadza zmienność.
7. Należy wdrożyć system przywództwa. Deming wprowadza rozróżnienie między przywództwem i nadzorem. Ten drugi bazuje na normach i poziomach. Celem nadzoru powinno być pomaganie ludziom, maszynom i urządzeniom w lepszym wykonywaniu zadań. Nadzór zarządu wymaga starannego przemyślenia, podobnie jak nadzór pracowników odpowiedzialnych za produkcję.
8. Trzeba pozbyć się lęku, aby każdy mógł wydajnie pracować dla firmy.
9. Należy znieść podziały między wydziałami. Osoby odpowiedzialne za badania, projektowanie, sprzedaż i produkcję muszą pracować jako zespół, aby przewidywać problemy z wytwarzaniem i stosowaniem produktów lub usług.
10. Trzeba pozbyć się sloganów, napomnień i norm dla siły roboczej, które dotyczą braku defektów i nowych poziomów produktywności. Takie napomnienia prowadzą wyłącznie do powstawania antagonizmów. Większość przyczyn niskiej jakości i wydajności leży po stronie systemu i tym samym znajduje się poza kontrolą siły roboczej.

11a. Należy wyeliminować standardy pracy (normy) dla fabryki. Trzeba zastąpić je przywództwem.

11b. Trzeba wyeliminować zarządzanie przez zadania oraz liczby (z celami numerycznymi) i zastąpić je przywództwem.

12a. Należy usunąć przeszkody, które pozbawiają pracowników zatrudnianych na godziny prawa do dumy z pracy. Pracownicy nadzoru powinni odpowiadać nie za same liczby, ale za jakość.

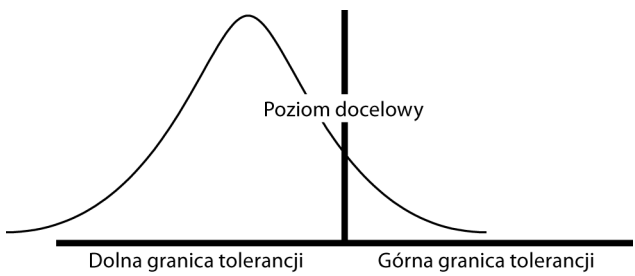
12b. Trzeba usunąć bariery, które pozbawiają zarząd i inżynierów prawa do dumy z pracy. Oznacza to między innymi rezygnację z dorocznych rankingów wydajności i zarządzania przez cele.

13. Należy wprowadzić dynamiczny program edukacji i samodoskonalenia.
14. Trzeba zachęcić wszystkie osoby w firmie do pracy nad przekształceniami. Transformacja to zadanie dla wszystkich.

współpracowników oraz **ciągłe usprawnianie** w obszarze procesów, a także produktów i usług. W Japonii podejście Deminga było z entuzjazmem studiowane i stosowane oraz miało znaczący wpływ na przemysł tego kraju. W 1951 roku Japońskie Stowarzyszenie Naukowców i Inżynierów (ang. *Japanese Union of Scientists and Engineers* — JUSE) uhonorowało Deminga, nazywając jego imieniem prestiżową nagrodę z dziedziny jakości. Jednak w Stanach Zjednoczonych teorie Deminga były ignorowane przez prawie 30 lat. Uważa się, że mogło to doprowadzić do utraty konkurencyjności wielu gałęzi amerykańskiego przemysłu, takich jak motoryzacja i AGD, w których japońskie korporacje poczyniły ogromne postępy.

Wiele prac Taguchiego było inspirowanych 14 punktami zarządzania Deminga. W szczególnym stopniu dotyczy to zasady **braku zależności od inspekcji przy osiąganiu jakości**. W procesie rozwoju produktu Taguchi cofnął się jeszcze o krok, kładąc nacisk na inspekcję w dziale badań i rozwoju oraz na etapie projektowania i inżynierii. Zwracał uwagę na znaczenie tego, aby produkt działał na stałym, docelowym poziomie, a nie był tylko ledwie zgodny ze specyfikacją. Na rysunkach 2.3, 2.4 i 2.5 zademonstrowaliśmy, że można to osiągnąć w dwóch krokach. Najpierw należy zmniejszyć wariancję, a następnie dostosować odpowiedni czynnik, aby osiągnąć wyniki możliwie jak najbliższe docelowym wymaganiom klienta, trzeba też wziąć pod uwagę koszty, projekt i inne ograniczenia.

RYSUNEK 2.3.
Rozkład wydajności w granicach specyfikacji, ale niestałej i poniżej poziomu docelowego



RYSUNEK 2.4.
Rozkład wydajności stałej, ale poniżej poziomu docelowego

RYSUNEK 2.5.
Rozkład wydajności stałej i w pobliżu poziomu docelowego

Ponadto Taguchi podkreślał wartość tolerancji projektu zarówno w środowisku produkcyjnym, jak i środowisku użytkownika. W tym momencie warto przedstawić główne nakazy filozofii jakości Taguchiego. Oto one.

1. **Ciągła poprawa jakości** i **redukcja kosztów** są niezbędne dla przetrwania biznesu.
2. Ważną miarą jakości produktu jest strata ogólna spowodowana przez produkt w społeczeństwie obliczana za pomocą **funkcji straty jakości**.
3. Zmiana przedprodukcyjnych procedur eksperymentalnych z różnicowania jednego czynnika na raz na **modyfikowanie wielu czynników jednocześnie**. Jest to tak zwane statystyczne projektowanie eksperymentów (ang. *Statistical Design of Experiments* — SDE) lub po prostu projektowanie eksperymentów (ang. *Design of Experiments* — DOE). Dlatego jakość można **wbudować** w produkt i proces.
4. Straty klienta wynikające z niskiej jakości są mniej więcej proporcjonalne do kwadratu odchylenia cech wydajności od poziomu docelowego (wartości nominalnej). Taguchi zmienił cele eksperymentów i definicję jakości z **osiągania zgodności ze specyfikacją** na **dążenie do poziomu docelowego i minimalizację zmienności**.
5. Zmienność wydajności produktu (lub usługi) można zmniejszyć, sprawdzając nieliniowy wpływ **czynników** (parametrów) **kontrolnych** na cechy wydajności. Wszelkie odchylenia od poziomu docelowego prowadzą do niskiej jakości.

Jednym z głównych celów Taguchiego jest usprawnienie projektu produktu i procesu poprzez wykrycie kontrolowalnych czynników i ich wartości, co minimalizuje zmienność w stosunku do wyniku docelowego. Ustawiając kontrolowalne parametry na ich optymalny poziom, można sprawić, że produkt będzie bardziej odporny na zmiany w działaniu, stosowaniu i warunkach środowiskowych. Podstawowa zasada metod Taguchiego dotyczy raczej usuwania negatywnych efektów przyczyn niż powodów tych niekorzystnych skutków. Dzięki temu można uzyskać produkt wyższej jakości najmniejszym możliwym kosztem. Ta strategia neutralizacji samych skutków, a nie przyczyn, jest mądrym rozwiązaniem, ponieważ może być łatwiejsza, a także bardziej wydajna ze względu na koszty i szybsza. Metody Taguchiego mają dwa kluczowe cele projektowe:

- zmniejszenie i minimalizację zmienności produktu i ekonomiczne osiągnięcie poziomu docelowego,
- zapewnienie tolerancji mierzonej na etapach tworzenia projektu i prototypu, które jest przenoszone na dalsze fazy w produkcji i środowisku użytkownika.

Istota metod solidnego projektowania Taguchiego

Solidność to kluczowy koncept metod Taguchiego. „Solidność" oznacza zdrowie, moc, energię i siłę. Taguchi definiuje ją jako „stan, w którym działanie technologii, produktu lub procesu jest w minimalnym stopniu narażone na czynniki powodujące zmienność (zarówno w produkcji, jak i środowisku użytkownika) przy najniższym koszcie wyprodukowania jednostki". Jest to zdolność produktu lub procesu do funkcjonowania i spełniania wymagań klientów (ze względu na niezawodność, bezpieczeństwo, zabezpieczenia i tak dalej), mimo obecności różnych czynników zakłócających, które mogą powodować zmienność. Solidny proces lub produkt działa w oczekiwany sposób niezależnie od wszelkich szkodliwych wpływów, zwanych szumem. Szum wynika z wszelkiego rodzaju zmienności: środowiskowej wariancji w trakcie stosowania produktu przez klienta, zmienności w czasie produkcji poszczególnych jednostek i komponentów oraz zróżnicowania komponentów w wyniku starzenia się i pogarszania cech.

Zagadnienie stosunku sygnału do szumu

Według Taguchiego na solidność trzeba zwrócić uwagę na etapie projektowania lub w dziale badań i rozwoju. Wiąże się to ze specyficznym filozoficznym założeniem, że prawdziwą solidność można jedynie zaprojektować i wbudować w produkt lub projekt, a nie skontrolować i naprawić. Mówiąc inaczej, podstawowym hasłem jest „zapobiegać, a nie leczyć". Wymaga to także rozwiązywania problemów na początku pracy, w dziale badań i rozwoju, w trakcie zaawansowanej inżynierii i projektowania, a nie w trakcie produkcji i kontroli lub, co gorsza, po dostarczeniu produktu do klienta. Metody Taguchiego zapewniają wydajną ze względu na koszty i czas metodologię projektowania oraz testowania produktów pod kątem solidności przed rozpoczęciem ich produkcji. Metody te służą także do rozwiązywania problemów różnego rodzaju czy modyfikowania projektów wadliwych procesów.

Poniżej znajdują się definicje niektórych podstawowych pojęć używanych w metodach Taguchiego.

Sygnał to coś, co produkt (lub część albo podzespół) ma dostarczać, zgodnie z jego charakterystyką działania lub funkcjonalną. W odbiorniku telewizyjnym lub telefonie sygnały to coś, co produkt (odbiornik lub aparat) przekazuje jako obraz lub dźwięk. Dobry sygnał to taki, który zachowuje jakość mimo szumu (wewnętrznych i zewnętrznych interferencji elektromagnetycznych w odbiorniku telewizyjnym lub telefonie).

Szum to wszystkie czynniki, które powodują wariancję. Taguchi stwierdził, że wielu czynników powodujących szum, takich jak sposób stosowania przez użytkownika (najczęstsza przyczyna zmienności), warunki na drodze czy pogoda, nie można kontrolować lub wyeliminować. To szum powoduje odchylenia charakterystyk działania i funkcjonalnych od docelowej jakości. Można wyróżnić trzy typy czynników związanych z szumem:

- **Szum zewnętrzny**, nazywany także **szumem środowiskowym**, obejmuje czynniki zewnętrzne (środowiskowe), takie jak interferencje cieplne lub elektromechaniczne, szoki mechaniczne i elektryczne, kurz czy nieprawidłowe korzystanie ze sprzętu przez użytkownika. W przypadku samochodu te czynniki obejmują temperaturę, śnieg, drogę, warunki jazdy, kurz i tak dalej.
- **Szum wewnętrzny**, zwany także wariancją komponentów lub wewnętrznym pogarszaniem sprawności części, wynika ze zużywania się i starzenia.
- **Szum między produktami**, nazywany także **wariancją produkcyjną** lub **wariancją elementów**, dotyczy zmienności obecnej między poszczególnymi produktami, mimo że są tworzone według tych samych specyfikacji. Może to wynikać ze zmienności w zakresie materiałów i procesów.

Szumy powodują, że produkty działają niezgodnie ze specyfikacją lub całkowicie zawodzą. Działanie produktu jest zdominowane przez szum jednostkowy (wewnętrzny) na wczesnych etapach cyklu życia produktu, zewnętrzny w trakcie stosowania i pogarszanie sprawności (między produktami) pod koniec życia. Solidność i solidne projektowanie prowadzą do braku wrażliwości na czynniki powodujące szum na różnych etapach cyklu życia produktu, choć wpływów tych nie można wyeliminować i usuwane są jedynie ich efekty. Dla odbiorników telewizyjnych powszechnym szumem jest „śnieżenie" ekranu, pioruny, sztormy, wahania napięcia i inne niekorzystne warunki działania.

W przypadku oprogramowania kluczowe czynniki związane z szumem, które powodują zmienność, to nieprawidłowe korzystanie z aplikacji przez klientów, napastnicy i hakerzy, robaki i wirusy, przypadkowe lub celowo złośliwe naruszenie zabezpieczeń, brak dokumentacji, nieodpowiednie szkolenia, awarie procedur, niedozwolony dostęp i używanie oraz korzystanie z systemu do wykonywania zadań, do których nie jest przeznaczony.

Taguchi sugeruje, że jako miary jakości należy używać stosunku **sygnału do szumu**, **SN** (ang. *signal-to-noise*)[12]. Taguchi twierdzi, że stosunek SN:

- może służyć do oceny solidności funkcji produktu, ponieważ reprezentuje funkcjonalność;
- reprezentuje stosunek tolerancji (lub „średnią" w terminologii statycznej) do zmienności;
- kiedy stosuje się go do oceny solidności produktu lub procesu, należy go określać mianem przetwarzalności (w energię, moc, informację, obraz, datę i tak dalej).

Przykładowo dla urządzenia elektromechanicznego, takiego jak silnik elektryczny, stosunek SN można wyrazić w następujący sposób:

$$S/N = \frac{\text{Przekształcenie energii elektrycznej w celu obsługi wymaganych funkcji mechanicznych}}{\text{Przekształcenie energii w wyniku niepożądanych nieekonomicznych funkcji}}$$

$$= \frac{\text{Przekształcenie energii na użyteczny moment obrotowy}}{\text{Przekształcenie energii w wyniku marnotrawstwa ciepła, iskrzenia, wibracji, zgrzytów itd.}}$$

Ogólnie stosunek SN w systemach bazujących na inżynierii, takich jak silniki czy generatory, można opisać w poniższy sposób:

$$S/N = \frac{\text{Przekształcenie energii w pożądane funkcje wyjściowe}}{\text{Przekształcenie energii w niepożądanych funkcje wyjściowe}}$$

W przypadku oprogramowania jest to przekształcanie informacji, danych, obrazów i innych elementów, a nie energii czy mocy. Solidne projektowanie zarówno w dziedzinie oprogramowania, jak i sprzętu, ma maksymalizować stosunek SN pod kątem optymalizacji projektu.

Zagadnienie funkcji utraty jakości

Jak już wspomnieliśmy, to zagadnienie jest podstawowym odstępstwem od zachodniego podejścia do jakości. Taguchi zajmuje się bardziej utratą jakości niż nią samą, a także skutkami takich strat dla klienta, producenta i społeczności jako ogółu. Jasno wynika z tego, że jakość produktu to coś więcej niż tylko niezawodność, a koszty produktu obejmują nie tylko cenę produkcji i rachunki za materiały. Klienci oczekują niezawodności w czasie trwania cyklu życia produktu, a w ostatecznym rozrachunku istotna jest optymalizacja kosztów tego cyklu. Klienci coraz bardziej domagają się bezbłędnych dostaw i działania. Oczekują różnych dodatkowych funkcji i udogodnień. Coraz częściej też poszukują stabilnego, wysokiej jakości działania na poziomie docelowym i nie zadowalają się niestabilnym funkcjonowaniem w ramach specyfikacji. Zapewnienie działania na poziomie docelowym może wymagać poniesienia znaczących kosztów, ale też zapewnia korzyści zarówno producentowi, jak i klientom. Jest to jedna z podstawowych zasad metodologii Taguchiego.

Fowlkes i Creveling klasyfikują koszty cyklu życia według następujących kategorii[13]:

- koszty dóbr, które obejmują faktury za materiały oraz wydatki na produkcję;
- koszty powiązane z obsługą konserwacji, gwarancjami, naprawami, wymianami, utylizacją i odzyskiwaniem;
- koszty klienta, które obejmują przestoje, koszty operacyjne i wynikające z rozwiązywania błędów;
- koszty marketingu, które obejmują czas wypuszczania produktu na rynek, promocje oraz zatrzymywanie klientów i zdobywanie nowych.

Utrata jakości jest definiowana jako „strata, którą produkt powoduje w społeczności od czasu jego wprowadzenia". Obejmuje to straty firmy związane z kosztami przeróbek, utylizacji, konserwacji i przestojów wynikających z awarii sprzętu, a także z roszczeniami gwarancyjnymi. Są to również koszty ponoszone przez klienta w wyniku zawodności i słabej wydajności produktu, co prowadzi do dalszych strat producenta wraz z utratą udziałów w rynku. Mogą pojawić się też straty w społeczności, jeśli dane produkty lub usługi wiążą się ze składowaniem odpadów, zanieczyszczeniem środowiska, przestępczością czy zagrożeniem bezpieczeństwa i zdrowia.

Określając wartość docelową cech jakościowych na najwyższym możliwym poziomie, Taguchi wiąże prostą kwadratową funkcję straty z odstępstwami od wartości docelowej. Funkcja utraty jakości pokazuje, że redukcja zmienności w okolicach poziomu docelowego prowadzi do zmniejszania się strat, z czego wynika wzrost jakości. Ta funkcja służy do podejmowania decyzji projektowych na podstawie czynników finansowych i pozwala określić, czy dodatkowe koszty, jakich wymaga wyższa jakość, okażą się warte poniesienia z perspektywy rynku. Z punktu widzenia producenta łączne koszty można wyrazić w następujący sposób:

Łączne koszty wynikające z rezygnacji z jakości = straty produkcyjne+utrata jakości

Straty produkcyjne obejmują utylizację, przeróbki, opóźnienia i tak dalej. Utrata jakości to koszt awarii produktów, który powstaje po ich udostępnieniu. Obejmuje to straty w wyniku zwrotów, gwarancji i napraw, a także utraty dobrej opinii, co prowadzi do zmniejszenia udziałów w rynku. Utrata Jakości może być bardzo duża nawet wtedy, kiedy produkt jest zgodny ze specyfikacją. Ta wartość jest równa zeru tylko wtedy, kiedy produkt działa dokładnie na poziomie docelowym.

Taguchi proponuje przybliżony i łatwy wzór na funkcję utraty jakości (ang. *quality loss function* — QLF):

Utrata = O^2K, gdzie

O = odstępstwo od poziomu docelowego;

K = koszty środków niezbędnych do zapewnienia działania produktu na poziomie docelowym.

To wyrażenie jest przybliżeniem, a nie „prawem natury"[10]. Jest to narzędzie dla inżynierów, a nie prawo naukowe. Wskazuje jedynie na fakt, że występuje „prawo rosnących kosztów" wraz z odchylaniem się działania produktu od poziomu docelowego. Trudno utworzyć ogólny i dokładny model kosztów. Jedną z przyczyn tej trudności jest to, że produkt może mieć różnych użytkowników i być używany w zmienny sposób w odmiennych warunkach środowiskowych. Taguchi sugeruje, że firmy mające lepsze sposoby szacowania kosztów powinny używać właśnie ich. Jednak przedstawiona wcześniej wersja przybliżenia funkcji QLF jest wartościowa z następujących względów.

- Zapewnia inżynierom i menedżerom prosty sposób szacowania kosztów odchyleń od poziomu docelowego.
- Pozwala określić na podstawie takich szacunków poziom docelowy tolerancji i jakości.
- Stanowi wydajny pod względem czasu i kosztów proces szacowania optymalnego projektu. Inaczej mówiąc, proces projektowania lepszego produktu jest tańszy i szybszy niż w przypadku tradycyjnego projektowania eksperymentów czy innych metod.

QLF i stosunek SN to dwie miary jakości w metodach Taguchiego. Oba te wskaźniki kładą nacisk na działania początkowe (projektowe), a przy ocenie jakości bazują na miarach związanych z klientem, takich jak koszty w dolarach i cechy działania (funkcjonalne). Jak piszemy w rozdziałach 16. i 17., wskaźniki te są powiązane ze sobą i stanowią wartościowe miary w metodologiach Taguchiego.

Zagadnienie solidnego projektowania

Taguchi zaleca następującą strategię solidnego projektowania:

- Stosowanie **tablic ortogonalnych** do przeprowadzania eksperymentów **na sucho**. Tablica ortogonalna to macierz, która jest integralną częścią metod Taguchiego. Analiza produktu lub procesu pod kątem solidności obejmuje identyfikację czynników zakłócających, które powodują odchylenia. Może to być żmudne i kosztowne zadanie. Taguchi zaprojektował tablice ortogonalne w celu wyizolowania czynników zakłócających spośród innych w sposób wydajny ze względu na czas i koszty. Zastosowanie tej techniki do rozwoju oprogramowania jest opisane w rozdziale 17.
- **Maksymalizacja miar wydajności i stosunku SN** w celu optymalizacji projektu z uwzględnieniem czynników kontrolnych.

Solidne projektowanie za pomocą metod Taguchiego przebiega w trzech następujących etapach:

- **Projektowanie systemu**, związane z rozwojem zarysu lub prototypu projektu. Jest to niezbędne w celu zdefiniowania wyjściowych cech produktu lub procesu. Ta faza w swej istocie przypomina projektowanie systemów stosowane na zachodzie.
- **Projektowanie parametrów**. Jest to kluczowy etap solidnego projektowania, w którym Japończycy wyróżniają się, tworząc solidne produkty niższym kosztem. Jest to metodologia redukująca zmienność poprzez zmniejszenie wrażliwości produktu lub procesu w zakresie wydajności na źródła wariancji, a nie poprzez próbę kontroli lub eliminacji tych czynników. Na tym etapie odbywają się testy nominalnych funkcji projektu lub wybranych poziomów czynników procesu oraz połączenia poziomów parametrów produktu lub poziomów operacyjnych procesu,

które są najmniej wrażliwe na zmiany w warunkach środowiskowych i inne niekontrolowalne czynniki (szum). Jest to istota metod Taguchiego w zakresie solidnego projektowania.

- **Projektowanie tolerancji,** które, jeśli to konieczne, służy dalszemu zmniejszaniu zmienności poprzez zwiększenie odporności na czynniki mające duży wpływ na wariancję. Na tym etapie stosowana jest funkcja utraty jakości w celu sprawdzenia, czy koszt poprawy jakości jest uzasadniony ekonomicznie. Inwestycje w zwiększenie tolerancji poprzez zastosowanie lepszych materiałów i sprzętu należy wprowadzać wtedy, kiedy wymaga tego rynek, a nie jako wyjściowe podejście.

W rozdziałach 16. i 17. opisujemy metody Taguchiego oraz sposób ich przystosowania do rozwoju oprogramowania. Ich zastosowanie na początkowych etapach rozwoju oprogramowania jest przedstawione w rozdziałach 18. i 19.

Wyzwania na drodze do niezawodności oprogramowania — projektowanie oprogramowania godnego zaufania

Oprogramowanie to najbardziej zdradliwy komponent każdego systemu informatycznego. Dwa pozostałe elementy, sprzęt i sieci komunikacyjne, uzyskały przez ostatnie 50 lat dużo wyższy poziom wydajności i niezawodności. Na przykład wydajność mikroprocesorów wzrosła w stopniu o około 200 milionów razy wyższym niż wydajność oprogramowania. Współczesne sieci komunikacyjne umożliwiają przenoszenie olbrzymich ilości danych, obrazów i dźwięku oraz dostęp do nich zarówno w obrębie organizacji, jak i globalnie. Współczesne sieci komunikacyjne, a zwłaszcza Internet, wiążą się z groźbą przypadkowego lub celowego nieupoważnionego dostępu oraz są podatne na inne zagrożenia, jednak to braki projektowe w oprogramowaniu odpowiadają za największą część wrażliwości i zawodności systemów informacyjnych. Nawet mimo niezwykłego poziomu wydajności sprzętu, ostateczne wyniki działania każdego systemu informatycznego zależą od niezawodności oprogramowania — i to jest tematem niniejszej książki.

W tabeli 2.2 opisujemy pewne często stosowane definicje i atrybuty jakości oprogramowania. Miary oprogramowania są omówione dość szczegółowo w rozdziale 3., jednak w tym miejscu warto omówić kilka podstawowych zagadnień. Najbardziej fundamentalne z nich to pojęcie samej jakości. Jakość oprogramowania można zdefiniować jako stopień spełniania wymagań lub oczekiwań klientów albo użytkowników przez system, komponent lub proces. Może to obejmować kilka elementów, z których najczęściej wymieniana jest wiarygodność oprogramowania. Związana jest ona z różnymi wymaganiami użytkownika, włączając w to niezawodność, bezpieczeństwo, zabezpieczenia i dostępność[14]. Jest to bliskie używanemu w tej książce pojęciu oprogramowania godnego zaufania, które jednak obejmuje dodatkowo możliwość zdobywania **zaufania klientów** i **spełnianie ich jawnych,**

TABELA 2.2.
Wybrane atrybuty związane z jakością oprogramowania

Jakość oraz atrybuty i systemy jakości	Opis
Jakość	Stopień, w jakim systemy, komponenty lub procesy spełniają, po pierwsze, jawne, niejawne i nieoczekiwane potrzeby lub oczekiwania klientów i użytkowników oraz, po drugie, określone i ukryte wymagania innych partnerów.
Projektowanie pod kątem Six Sigma	System projektowania i rozwoju nowych produktów, procesów i usług, które spełniają wymagania klientów i są pozbawione defektów.
Projektowanie oprogramowania godnego zaufania (DFTS)	System projektowania i rozwoju oprogramowania, na którym można polegać (obejmuje to niezawodność, bezpieczeństwo, zabezpieczenia, dostępność i możliwość konserwacji, choć nie ogranicza się do tych cech) i które pozwala reagować na klienta na różnych etapach cyklu życia oprogramowania.
Solidna architektura (nazywana także modelem rozwoju solidnego oprogramowania — RSDM)	Model rozwoju oprogramowania służący do tworzenia oprogramowania godnego zaufania (zobacz rysunek 2.6).
Solidne projektowanie	Metodologia utworzona przez Genichiego Taguchiego w celu rozwoju najniższym możliwym kosztem produktów i procesów działających na poziomie docelowym zgodnie z wymaganiami klienta, mimo obecności czynników, które powodują zmienność w środowisku użytkownika i produkcyjnym.
Six Sigma	Filozofia, system zarządzania i metodologia stosowane w celu usprawniania istniejących produktów, procesów i usług tak, aby były pozbawione błędów i spełniały wymagania klientów w ekonomiczny sposób.
Oprogramowanie	Programy komputerowe, procedury i (zwykle) związane z nimi dokumentacja oraz dane dotyczące działania systemu komputerowego.
Dostępność oprogramowania	Możliwość udostępniania przez oprogramowanie określonych funkcji, kiedy użytkownik ich potrzebuje.
Wiarygodność oprogramowania	Stopień, w jakim systemy, komponenty lub procesy producenta oprogramowania potrafią spełniać określone wymagania oraz potrzeby i oczekiwania użytkowników.

TABELA 2.2.
Wybrane atrybuty związane z jakością oprogramowania — *ciąg dalszy*

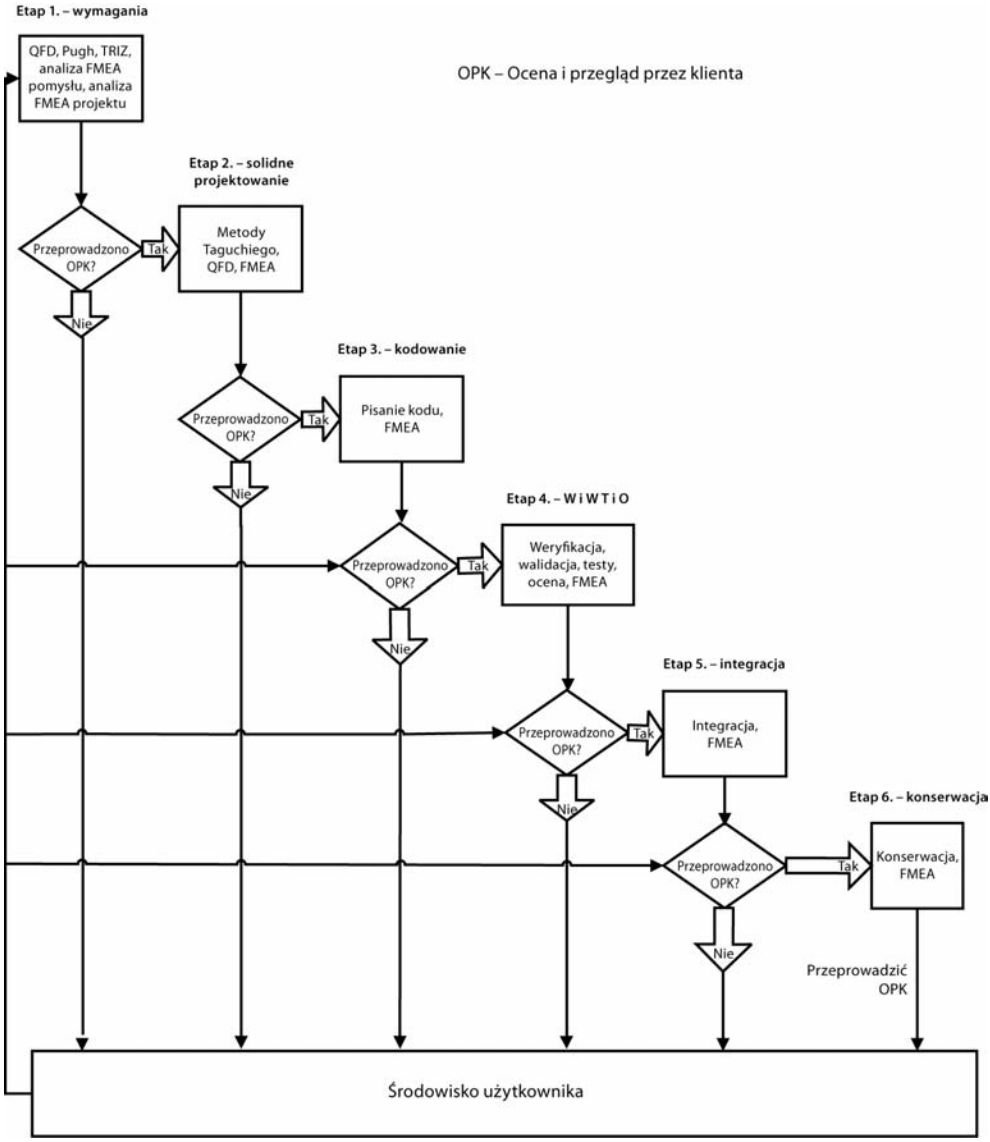
Jakość oraz atrybuty i systemy jakości	Opis
Solidność oprogramowania	Obejmuje, wśród innych atrybutów, niezawodność, bezpieczeństwo, zabezpieczenia i dostępność.
Projekt oprogramowania	Architektura i kod programu wykonującego określoną funkcję.
Możliwość konserwacji oprogramowania	Łatwość modyfikowania systemów lub komponentów oprogramowania po ich udostępnieniu w celu naprawy błędów, poprawy działania i innych cech lub zaadaptowania do zmienionego środowiska. Często określa się ją jako MTBF/(MTBF + MTTR).
Jakość oprogramowania	Zdatność oprogramowania do użytku. Stopień, w jakim oprogramowanie ma określony zestaw atrybutów potrzebnych do spełniania jawnych lub ukrytych potrzeb klienta i zapewnia jego zadowolenie. Prawidłowe działanie programu jest niezbędne, ale niewystarczające, jeśli oprogramowanie nie zapewnia satysfakcji klienta.
Atrybuty jakości oprogramowania	Różne wymagania dotyczące oprogramowania, takie jak niezawodność, bezpieczeństwo, zabezpieczenia i dostępność, potrzebne do spełnienia określonych lub ukrytych potrzeb.
Niezawodność oprogramowania	To pojęcie jest związane z jakością projektu oprogramowania. Wiąże się raczej z wykrywaniem błędów niż ich naprawianiem. Jest to możliwość wykonywania określonej funkcji przez system lub komponent oprogramowania w określonych warunkach i czasie.
Bezpieczeństwo oprogramowania	Brak czynników, które mogą spowodować śmierć, obrażenia, chorobę, uszkodzenia, brak kontroli lub dostępu do danych, naruszenie prywatności lub szkody w sprzęcie, mieniu i środowisku.
Skalowalność oprogramowania	Możliwość uruchomienia aplikacji komputerowej na większej maszynie lub procesorach równoległych w celu obsługi większej liczby transakcji lub zapewnienie wyższej przepustowości tak, aby wydajność skalowała się liniowo lub prawie liniowo pod względem ilości operacji. Oznacza to, że jeśli aplikacja potrafi obsługiwać określoną liczbą transakcji na danym serwerze, powinna skalować się tak, aby obsługiwała cztery razy większą ich liczbę na czterokrotnie większym serwerze.

TABELA 2.2.
Wybrane atrybuty związane z jakością oprogramowania — *ciąg dalszy*

Jakość oraz atrybuty i systemy jakości	Opis
Zabezpieczenia oprogramowania	Cechy oprogramowania dotyczące jego odporności na atak i zapewniające ochronę poufności, integralności danych oraz dostępności systemu.
Szybkość realizacji transakcji przez oprogramowanie	Tempo obsługi transakcji przez aplikację na danym komputerze, zwykle mierzone w tysiącach transakcji na minutę.
Możliwość aktualizowania oprogramowania	Możliwość łatwej zmiany konfiguracji oprogramowania w celu obsługi większej liczby, większych lub bardziej skomplikowanych transakcji.
Przetwarzanie godne zaufania	System składający się ze sprzętu, oprogramowania i sieci, na którym można polegać (obejmuje to niezawodność, bezpieczeństwo, zabezpieczenia, dostępność i możliwość konserwacji, choć nie ogranicza się do tych cech) i który umożliwia reagowanie na klienta na różnych etapach cyklu życia.
Oprogramowanie godne zaufania	Oprogramowanie, na którym można polegać (obejmuje to niezawodność, bezpieczeństwo, zabezpieczenia, dostępność i możliwość konserwacji, choć nie ogranicza się do tych cech) i które umożliwia reagowanie na klienta na różnych etapach cyklu życia.

niejawnych, a nawet nieoczekiwanych potrzeb, a cechy te są tu bardzo istotne. Pięć głównych wyzwań związanych z tworzeniem oprogramowania godnego zaufania to zapewnienie następujących cech:

- **Niezawodności**, czyli możliwości wykonywania przez oprogramowanie oczekiwanych funkcji w określonych warunkach i czasie. Jest to jakość związana z projektem oprogramowania i dotyczy raczej wykrywania błędów niż ich naprawiania.
- **Bezpieczeństwa**, które dotyczy nieobecności czynników mogących spowodować śmierć, obrażenia, chorobę, uszkodzenia, brak dostępu lub kontroli danych, naruszenie prywatności czy szkody w sprzęcie, mieniu lub środowisku.
- **Zabezpieczeń**, związanych z odpornością oprogramowania na atak, co zapewnia ochronę poufności i integralności danych oraz dostępu do systemu.



RYSUNEK 2.6.
Model rozwoju solidnego oprogramowania

- **Możliwości konserwacji**, która dotyczy łatwości modyfikowania systemów lub komponentów oprogramowania po ich udostępnieniu w celu naprawy błędów, poprawy działania lub innych cech albo adaptacji produktu do zmieniającego się środowiska.
- **Reagowania na klienta**, czyli możliwości producenta związanych z uzyskiwaniem i interpretowaniem wymagań klienta oraz reagowaniem na nie. Wymaga to także

obecności odpowiednich cech solidnego projektu oprogramowania, możliwości prowadzenia szkoleń i przekazywania wiedzy, umiejętności pomocy w integracji istniejących systemów, udostępniania pomocy technicznej po wdrożeniu, możliwości udostępniania zaktualizowanego oprogramowania i systemów oraz spełniania wymagań klientów w zakresie kosztów i harmonogramu dostarczania produktów. Szczególnie istotna jest możliwość zdobywania **zaufania klientów** i **spełnianie ich jawnych, niejawnych, a nawet nieoczekiwanych potrzeb**.

Są to główne aspekty oprogramowania godnego zaufania, które jednak są potrzebne w różnym stopniu w zależności od kategorii oprogramowania i jego zastosowań. Przykładowo **reagowanie na klienta** to szczególnie ważny element w przypadku oprogramowania dla przedsiębiorstw. Ważne jest tu, aby twórca oprogramowania znał i uwzględniał **głos klienta** (ang. *voice of customer* — VOC), interpretował go prawidłowo i mógł na tej podstawie tworzyć oprogramowanie godne zaufania.

Warto poczynić pewne uwagi na temat zwrotu „godne zaufania". W dziedzinie zarządzania jakością tego wyrażenia po raz pierwszy użył Deming, który stosował je w znaczeniu czynnika decydującego o wyborze dostawców w kontekście „usuwania lęków" wśród pracowników. Uważamy zastosowanie tego słowa przez Deminga i kontekst, w jakim go używał, za bardzo znaczące dla komunikatu, który sami chcemy przekazać: godne zaufania i niezawodne oprogramowanie, a w zasadzie dowolny produkt i każda usługa, mogą być udostępniane tylko przez osoby godne zaufania. Tego zwrotu użyto także w programie przetwarzania godnego zaufania (ang. *Trushworthy Computing* — TWC) Microsoftu uruchomionym w 2002 roku. Prezes Microsoftu, Bill Gates, w przełomowych powiadomieniach przesłanych w styczniu i lipcu 2002[15] do 50000 pracowników korporacji na całym świecie napisał: „W przeszłości staraliśmy się, aby nasze oprogramowanie i usługi były atrakcyjne dla klientów ze względu na nowe funkcje i przez możliwość bogatego rozszerzania naszej platformy [...] wykonaliśmy pod tym względem doskonałą robotę, jednak wszystkie te wspaniałe możliwości okażą się nieistotne, jeśli klienci nie będą ufać naszym produktom. Dlatego teraz w sytuacji wyboru między nowymi funkcjami i rozwiązywaniem problemu z zabezpieczeniami musimy stawiać na zabezpieczenia". Gates ponadto napisał, że wierzy, iż TWC „będzie najwyższym priorytetem dla firmy i przemysłu przez następną dekadę — celem jest utworzenie dla klientów środowiska przetwarzania godnego zaufania, które jest równie niezawodne, co elektryczność zasilająca obecnie nasze domy i firmy". Zapewnienie oprogramowaniu równej niezawodności, co w przypadku elektryczności, to olbrzymie wyzwanie dla przemysłu związanego z produkcją oprogramowania. Wyraźnie widać potrzebę współpracy między przemysłem rozwoju oprogramowania, profesjonalistami z branży oprogramowania, użytkownikami oprogramowania, agencjami odpowiedzialnymi za regulacje i instytutami badawczymi na całym świecie. Proponowana w tej książce metodologia DFTS zapewnia spójną strukturę i technologię do rozwiązywania tego typu problemów z jakością.

Model rozwoju solidnego oprogramowania — proces DFTS w praktyce

Oprogramowanie w porównaniu z innymi produktami tworzonymi za pomocą inżynierii to przykład czystego projektu. Jak już wspomnieliśmy, zawodność oprogramowania to zawsze wynik błędów w projekcie i intelektualnych braków człowieka[16]. Dlatego jeszcze bardziej istotny jest w tym przypadku moment, w którym rozwiązywane są problemy z jakością. Filozofia i systemy proponowane w tej książce zapewniają twórcom oprogramowania działającą na wczesnych etapach metodologię, która pozwala zidentyfikować optymalne funkcje i ustawienia solidnego (godnego zaufania) oprogramowania. Omówione wcześniej elementy systemu są przedstawione w proponowanym przez nas modelu rozwoju solidnego oprogramowania (ang. *Robust Software Development Model* — RSDM) utworzonym jako ułatwienie do rozwoju oprogramowania godnego zaufania (zobacz rysunek 2.6). Ten model spełnia **siedem kluczowych wymagań** procesu rozwoju solidnego oprogramowania omówionego w rozdziale 1. Bazuje na logicznych zasadach zarządzania i sprawdzonych narzędziach, technikach i metodologiach charakteryzujących się następującymi kluczowymi elementami:

- Infrastrukturą zapewniającą organizacji konieczne przywództwo i system komunikacji, szkoleń oraz nagród, który wyraźnie wspiera proces DFTS (zobacz rozdział 5.).
- Niezawodnym systemem zbierania danych, który pozwala prawidłowo wykrywać wymagania użytkowników (VOC) w iteracyjny sposób na różnych etapach cyklu rozwoju oprogramowania (zobacz rozdział 11.).
- Wdrażaniem metod Taguchiego w celu optymalizacji projektowania oprogramowania i jednocześnie uwzględniania różnych wymagań klienta, takich jak niezawodność, koszty i czas trwania cyklu (zobacz rozdziały 16. i 17.).
- Ustanowieniem praktyki jednoczesnego pisania kodu i przeprowadzania testów oraz zapewniania wystarczającego czasu na usuwanie błędów. Ta strategia prowadzi do procesu debugowania wydajnego ze względu na koszty i czas, ponieważ informacje o natężeniu lub częstotliwości błędów oprogramowania są wtedy szybciej dostępne (zobacz rozdział 18.).
- Tworzeniem wielu wersji programu[17] w sytuacji, kiedy potrzebne jest nadmiarowe oprogramowanie. Sprawia to, że statystycznie awarie nadmiarowych kopii są tak niezależne, jak to możliwe. W tym celu w różnych programach należy stosować odmienne języki programowania, narzędzia programistyczne, technologie rozwoju i strategie testowania (zobacz rozdział 14.).
- Wdrażaniem odpowiednich narzędzi do zarządzania jakością i planowania, takich jak QFD, TRIZ, Pugh i FMEA, które są szeroko stosowane w produkcji, co jest zgodne z najlepszymi praktykami (zobacz odpowiednio rozdziały 11., 12. i 13.).

- Stosowaniem innowacyjnych narzędzi do rozwoju oprogramowania, takich jak projektowanie obiektowe (ang. *Object-Oriented Design* — OOD), programowanie ekstremalne czy odpowiednie narzędzia CASE.

Będziemy na zmianę odnosić się do modelu RSDM i procesu DFTS — model opisuje proces, a proces jest ilustrowany przez model. W kilku ostatnich dziesięcioleciach wyewoluowały liczne modele rozwoju oprogramowania. Wiele z nich, na przykład model wodospadu, etapowe modele cyklu życia, model spiralny czy model V, mają swe początki w lotnictwie i innych gałęziach przemysłu produkcyjnego, dlatego nie zawsze odpowiadają rzeczywistości procesu rozwoju oprogramowania. RSDM to model iteracyjny, który umożliwia interakcję zarówno z wewnętrznymi, jak i z zewnętrznymi klientami oraz uchwycenie VOC w procesie rozwoju. Ponadto jest solidny i elastyczny, a także można go dostosować do dowolnego procesu rozwoju oprogramowania. W następnych rozdziałach opisujemy zastosowania tego modelu i jego elementy ze szczególnym uwzględnieniem kontekstu rozwoju oprogramowania dla przedsiębiorstw.

Chcemy przypomnieć, że w organizacjach zajmujących się rozwojem oprogramowania i w innych firmach, w których prace nad technologiami informatycznymi stanowią istotną działalność, rozwój oprogramowania jest zbyt ważny, aby pozostawić go samym inżynierom oprogramowania. Zarządzanie tą aktywnością należy do obowiązków prezesa i wyższej kadry zarządzającej. Muszą oni zapewnić niezbędne przywództwo, utworzyć prawidłową infrastrukturę zarządzania i rozwijać środowisko pod kątem tworzenia oprogramowania godnego zaufania.

Kluczowe zagadnienia

- Wieloaspektowe spojrzenie na jakość jest niezbędne do zrozumienia i spełnienia zestawu jawnych oraz niejawnych wymagań klientów i innych partnerów.
- Zazwyczaj zasady, systemy i metodologie zarządzania jakością odpowiednie dla wyrobów produkowanych można stosować także dla oprogramowania. Jednak oprogramowanie wiąże się ze specyficznym środowiskiem projektowania i rozwoju. Trzeba zrozumieć złożoność często związaną z oprogramowaniem i uwzględnić innowacyjność oraz trudność zadań, do których obsługi jest projektowane.
- Zadanie utworzenia oprogramowania godnego zaufania to prawdziwe wyzwanie i wymaga istotnego zaangażowania kadry zarządzającej.
- Tradycyjne systemy kontroli jakości bazują na dwóch błędnych założeniach: po pierwsze, wymagania klienta są spełnione, jeśli produkt jest zgodny ze specyfikacją, a po drugie, biznesowe skutki sytuacji, w których jakość jest „ledwie zgodna ze specyfikacją" i „na poziomie docelowym", są takie same.
- 14 punktów zarządzania Deminga służących do poprawy jakości obejmuje wsłuchiwanie się w głos klientów, zmniejszanie wariancji, stosowanie miar

statystycznych, zdobywanie zaufania i szacunku współpracowników oraz ciągłe usprawnianie. Deming wskazuje na braki systemów zarządzania jakością zależnych od kontroli.

- Japoński inżynier przemysłowy Genichi Taguchi rozwinął alternatywny system zarządzania jakością — metody Taguchiego. Taguchi podkreśla wartość „poziomu docelowego" i zaleca dbałość o jakość na wczesnych etapach, zamiast zależności od kontroli mających wykryć i naprawić błędy w dalszych fazach rozwoju.
- Filozofię zarządzania jakością Taguchiego można podsumować w następujący sposób:
  - Ciągła poprawa jakości i redukcja kosztów są konieczne dla przetrwania biznesu.
  - Ważną miarą jakości jest ogólna strata wygenerowana przez dany produkt w społeczności, mierzona funkcją utraty jakości.
  - Należy wbudować jakość w produkt lub proces poprzez projekt, używając techniki statystycznego projektowania eksperymentów.
  - Straty klientów z powodu niskiej jakości są nieliniowe i możne je oszacować jako kwadrat odchylenia cech działania od ich poziomu docelowego.
  - Zmienność działania produktu można zmniejszyć, analizując nieliniowy wpływ „czynników kontroli" na cechy funkcjonowania.
- Solidne projektowanie przy użyciu metod Taguchiego przebiega w trzech fazach: projektowania systemu, projektowania parametrów i projektowania tolerancji.
- Oprogramowanie godne zaufania spełnia liczne jawne i niejawne potrzeby klientów, a także musi umożliwiać dostosowanie produktu do nich. W kontekście oprogramowania dla przedsiębiorstw oprogramowanie godne zaufania musi spełniać przynajmniej następujące wymagania: niezawodność, bezpieczeństwo, zabezpieczenia, możliwość konserwacji i reagowanie na klienta.
- Proces DFTS charakteryzuje się określonymi cechami. Oto one:
  - prawdziwe zaangażowanie kadry zarządzającej i wspierająca to infrastruktura;
  - możliwość identyfikacji jawnych i niejawnych wymagań za pomocą QFD;
  - optymalizacja wymagań klienta poprzez wdrożenie metod Taguchiego;
  - ustanowienie praktyki jednoczesnego pisania kodu i przeprowadzania testów;
  - stosowanie w razie konieczności nadmiarowego oprogramowania;
  - wdrażanie odpowiednich narzędzi do zarządzania jakością i planowania, takich jak TRIZ, Pugh i FMEA;
  - stosowanie innowacyjnych narzędzi do rozwoju oprogramowania, takich jak projektowanie obiektowe.

Dodatkowe materiały

http://www.prenhallprofessional.com/title/0131872508
http://www.agilenty.com/publications

Ćwiczenia internetowe

http://www.asiusa.com/publications/images/HBR.pdf

Po przeczytaniu artykułu Taguchiego i Clausinga dostępnego pod powyższym adresem przygotuj się do dyskusji wniosków na jego temat.

1. Przeanalizuj problemy związane ze stylem myślenia „w ramach specyfikacji — poza specyfikacją" powiązanym z podejściem „zero defektów" w kontekście oprogramowania. Jakie rozwiązanie oferuje metodologia Taguchiego?
2. Podaj trzy przykłady „sygnału" i „szumu" w kontekście rozwoju oprogramowania.
3. Zaproponuj zestaw zaleceń umożliwiających usprawnienie procesu rozwoju oprogramowania.

Ten artykuł jest dostępny także w następujących miejscach:

- Genichi Taguchi i Don Clausing, *Robust Quality*, „Harvard Business Review", Styczeń – Luty 1990.
- *http://harvardbusinessonline.hbsp.harvard.edu/b01/en/common/item_detail.jhtml?id=*
  ↪90114&referral=8636&_requestid=9765.

Pytania przeglądowe

1. Opisz, jak wieloaspektowe spojrzenie na jakość pomaga zaspokoić potrzeby różnorodnych partnerów. Zilustruj odpowiedź przykładami związanymi ze znanym Ci oprogramowaniem.
2. Czy problemy z jakością oprogramowania są zasadniczo odmienne od występujących w wyrobach produkowanych? Jakie są podobieństwa i różnice między oprogramowaniem i wyrobami produkowanymi w zakresie procesu ich rozwoju?
3. Porównaj niezawodność oprogramowania i sprzętu. Wyjaśnij skutki takiego stanu rzeczy na przykładzie trzech złożonych systemów składających się z oprogramowania i sprzętu.
4. Wymień główne powody zawodności oprogramowania. Które z nich uważasz za najważniejsze?
5. Opisz dwa najważniejsze problemy z tradycyjnymi systemami kontroli jakości i wpływ, jaki mają na oprogramowanie.

6. Opisz istotę 14 punktów zarządzania Deminga i wyjaśnij ich znaczenie w dzisiejszym świecie.
7. Wyjaśnij, jak metody Taguchiego wiążą się z zaleceniami Deminga wymienionymi w 14 punktach i jak je wspierają.
8. Jak brzmi pięć filozoficznych nakazów podejścia Taguchiego? Wyjaśnij ich związek z rozwojem oprogramowania. Czy w przypadku sprzętu są one inne? Jeśli tak, to w jaki sposób?
9. Podsumuj kluczowe zasady metod Taguchiego i trzy etapy solidnego oprogramowania. Opisz, jak wspomagają one proces rozwoju oprogramowania.
10. Wymień i opisz pięć głównych wyzwań na drodze do tworzenia oprogramowania godnego zaufania. Zilustruj odpowiedź dwoma znanymi Ci produktami z dziedziny oprogramowania.
11. Opisz cechy modelu rozwoju solidnego oprogramowania. Wyjaśnij, jak te właściwości oraz sam model jako całość można porównać z dwoma podobnymi modelami opisanymi w rozdziale 1.

Pytania do dyskusji i projekty

1. Jak 14 punktów zarządzania Deminga rozwiązuje ograniczenia tradycyjnych systemów kontroli jakości? Możesz posłużyć się tabelami 5.2, 5.3 i 5.4 z rozdziału 5. Jak zastosować wskazówki Deminga do branży rozwoju oprogramowania? Przedstaw odpowiedź na zajęciach.
2. Omów i porównaj zastosowanie metodologii Taguchiego w przypadku oprogramowania dla przedsiębiorstw i produktów fabrykowanych. Możesz posłużyć się materiałem z rozdziałów od 16. do 19. Przedstaw odpowiedź na zajęciach.
3. Opisz zalety i wyzwania związane z wdrażaniem w organizacji metodologii projektowania oprogramowania godnego zaufania (DFTS). Jakie są możliwe źródła oporu przy jej wprowadzaniu? Jak można sobie z nimi poradzić? Przedstaw odpowiedź na zajęciach.
4. Wyobraź sobie, że jesteś członkiem zespołu zarządzającego wysokiego stopnia kierowanego przez prezesa. Zespół otrzymał od zarządu firmy zajmującej się rozwojem oprogramowania zadanie identyfikacji głównych przyczyn problemów z jakością oprogramowania i zaproponowania platformy umożliwiającej ich rozwiązanie. Napisz informację dla zarządu po wstępnej ocenie. Zaprezentuj ją na zajęciach.

Przypisy

[1] Bev Littlewood i Lorenzo Strigini, *Software Reliability and Dependability: A Roadmap*, Proc. ICSE 2000, 22nd International Conference on Software Engineering, s. 2.

[2] W. Kuo, V. Rajendra Prasad, F. A. Tillman, Ching-Lai Wand, *Optimal Reliability Design* (Cambridge, Cambridge University Press, 2001), punkt 13.5.1, s. 325.

[3] Ibid., punkt 1.3.3., s. 5.

[4] D. Simmons, N. Ellis, H. W. Kuo, *Software Measurement: A Visualization Toolkit for Project Control and Process Improvement* (Prentice-Hall, 1998).

[5] N. Logothetis, *Managing for Total Quality* (London, Prentice-Hall International, 1992), s. 27.

[6] Zaadaptowane za przyzwoleniem, op. cit., Kuo i inni, s. 4.

[7] W. Edwards Deming, *Out of the Crisis* (Cambridge, MA, MIT Press, 2000). Należy zwrócić szczególną uwagę na punkt 3. z 14 punktów o zarządzaniu.

[8] *http://www.asiusa.com/about/asi_thought_genichi.aspx*.

[9] *http://www.berr.gov.uk*.

[10] Genichi Taguchi i Don Clausing, *Robust Quality*, „Harvard Business Review", styczeń – luty 1990, s. 68.

[11] Zaadaptowane z książki *Out of Crisis* Deminga.

[12] Yuin Wu i Alan Wu, *Taguchi Methods for Robust Design* (Nowy Jork, ASME, 2000), s. 13 – 28.

[13] William Y. Fowlkes i Clyde M. Creveling, *Engineering Methods for Robust Product Design: Using Taguchi Methods in Technology and Product Development* (Reading, MA, Addison-Wesley, 1997).

[14] Op. cit. Littlewood i Strigini, s. 1.

[15] *http://www.microsoft.com/mscorp/execmail/2002/07-18twc.mspx*.

[16] Op. cit. Littlewood i Strigini, s. 2.

[17] N. Ashrafi, O. Berman, M. Cutler, *Optimal design of large software-systems using N-version programming*, „IEEE Transactions on Reliability", R-43 (2): 344 – 350, 1994.