



SZKOŁA GŁÓWNA HANDLOWA W WARSZAWIE
KATEDRA TEORII ZARZĄDZANIA
PODYPLOMOWE STUDIA
ZARZĄDZANIA PRODUKTAMI I USŁUGAMI

Jacek Śliwerski
Nr albumu: 55439

**Formalna weryfikacja programów
komputerowych**
Nowy segment rynku usług IT

Praca końcowa napisana w Katedrze Teorii Zarządzania
pod kierunkiem naukowym dra Rafała Mrówki

Warszawa, 2009

Spis treści

Wstęp	1
1. Opis produktu	2
1.1. Definicja usługi	2
1.2. Proces weryfikacji	4
1.3. Techniki weryfikacji	5
1.4. Podsumowanie	6
2. Analiza rynku	8
2.1. Otoczenie makroekonomiczne	8
2.1.1. Trendy	9
2.1.2. Błędy w oprogramowaniu	10
2.1.3. Przeciwdziałanie błędom	11
2.2. Analiza strukturalna sektora	12
2.2.1. Rywalizacja w sektorze	13
2.2.2. Zagrożenie ze strony substytutów	13
2.2.3. Groźba nowych wejść	15
2.2.4. Siła przetargowa nabywców	15
2.2.5. Siła przetargowa dostawców	15
2.3. Analiza SWOT	16
2.3.1. Mocne strony	17
2.3.2. Słabe strony	17
2.3.3. Szanse	17
2.3.4. Zagrożenia	18
2.4. Podsumowanie	19
3. Strategia	20
3.1. Segmentacja	20
3.2. Pozycjonowanie	22
3.3. Plan strategiczny	23
3.3.1. Misja	23
3.3.2. Wartości	24

3.3.3. Cele	24
3.4. Kompozycja marketingowa	25
3.4.1. Produkt	25
3.4.2. Cena	26
3.4.3. Dystrybucja	26
3.4.4. Promocja	27
3.5. Monitoring ryzyka	29
3.6. Podsumowanie	30
Bibliografia	31

Wstęp

Niniejszy dokument jest pracą końcową napisaną na zakończenie Podyplomowych Studiów Zarządzania Produktami i Usługami. Celem niniejszej pracy jest opisanie strategii wejścia na rynek usługi polegającej na formalnej weryfikacji zgodności programów komputerowych ze stawianymi im wymaganiami.

W rozdziale 1 została opisana usługa weryfikacji oprogramowania. Następujący po nim rozdział 2 analizuje rynek produktów i usług związanych z formalną weryfikacją oprogramowania oraz nakreśla najważniejsze atuty i problemy produktu na tle tak opisaney sytuacji. W rozdziale 3 została przybliżona strategia wejścia na rynek. Każdy z rozdziałów kończy się krótkim podsumowaniem.

Rozdział 1

Opis produktu

Program komputerowy jest tylko połową tezy. Drugą połową jest specyfikacja funkcjonalna, którą ów program powinien spełniać. Zadaniem programisty jest przedstawienie pełnej tezy w postaci udowodnionego twierdzenia[1].

Edsger W. Dijkstra

W rozdziale tym została przedstawiona usługa niematerialna polegająca na przygotowaniu dowodu zgodności programu komputerowego ze stawianymi wymaganiami. W sekcji 1.1 została omówiona istota weryfikacji programów komputerowych. Proces jej realizacji opisuje sekcja 1.2, po której następuje uszczegółowienie technik i metodologii (sekcji 1.3). Rozdział kończy się krótkim podsumowaniem w sekcji 1.4.

1.1. Definicja usługi

Kupując samochód, lodówkę, odkurzacz, aparat fotograficzny, telefon czy telewizor, jesteśmy objęci ochroną prawną gwarantowaną w kodeksie cywilnym i ustawie o szczególnych warunkach sprzedaży konsumenckiej. Nawet, jeśli producent urządzenia nie objął jego poprawnego działania gwarancją, przysługuje nam prawo do reklamacji wadliwego produktu na zasadach rękojmi, o ile ma on wady fizyczne bądź prawne. Jeśli dane urządzenie nie działa zgodnie ze swoim przeznaczeniem, możemy zażądać jego naprawy, wymiany bądź obniżenia ceny.

Zupełnie inaczej wyglądają prawa konsumenta w odniesieniu do oprogramowania. Twórcy i dystrybutorzy programów komputerowych nie podlegają

zobowiązaniom ustawowym (arkusz kalkulacyjny nie może wszak mieć istotnej wady fizycznej) i rozpowszechniają je na zasadach przypominających sprzedaż książek, muzyki oraz innych dóbr objętych przepisami prawa autorskiego. I choć program komputerowy może zawierać istotne wady, które uniemożliwiają wykorzystanie go do pracy, użytkownicy są zmuszani do akceptowania warunków licencyjnych, które nie tylko zdejmują z producenta wszelką odpowiedzialność za konsekwencje wadliwego działania, lecz bardzo często dodatkowo ograniczają prawa użytkownika do wykorzystania zakupionego programu zgodnie z jego wolą.

Produktem w niniejszej pracy jest usługa polegająca na przygotowaniu formalnego dowodu zgodności programu z zadanymi wymaganiami zwana również (formalną) weryfikacją, bądź formalną weryfikacją. Umożliwia ona producentom programów komputerowych udzielanie użytkownikom gwarancji ich poprawności w ramach precyzyjnych wymogów. Gwarancja ta, oparta na ścisłym matematycznym wnioskowaniu jest wieczysta, a jej realizacja nie wymaga żadnych dodatkowych kosztów. W wyniku realizacji tej usługi, powstają następujące elementy:

Dowód. Na dowód poprawności składają się: konkretna wersja weryfikowanego programu, zestaw kontrolowanych wymagań oraz potwierdzenie zgodności systemu z zadanymi warunkami.

Metodologia. Istnieje wiele sposobów weryfikacji oprogramowania, ale żaden z nich nie jest uniwersalny. W sekcji 1.3 zostały opisane trzy przykładowe techniki.

Narzędzia. Ocena zgodności dowodu z metodologią może być procesem niezwykle żmudnym. Dlatego konieczne są dodatkowe narzędzia, które mogą ów proces zautomatyzować.

Przyjmijmy dla przykładu, że weryfikowanym programem jest system autopilota samolotu pasażerskiego. Przykładowym wymogiem może być niedopuszczenie, aby wartości komórek pamięci przechowujących szerokość geograficzną wykroczyły poza zakres od zera do dziewięćdziesięciu¹.

Dowód zgodności programu z wymaganiami niewiele różni się od dowodów twierdzeń matematycznych i powinien podlegać takiej samej procedurze sprawdzenia przez niezależnych ekspertów. Jednak przeciętny użytkownik nie dysponuje wiedzą i doświadczeniem niezbędnymi do tego aby samemu dokonać weryfikacji, dlatego też niezbędne jest otwarcie zarówno metodologii jak i samego dowodu dla szerokiej publiczności, tak, by jak

¹Błąd w jednej z pierwszych wersji oprogramowania ustalającego pozycję samolotów F-16 powodował, że po przekroczeniu równika maszyny te obracały się automatycznie podwoziem do góry.

największa liczba osób mogła naocznie przekonać się do prawdziwości i poprawności przedstawionego wniosku.

Ponieważ jednak program komputerowy jest obiektem znacznie bardziej skomplikowanym od przeciętnej abstrakcji matematycznej, proces kontroli musi być wspomagany odpowiednio przygotowanymi narzędziami, które również muszą być publicznie dostępne, a zasada ich działania musi być jawna w najdrobniejszych szczegółach.

Przejrzystość procesu weryfikacji, publiczna dostępność narzędzi oraz jawny charakter samego dowodu są jedynymi gwarantami wiarygodności ostatecznego wyniku. W następnej sekcji zostały opisane etapy, w których taki właśnie dowód powstaje.

1.2. Proces weryfikacji

Proces dowodzenia zgodności programu z wymaganiami nie sprowadza się wyłącznie do stwierdzenia, że system jest bądź nie jest poprawny. Analiza programu pod kątem jego formalnej poprawności pozwala znaleźć błędy zarówno na poziomie koncepcji jak i detali implementacyjnych.

1. W pierwszej kolejności formułowane są wymagania dotyczące systemu. Muszą one być sformalizowane w jednym z dostępnych języków logiki, a wszelkie konkrety – przekształcone na odpowiadające im abstrakcje. W zależności od rodzaju oprogramowania, stosuje się różnego rodzaju języki:
 - *Logiki temporalne* są szczególnie przydatne w przypadku tzw. *systemów reaktywnych*. Pozwalają one opisywać właściwości stale działającego programu w odniesieniu do czasu, np. „za każdym razem, gdy program blokuje semafor, po pewnym czasie semafor ten zostanie odblokowany”.
 - Klasyczny *rachunek zdań* pierwszego bądź drugiego rzędu stosowany jest przede wszystkim dla systemów nie wchodzących w interakcje z równoległe uruchomionymi procesami.

Formalizacja wymagań pozwala zwykle lepiej zrozumieć system i już na tym etapie jest możliwa identyfikacja pierwszych błędów systemu powstałych za zwyczaj na etapie projektu.

2. Następnie weryfikowany program należy zamienić na odpowiednio dobrany model, który opisuje cechy programu istotne z punktu widzenia procesu weryfikacji, a abstrahuje wszystkie pozostałe bez utraty poprawności wniosku. W zależności od techniki weryfikacji może to

na przykład oznaczać redukcję liczby możliwych do przyjęcia stanów programu.

3. Tak powstały model programu jest konfrontowany z formalnym opisem wymagań. Ewentualne niedostatki wskazują błędy, które należy poprawiać tak długo, dopóki modele systemu i wymagań nie będą ze sobą zgodne.

Zarówno przekształcenie programu do odpowiedniego modelu, jak i jego konfrontacja z wymaganiami są zależne od rodzaju przyjętej techniki weryfikacyjnej. Kilka przykładowych metodologii zostało opisanych w następnej sekcji.

1.3. Techniki weryfikacji

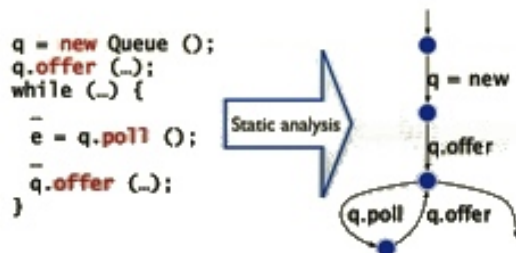
Poniżej przedstawiono krótki przegląd najpopularniejszych technik stosowanych do dowodzenia poprawności programów komputerowych wraz z ich zastosowaniami.

Model checking jest techniką, która wymaga przeprowadzenia dowodu poprawności w następujący sposób:

1. W pierwszej kolejności program upraszcza się do automatu o skończonej liczbie stanów (tzw. struktury Kripkego). Automat taki jest modelem programu.
2. Wymagania dotyczące programu formułuje się przy pomocy logik temporalnych. Pozwalają one na opisywanie reguł, w jaki sposób stan programu powinien zmieniać się w trakcie działania.
3. Zgodność modelu z wymaganiami sprawdza się albo poprzez przeszukanie wszystkich możliwych stanów, albo poprzez odpowiednie przekształcenie obydwu abstrakcji do równoważnych im automatów Büchiego i ich „konfrontację”.

Technika ta została po raz pierwszy zastosowana przez naukowców Bell Laboratories w celu zagwarantowania poprawności działania oprogramowania central telefonicznych firmy Lucent[2]. Pracownicy NASA używali tego samego programu do weryfikacji algorytmów stosowanych w misjach eksploracyjnych Marsa (m.in. w bezzałogowej sondzie Cassini)[3]. Narzędzie o podobnej zasadzie działania stosowane jest przez firmę Microsoft do zapewnienia zgodności sterowników do systemów operacyjnych Windows z wymaganiami architektów systemu[4].

JADET: Building Models



Rysunek 1.1: Schemat działania programu JADET przekształcającego fragment programu komputerowego w odpowiadający mu model opisujący sekwencje wykonań.

Logika Hoare’a to formalizm stosowany do określania warunków wstępnych i końcowych, jakie musi spełniać stan programu, aby mógł się poprawnie wykonać. Dzięki zbiorowi precyzyjnie określonych reguł przekształcania stanu w zależności od wykonywanych instrukcji, można opisać stan programu w dowolnym momencie wykonania programu. Systemy oparte o logikę Hoare’a mogą (w przeciwieństwie do struktur Kripkego) opisywać modele również o nieskończonej ilości osiągalnych stanów.

Inferencja reguł Programy komputerowe można również traktować jako dane i przetwarzać przy pomocy klasycznych technik *data mining*. Pozwala to na automatyczną ekstrakcję reguł dotyczących części składowych programów nawet bez ich znajomości[5]. Rysunek 1.1 przedstawia schemat działania programu JADET, który na podstawie kodu źródłowego programu potrafi wywnioskować, że każde wywołanie metody `poll` jest poprzedzone wywołaniem metody `offer`. Tak wyprowadzona zasada może następnie zostać wykorzystana do weryfikacji pozostałych części systemu pod kątem jej przestrzegania.

1.4. Podsumowanie

Opisana w tym rozdziale usługa formalnej weryfikacji oprogramowania polega na przygotowaniu dowodu zgodności programu komputerowego ze sta-

wianymi wymaganiami. Umożliwia ona usługobiorcy zaoferowanie swoim klientom gwarancji na poprawność działania programu. W następnym rozdziale została przedstawiona aktualna sytuacja gałęzi sektora usług informatycznych związanej z zapewnianiem wysokiej jakości oprogramowania.

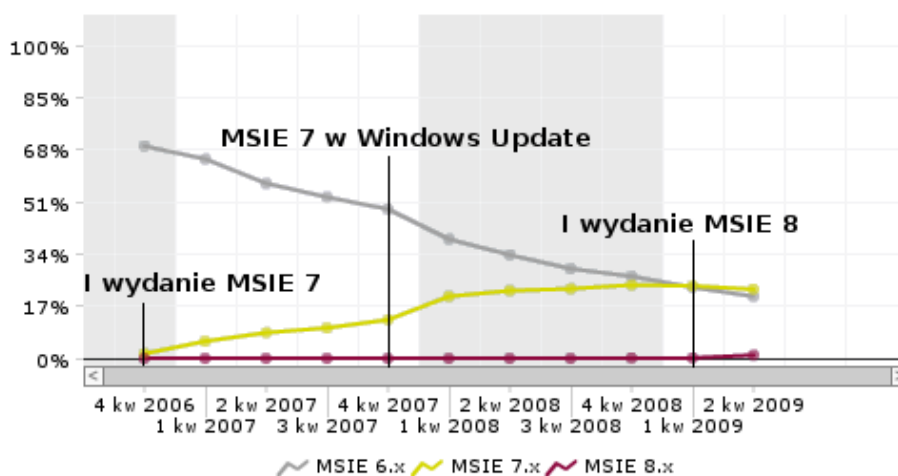
Rozdział 2

Analiza rynku

Aby móc oprzeć strategię wejścia na rynek na racjonalnych przesłankach, konieczna jest zarówno szczegółowa analiza docelowego rynku jak i kształtujących się na nim tendencji. Sekcja 2.1 opisuje otoczenie makroekonomiczne i kluczowe trendy panujące na rynku oprogramowania. Opis ten został pogłębiony przy pomocy *modelu pięciu sił Portera* w sekcji 2.2. Aby nie pozostawić tej analizy w oderwaniu od produktu, sekcja 2.3 przedstawia *analizę SWOT* usługi formalnej weryfikacji osadzonej w realiach tak zaprezentowanego rynku. Rozdział kończy się krótkim podsumowaniem w sekcji 2.4.

2.1. Otoczenie makroekonomiczne

Programy komputerowe odgrywają coraz ważniejszą rolę w życiu przeciętnego człowieka. Jesteśmy stale otoczeni urządzeniami sterowanymi przez programy komputerowe i coraz bardziej od nich zależni. Prowadzenie praktycznie każdego współczesnego środka transportu jest wspomagane przez oprogramowanie – od ABS, przez GPS i tempomaty aż po systemy naprowadzania samolotów do lądowania. Podróżowanie bez software’owego wspomagania może się stać w przyszłości niemożliwe. Ponad miliard komputerów zostało wyprodukowanych od roku 1970, z czego około połowa była w użyciu w 2008 roku[6]. Osiem lat temu sprzedaż programów komputerowych w USA osiągnęła wartość 180 milionów dolarów, a sektor IT zatrudniał 697 000 inżynierów oprogramowania i 585 000 programistów. Pod koniec 2008 roku sam IBM zatrudniał 426 969 pracowników i osiągnął przychody w wysokości 74 miliardów dolarów.



Rysunek 2.1: Udziały rynkowe trzech kolejnych wersji przeglądarki Microsoft Internet Explorer według danych firmy Gemius.

2.1.1. Trendy

Poniżej przedstawiono kilka istotnych trendów, które mają istotny wpływ na rynek usług związanych z oprogramowaniem:

Oprogramowanie jest czynnikiem różnicującym hardware. W sektorze elektroniki użytkowej obserwujemy trend przenoszenia ciężaru konkurencji z pola usprawnień czysto technicznych (więcej megaherców, megapikseli i gigabajtów) na pole coraz bardziej niezawodnego i prostszego w obsłudze oprogramowania ułatwiającego korzystanie z funkcji urządzenia. W wywiadzie dla tygodnika BusinessWeek, prezes firmy Microsoft Steve Ballmer powiedział[7]:

Mógłbym (...) sporządzić listę przypadków, w których oprogramowanie jest sprzedawane przez hardware. Tym właśnie jest iPod. iPod jest produktem software'owym. Po prostu przychody są generowane ze sprzedaży urządzenia.

Niechęć użytkowników do aktualizacji oprogramowania. Nowe wersje oprogramowania nadal przyjmują się niezwykle powoli. Siódma wersja przeglądarki Microsoft Internet Explorer potrzebowała ponad dwóch lat aby zrównać się w udziale w rynku ze swoją poprzedniczką (ilustracja trendów na rysunku 2.1), mimo że w połowie tego okresu firma Microsoft umieściła ją na liście oprogramowania automatycznie aktualizowanego wraz z systemem operacyjnym.

Upowszechnianie się oprogramowania open source. Według raportu

firmy Gartner już 85% firm korzysta z oprogramowania z wolnodostępnym kodem źródłowym[8]. Wszystkie pozostałe ankietowane organizacje spodziewały się rozpocząć korzystanie z bezpłatnych systemów w ciągu dwunastu miesięcy. Darmowe systemy stają się coraz większym zagrożeniem dla oprogramowania komercyjnego i odróżnienie się od konkurencji, której nie da się pobić na polu ceny staje się nie lada wyzwaniem.

Regulacja rynku. Globalny rynek oprogramowania nie podlega obecnie żadnym regulacjom prawnym. Ekspert bezpieczeństwa komputerowego Bruce Schneier zachęca ustawodawców do nakładania na firmy software'owe odpowiedzialności za sprzedawane produkty[9]. Uważa on, że jest to najlepszy sposób na podniesienie jakości programów komputerowych, a w konsekwencji bezpieczeństwa ich użytkowników. Specjalista prawa z zakresu oprogramowania Mark Methenitis przewiduje, że Federalna Komisja Komunikacji (FCC) nałoży ograniczenia na klauzule, jakimi producenci oprogramowania będą się mogli posługiwać w licencjach[10]. Komisja Europejska, w ramach tzw. „agendy cyfrowej” prowadzi już działania na rzecz rozszerzania zasad ochrony konsumentów na umowy licencyjne dotyczące m.in. oprogramowania[11].

2.1.2. Błędy w oprogramowaniu

Niestety, oprogramowanie, z którego korzystamy na co dzień nie jest pozbawione usterek. W 2002 roku *National Institute of Standards and Technology* oszacował roczne straty amerykańskiej gospodarki spowodowane błędami oprogramowania na 59,5 miliarda dolarów[12]. I choć ponad połowa tych strat była spowodowana przez użytkowników, około 22,2 miliarda dolarów mogło zostać zaoszczędzonych, gdyby odsetek błędów odkrytych w fazie, w której zostały wprowadzone, był większy.

Szacuje się, że nawet 80% kosztów wytworzenia programu jest związanych z wykrywaniem i usuwaniem usterek. Mimo tego, software należy do kategorii produktów dostarczanych z najwyższą liczbą wad. Wśród przyczyn takiego stanu rzeczy, naukowcy z *Research Triangle Institute* z Północnej Karoliny zidentyfikowali następujące powody:

- strategie marketingowe firm software'owych,
- ograniczona odpowiedzialność producentów oprogramowania,
- malejący zwrot z inwestycji w testowanie i poprawianie tworzonych programów.

2.1.3. Przeciwdziałanie błędom

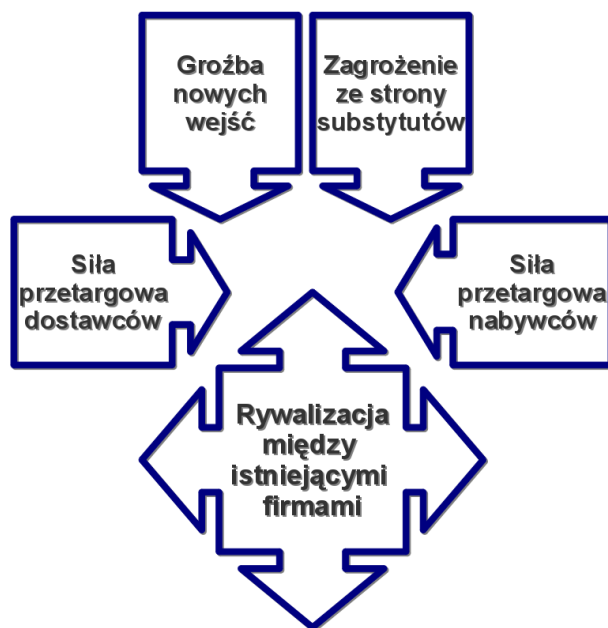
Wśród zaleceń Departamentu Handlu USA dla firm software'owych znalazły się znaczne usprawnienie procesu testowania poprzez standaryzację narzędzi, skryptów i zestawów testowych, wprowadzenie jednolitych metryk oraz rygorystyczna certyfikacja procesu kontroli jakości. I chociaż testowanie oprogramowania jest niewątpliwie niezbędnym elementem procesu jego wytwarzania, ma ono jedną poważną wadę: może tylko wykazać istnienie błędów, nigdy ich brak[13]. Nawet gruntownie przetestowane systemy doprowadziły kilkakrotnie do katastrof o gigantycznych kosztach. Oto kilka przykładów:

Ariane 5 Błąd przepełnienia szesnastobitowej zmiennej całkowitej spowodował awarię komputera rakiety *Ariane 5*, która pociągnęła za sobą łańcuch kolejnych defektów prowadzących ostatecznie do przegrzania silnika i uruchomienia procedury autodestrukcji po trzydziestu siedmiu sekundach lotu. Błąd programu wyrządził straty oszacowane na 370 milionów dolarów[14], pomimo że ten sam zestaw oprogramowania z powodzeniem zastosowano podczas lotu wcześniejszej wersji rakiety.

ESS4 Około dziewięciu godzin trwała przerwa w dostępie do rozmów międzymiastowych wywołana błędem w oprogramowaniu central telefonicznych firmy AT&T. Problem pojawiał się, gdy jedna z central wysyłała wiadomość do pozostałych o tym, że uruchomiła się ponownie po awarii[15]. Błąd w procedurze obsługującej otrzymanie takiej wiadomości doprowadzał do restartu maszyny. Reakcja łańcuchowa doprowadziła 114 urządzeń do ciągłego uruchamiania się co około 6 sekund.

Zune 30 Milion odtwarzaczy *Zune* firmy *Microsoft* nie działał przez 24 godziny ostatniego dnia roku 2008 na skutek błędu w procedurze rozbijającej datę na części składowe[16]. Problem ujawniał się wyłącznie ostatniego dnia roku przestępnego, lecz wywołał falę złości użytkowników skierowaną przeciwko producentowi. W styczniu 2009 roku *Microsoft* ogłosił plan zwolnień grupowych obejmujących w pierwszej kolejności pracowników pionu *Entertainment and Devices Division*, odpowiedzialnego między innymi za *Zune*.

Jak widać w przytoczonych przykładach, samo testowanie okazuje się niewystarczające do zapewnienia bezbłędnego działania oprogramowania, przede wszystkim dla tego, że błędy następują w sytuacjach, których nie sposób wypróbować. W przeciwieństwie do testowania, formalne metody weryfika-



Rysunek 2.2: Schemat modelu pięciu sił Portera

cji dają stuprocentową gwarancję, że oprogramowanie zachowa się zgodnie ze specyfikacją również w tych sytuacjach, które nie zostaną przetestowane.

2.2. Analiza strukturalna sektora

Jedną z popularnych metod analizy otoczenia konkurencyjnego przedsiębiorstwa jest *model pięciu sił Portera*[17]. Zgodnie z tą koncepcją atrakcyjność danego sektora jest w dużej mierze uzależniona od następujących pięciu czynników przedstawionych na diagramie 2.2:

1. **Rywalizacja w sektorze.** Centralnym kryterium kształtującym rentowność sektora jest aktualna rywalizacja podmiotów w nim funkcjonujących. Opłacalność działalności w danym środowisku jest odwrotnie proporcjonalna do intensywności konkurencji.
2. **Zagrożenie ze strony substytutów.** Istnienie zbliżonych produktów zwiększa skłonność klientów do wyboru alternatywy w odpowiedzi na zwiększanie ceny.
3. **Grożba nowych wejść.** Każdy wysokodochodowy rynek przyciąga nowe firmy, a rosnąca konkurencja wymusza ograniczanie marży, o ile bariery wejścia na rynek nie są zbyt wysokie.

4. **Siła przetargowa nabywców.** Silna pozycja negocjacyjna nabywców zmusza do obniżania cen.
5. **Siła przetargowa dostawców.** Silna pozycja dostawców wymusza podwyższanie cen – bądź ograniczanie marży.

W kolejnych sekcjach zostaną omówione wszystkie te czynniki z uwzględnieniem specyfiki sektora usług formalnej weryfikacji oprogramowania.

2.2.1. Rywalizacja w sektorze

Zdaniem firm wchodzących w skład konsorcjum Verisoft (sześć uniwersytetów oraz, między innymi, Audi, Bosch i Microsoft Research), w globalnym sektorze usług związanych z formalną weryfikacją programów komputerowych nie ma w tej chwili żadnej konkurencji[18]. Usługi związane z formalną weryfikacją oprogramowania są realizowane prawie wyłącznie przez uczelnie wyższe i instytuty naukowe oraz spółki celowe zakładane na potrzeby realizacji pojedynczych kontraktów.

W zbliżonej branży EDA (Electronic Design Automation), do której zalicza się firmy tworzące oprogramowanie do weryfikacji poprawności układów scalonych, działa czterech producentów oprogramowania. Lider rynku, firma Synopsys, zatrudnia ponad 5 600 osób i zanotowała ponad 1,3 miliarda dolarów przychodu za rok fiskalny 2008.

2.2.2. Zagrożenie ze strony substytutów

Substytutem dla weryfikacji programów komputerowych jest każdy produkt bądź usługa, na które twórcy oprogramowania mogą wydać pieniądze w celu poprawienia jakości swoich produktów czy też w celu przekonania użytkowników o ponadprzeciętnym poziomie wykonania ich programów. Do najważniejszych zamienników zaliczamy:

Outsourcing testowania. Pion testowania oprogramowania w firmie Wi-pro nazywa się *Verification & Validation*, mimo że nie oferuje dowodzenia poprawności programów. Dużymi dostawcami usług związanych z testowaniem i automatyzacją testów są również Lionbridge i Capgemini (pod marką Sogeti). Firmy te oferują ustandaryzowane i zautomatyzowane zestawy testowe dla szerokiej palety produktów software'owych. W ich interesie jest rozmywanie granic między formalnymi metodami dowodzenia poprawności – a weryfikacją przy pomocy różnych strategii testowania.

Certyfikacja procesu tworzenia oprogramowania. Gwarancję na produkt można próbować zastąpić gwarancją najwyższej jakości procedur stosowanych przy jego produkcji. Oprócz uniwersalnych norm ISO 9000 określających standardy zarządzania jakością, istnieją przynajmniej dwa inne – specyficzne dla oprogramowania – standardy certyfikacji jakości – *Common Criteria for Information Technology Security Evaluation* to obowiązująca norma ISO określająca sposób oceniania jakości oprogramowania. Natomiast *Software Considerations in Airborne Systems and Equipment Certification* to standard, któremu musi podlegać oprogramowanie samolotów dopuszczonych do lotów przez europejskie i amerykańskie agencje lotnictwa cywilnego.

Quality Assurance. Producenci oprogramowania mogą również inwestować swoje środki w budowę i utrzymanie własnych zespołów dbających o kontrolę jakości oprogramowania. Firma, która jest zainteresowana podniesieniem jakości swojego oprogramowania może zainwestować w narzędzia ułatwiające zarządzanie i analizę kodu źródłowego. Kilka firm, które dostarczają tego typu produkty, to:

AbsInt to firma założona przez pracowników Uniwersytetu Kraju Saary w Saarbrücken specjalizująca się w wykorzystywaniu technik kompilacji do analizy statycznej programów komputerowych. Obecnie zatrudnia 38 osób, a w swoim portfolio ma takich klientów jak Airbus, BMW, Bosch, HP, IBM, Intel, Nokia, Mercedes-Benz i Volkswagen.

Coverity to firma założona w 2002 roku przez naukowców z Uniwersytetu Stanforda. Zatrudnia 120 pracowników a wśród swoich klientów ma koncerny takie jak Philips, Research In Motion, Samsung i UBS.

Fortify istnieje na rynku od 2003 roku i specjalizuje się w analizie oprogramowania pod kątem bezpieczeństwa. Klientami Fortify są, między innymi, ABN Amro, GAP, Lockheed Martin, JPMorgan i Oracle.

Parasoft to firma założona przez byłych pracowników California Institute of Technology. Początkowo zajmowała się technologiami przetwarzania równoległego. Dziś sprzedaje oprogramowanie statycznej i dynamicznej analizy programów. Do klientów Parasoft należą AT&T, Bank of America, Boeing, Cisco i Lehman Brothers.

Rosnąca liczba wysokiej jakości powszechnie dostępnych i darmowych narzędzi do lokalizacji błędów i automatyzacji testów będzie wypy-

chać te firmy w kierunku oferowania usług zamiast licencjonowania programów.

2.2.3. Groźba nowych wejść

Bezpośrednie bariery wejścia na rynki związane z oprogramowaniem są dosyć niskie. Brak jest konieczności wyposażenia firmy w drogi sprzęt, a programiści są kształceni w prawie wszystkich krajach świata. Dzięki Internetowi dostęp do wiedzy jest powszechny, a część narzędzi jest dostępna zupełnie za darmo. Właściwie każda z firm wymienionych w punkcie 2.2.2 dysponuje odpowiednim *know-how* wymaganym do rozpoczęcia tego typu działalności.

2.2.4. Siła przetargowa nabywców

Większa niezawodność i bezpieczeństwo zweryfikowanego oprogramowania przekłada się bezpośrednio na wygodę i oszczędności użytkowników – zarówno indywidualnych konsumentów, jak i podmiotów zinstytucjonalizowanych, zamawiających programy dostosowane do ich potrzeb. Ale potencjalnymi odbiorcami usług dowodzenia poprawności (i ich pośrednimi beneficjentami) są firmy software'owe. Wynika to z konieczności ścisłej współpracy pomiędzy programistami rozwijającymi zweryfikowany produkt oraz analitykami przygotowującymi dowód jego zgodności z wymaganiami. Według danych czasopisma *Software Magazine*, 500 największych firm tworzących oprogramowanie przyniosło w roku 2008 w sumie 451,8 miliarda dolarów dochodu, o 14,7% więcej w porównaniu z rokiem 2007[19]. Nie sposób jednak oszacować, jak dużą część ich budżetów stanowiły wydatki na poprawienie jakości. Brak tych danych oraz brak podmiotów realizujących usługi weryfikacji oprogramowania uniemożliwia wiarygodne oszacowanie wartości potencjalnego rynku.

Należy również zwrócić uwagę na fakt, że dostęp do kodu źródłowego, wymaganego w procesie weryfikacji, może okazać się niemożliwy. Szczególnie w przypadku tych spółek, które swój kod źródłowy traktują jako *własność intelektualną* i zobligowane są prawnie bądź przez udziałowców do jego strzeżenia.

2.2.5. Siła przetargowa dostawców

Weryfikacja oprogramowania, podobnie jak testowanie produktów i usług, jest funkcją zaufania społecznego. Przeciętny użytkownik nie będzie w stanie samodzielnie sprawdzić, czy dowód zgodności systemu z założeniami jest poprawny. Z tego powodu, należy zapewnić jak największą prostotę



Rysunek 2.3: Diagram analizy SWOT

i przejrzystość samego procesu. Wymaga to oparcia się na powszechnie dostępnych programach o niekwestionowanej jakości, tak, aby jak największa ilość detali mogła zostać skontrolowana przez osoby trzecie.

2.3. Analiza SWOT

Analiza SWOT jest jednym z najpopularniejszych narzędzi do oceny wewnętrznego i zewnętrznego środowiska danej organizacji, projektu bądź produktu. Po raz pierwszy została zastosowana w latach sześćdziesiątych ubiegłego stulecia przez profesora Alberta Humphrey'a z Uniwersytetu Stanforda na podstawie danych udostępnianych przez firmy z indeksu Fortune 500. Polega ona na podzieleniu czynników strategicznych na cztery grupy przedstawione schematycznie na rysunku 2.3:

1. **Strengths** (*mocne strony*) – to wszystkie te cechy organizacji bądź produktu, które wspomagają osiągnięcie celu.
2. **Weaknesses** (*słabe strony*) – to cechy organizacji bądź produktu, które utrudniają osiągnięcie celu.
3. **Opportunities** (*szanse*) – to zewnętrzne czynniki, które mogą być przydatne do osiągnięcia celu.

4. **Threats** (*zagrożenia*) – to zewnętrzne czynniki, które mogą stanowić przeszkodę w osiągnięciu celu.

W kolejnych sekcjach zostały omówione wszystkie powyższe kategorie w odniesieniu do usługi weryfikacji oprogramowania komputerowego.

2.3.1. Mocne strony

- Wynikiem formalnej weryfikacji programu jest matematyczna gwarancja zachowania wymagań podczas wykonania. W przeciwieństwie do certyfikacji systemu kontroli jakości, potwierdza ona nie tylko zgodność procesu tworzenia oprogramowania z międzynarodowymi standardami, ale także dotyczy efektu końcowego tego procesu.
- Powstały w wyniku weryfikacji dowód zgodności programu z zadanymi wymaganiami jest dla danej wersji programu wieczny. Nie wymaga cyklicznego odnawiania ani nie wygasa.
- W trakcie procesu weryfikacji jest możliwa lokalizacja błędów w nieprzetestowanych, bądź niemożliwych do przetestowania scenariuszach wykonania.
- Dowód zgodności programu z wymaganiami jest bardzo silnym wyróżnikiem na tle konkurencji, szczególnie tej oferującej produkty o niższej cenie.

2.3.2. Słabe strony

- Powodzenie weryfikacji zależy w dużej mierze od właściwego doboru wymagań, które muszą być bardzo precyzyjnie określone.
- Proces weryfikacji wymaga bardzo dużego nakładu pracy i jest niezwykle czasochłonny.
- Przeprowadzenie dowodu poprawności programu bez dostępu do kodu źródłowego weryfikowanego systemu jest bardzo utrudnione, a w niektórych przypadkach wręcz niemożliwe.

2.3.3. Szanse

- Twierdzenia Turinga i Rice'a gwarantują, że procesu weryfikacji nie będzie można nigdy w pełni zautomatyzować. Oznacza to, że usługi polegającej na dostarczeniu dowodu zgodności programu z wymaganiami nigdy nie będzie można zastąpić programem komputerowym, który dowód taki przeprowadzi automatycznie.

- Upowszechnianie się oprogramowania z otwartym kodem źródłowym i stale rosnąca konkurencja ze strony darmowych zamienników będą zmuszać firmy software'owe do poszukiwania takich atutów swoich produktów, które nie będą łatwe do skopiowania.
- Ponadto, rosnąca popularność oprogramowania z dostępnym kodem źródłowym ułatwi rozpoczęcie działalności poprzez obniżenie kosztów znalezienia pierwszego klienta oraz poprzez zwiększoną dostępność wysokiej jakości darmowych podsystemów niezbędnych w procesie weryfikacji.
- Rynek oprogramowania jest obecnie nieuregulowany. Sytuacja ta jednak z całą pewnością ulegnie zmianie i producenci oprogramowania będą musieli przyjmować na siebie odpowiedzialność za swoje produkty. Dowód poprawności może znacząco obniżyć ryzyko poniesienia odpowiedzialności za szkody wyrządzone przez program i, w ten sposób, obniżyć również koszt zabezpieczenia się przed taką ewentualnością.
- Brak konkurencji na rynku stwarza okazję do zajęcia dominującej pozycji pierwszemu graczowi oraz na monopolizację tego sektora usług.
- Rosnąca świadomość użytkowników skłoni ich do poszukiwania niezależnych metod porównywania jakości oprogramowania.

2.3.4. Zagrożenia

- Stale zwiększająca się liczba narzędzi i technologii utrudnia analizę programów pod kątem formalnym ze względu na poziom komplikacji dostarczanych produktów. Wszelkie stosowane narzędzia muszą być odpowiednio dostosowywane pod kątem choćby nowych języków oprogramowania.
- Niskie bariery wejścia na rynek powodują, że konkurencja może pojawić się stosunkowo szybko.
- Ustawodawcy mogą wykluczyć niektóre podmioty z udziału w rynku usług związanych z zapewnieniem wysokiego standardu oprogramowania poprzez np. wprowadzenie koncesji i nieudzielanie jej podmiotom zagranicznym w celu ochrony krajowej własności intelektualnej przed szpiegostwem przemysłowym.

2.4. Podsumowanie

Obecnie brak jest konkurencji na globalnym rynku usług formalnej weryfikacji oprogramowania. Nie sposób jest, w związku z tym, ocenić wartości rynku. Firmy software'owe to jedyne podmioty, które mogą być zainteresowane finansowaniem usługi dowodzenia poprawności, ale nie można ocenić, jak duża część ich budżetów jest przeznaczona na zapewnienie wysokiej jakości tworzonych produktów.

Pomimo niskiego nasycenia rynku akurat tym segmentem działalności, istnieje co najmniej kilka substytutów, które należy już dziś traktować jako ważnych uczestników gry rynkowej. Z jednej strony na rynku tym brakuje silnego gracza, z którym konsumenci mogliby kojarzyć tego typu usługi, z drugiej jednak strony charakteryzuje się on niskimi barierami wejścia, co oznacza, że ewentualna konkurencja może pojawić się wkrótce po osiągnięciu pierwszych sukcesów komercyjnych. W następnym rozdziale przedstawiono strategię wejścia na ten obiecujący rynek i zajęcia na nim dominującej pozycji.

Rozdział 3

Strategia

Strategia marketingowa stanowi wybór celów, rodzajów zasad czy reguł, które w określonym czasie nadają kierunek marketingowym działaniom przedsiębiorstwa, wyznaczając rozmiary, kombinacje i alokację środków w zależności od zmieniającej się sytuacji rynkowej.

Philip Kotler

W niniejszym rozdziale została opisana strategia przedsiębiorstwa wprowadzającego na rynek usługę formalnej weryfikacji oprogramowania komputerowego. Rozdział rozpoczyna się od wyodrębnienia segmentów potencjalnych klientów w sekcji 3.1. Na podstawie tych informacji zostało opracowane pozycjonowanie usług weryfikacji w sekcji 3.2. Sekcja 3.3 omawia misję, wartości oraz cele do realizacji. W sekcji 3.4 została przedstawiona kompozycja instrumentów i działań marketingowych dostosowanych do opisanych zamierzeń.

3.1. Segmentacja

Chociaż producenci oprogramowania nie czerpią bezpośrednich zysków z wysokiej niezawodności swoich produktów, są jej pośrednimi beneficjentami i potencjalnymi odbiorcami usług weryfikacji. Oprogramowanie komputerowe może być podzielone według kilku różnych kryteriów:

- Pierwszy podział to stopień przygotowania sprzedawanej aplikacji. Na jednym biegunie tego podziału znajdują się programy pisane na zamówienie odbiorcy. Nie istnieją one w chwili składania zamówienia, są

tworzone w oparciu o analizę wymagań klienta i dostosowane do jego indywidualnych wymagań. Na drugim biegunie są gotowe programy rozprowadzane w takiej samej formie do wszystkich użytkowników, na przykład gry komputerowe, których każdy użytkownik otrzymuje dokładnie taką samą kopię, co wszyscy pozostali. Pomędzy tymi dwiema skrajnościami istnieje szerokie spektrum programów integrowanych z osobnych modułów jak również software indywidualizowany zgodnie z wymaganiami i możliwościami finansowymi klientów.

- Drugim często stosowanym kryterium jest podział branżowy, uwzględniający docelowego klienta oprogramowania. Często wyodrębniane gałęzie to:

Consumers to ta część rynku, do której docierają najbardziej powszechne produkty. Obejmuje zarówno sektor oprogramowania dla urządzeń elektroniki użytkowej jak również systemy operacyjne i pakiety biurowe.

SME (*Small & Medium Enterprises*, w tym **SOHO** – *Small Office/Home Office*) to sektor mikro, małych i średnich przedsiębiorstw. Do takich firm kieruje się przede wszystkim zintegrowane pakiety do zarządzania firmą obejmujące zarządzanie kontrahentami, prowadzenie księgowości i kontrolowanie stanu rozliczeń z klientami i dostawcami.

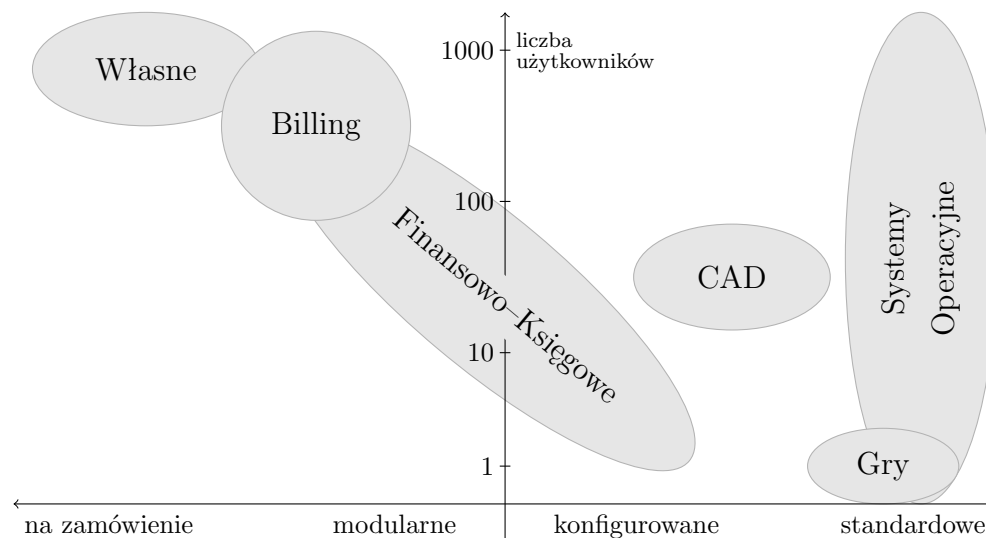
M&D (*Manufacturing & Distribution*) to głównie systemy klasy ERP do wspomaganie operacji logistycznych w firmach produkcyjno-handlowych.

IT&Telecom obejmuje przede wszystkim systemy do zarządzania sieciami telekomunikacyjnymi oraz taryfikacji i automatycznego fakturowania klientów.

Financial to oprogramowanie dla branży finansowej obejmujące systemy transakcyjne, oprogramowanie wspierające bankowość korporacyjną i detaliczną oraz pakiety do zarządzania ryzykiem. Do branży tej zaliczyć należałoby również systemy lojalnościowe, choć ich tradycyjnymi odbiorcami są przedsiębiorstwa z branży handlu detalicznego, linie lotnicze oraz sieci sprzedaży paliw.

CAD (*Computer Aided Design*) czyli oprogramowanie wspierające firmy zajmujące się projektowaniem wszelkiego rodzaju dóbr materialnych – od układów elektronicznych, przez środki transportu aż po budynki.

Components to narzędzia deweloperskie i gotowe komponenty systemów komputerowych. Do kategorii tej zaliczają się między in-



Rysunek 3.1: Podział oprogramowania komputerowego według stopnia przygotowania oraz ilości użytkowników.

nymi systemy zarządzania bazami danych, serwery pośredniczące i frameworki.

Lista ta pomija wiele innych dziedzin życia gospodarczego, jak firmy farmaceutyczne, budowlane, petrochemiczne, aeronautyczne a także administrację publiczną. Wynika to z dwóch powodów: przygotowanie wyczerpującego wykazu branż wykracza poza zakres tej pracy, a większość firm i tak korzysta z szeroko dostępnego oprogramowania konsumenckiego.

Na rysunku 3.1 została zobrazowana segmentacja kilku rodzajów programów w zależności od formy, w jakiej są dystrybuowane, oraz od wielkości organizacji wykorzystującej dany system.

3.2. Pozycjonowanie

Weryfikacja oprogramowania jest przede wszystkim metodą na obniżenie ryzyka następstwa nieprzewidzianych usterek w dystrybuowanym oprogramowaniu. Aby inwestycja w taką usługę miała szansę się zwrócić, produkt, dla którego przeprowadzany jest dowód, musi spełniać jeden z następujących warunków:

1. Cykl życia produktu jest długi. Długie pozostawanie systemu na rynku zwiększa prawdopodobieństwo wystąpienia błędu o poważnych skut-

kach. W najlepszym przypadku będzie to tylko utrata reputacji, w najgorszym – konsekwencje finansowe.

2. Liczba użytkowników programu jest duża. Podobnie, jak w punkcie 1, liczba użytkowników jest wprost proporcjonalna do prawdopodobieństwa wykrycia istotnego błędu. Oprogramowanie skierowane na rynek konsumencki jest dodatkowo narażone na eskalację trudnej do opanowania złości użytkowników.
3. Program spełnia krytyczną rolę dla powodzenia przedsięwzięcia lub zapewnia bezpieczeństwo ludzkiego życia. Na przykład, od oprogramowania kontrolującego aparaturę medyczną, systemy nawigacji lotniczej, czy wspomagania kierowcy (np. ABS) może zależeć ludzkie życie, stąd konieczność zapewnienia jego jak najwyższej jakości.

Duży nakład pracy i związany z tym długi czas wykonywania usługi ogranicza grupę potencjalnych klientów do ugruntowanych na rynku producentów oprogramowania, dla których *time-to-market* nie jest decydującym czynnikiem przy podejmowaniu decyzji oraz których cykl życia produktu jest stosunkowo długi. Tylko dla takich produktów istnieje bowiem szansa na zwrot z inwestycji w weryfikację oprogramowania.

3.3. Plan strategiczny

Zaprezentowany poniżej plan strategiczny składa się z trzech zasadniczych części: misji, wartości oraz celów.

3.3.1. Misja

O ile misję tworzy się zwyczajowo dla konkretnej organizacji, w celu określenia jej przedmiotu aspiracji oraz wyróżnika wśród konkurencji, o tyle w przypadku usługi dowodzenia poprawności programów komputerowych jej misję ujmijemy w sposób zaczerpnięty z oświadczenia misyjnego Euro NCAP:

Być katalizatorem zmian na rzecz bezpieczniejszego i bardziej niezawodnego oprogramowania.

Sensem weryfikacji oprogramowania jest stałe poprawianie jakości istniejących programów oraz działanie na rzecz promocji technik wytwarzania bardziej niezawodnego i bezpiecznego software'u. Do zadań organizacji pełniącej tę misję muszą należeć:

1. Formalna weryfikacja krytycznych elementów oprogramowania, w tym najczęściej stosowanych bibliotek podprocedur.
2. Zaangażowanie w tworzenie najlepszych praktyk i promowanie technik gwarantujących najwyższy poziom jakości.
3. Działanie na rzecz budowania wśród konsumentów świadomości jakości oprogramowania.

3.3.2. Wartości

Weryfikacja oprogramowania, podobnie jak testowanie produktów i usług, jest funkcją zaufania społecznego. Reputacja organizacji przygotowującej dowody zgodności programów z wymaganiami jest zatem jej najcenniejszym aktywem. Zbudowanie odpowiedniej renomy wymaga postępowania w zgodzie z następującymi trzema wartościami:

Przejrzystość. Wszystkie informacje dotyczące zarówno metodologii jak i narzędzi służących do otrzymania dowodu będą ujawnione w najdrobniejszych szczegółach, tak, aby umożliwić środowisku naukowemu na całym świecie powtórzenie tych samych wyników oraz wskazanie ewentualnych uchybień.

Obiektywizm. Weryfikacja oprogramowania będzie opierać się wyłącznie na niebudzących wątpliwości badaniach naukowych, które zostały potwierdzone w niezależnych od siebie źródłach.

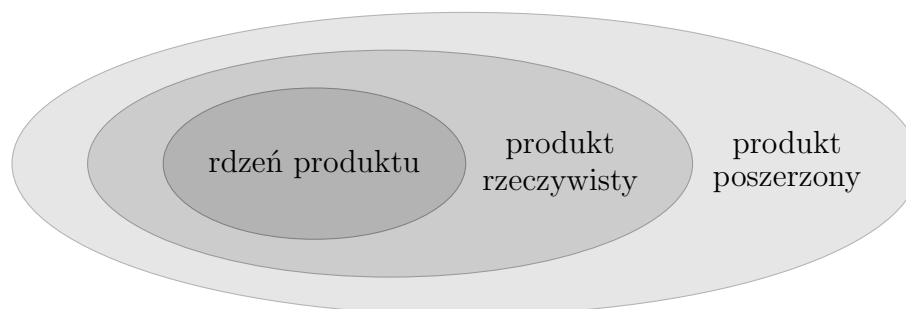
Odpowiedzialność. Nie będą stosowane metodologie, które mogą prowadzić do błędów typu I. To znaczy, że żaden program, który nie spełnia postawionych założeń, nie będzie pozytywnie zweryfikowany.

Wartości te przyznają nadrzędną rolę interesom konsumenta nad interesami twórców oprogramowania.

3.3.3. Cele

Ponieważ usługa weryfikacji nie istnieje jeszcze na rynku, najważniejszymi zadaniami stojącymi przed organizacją rozpoczynającą działalność w tym sektorze jest:

- Budowa zespołu analityków zdolnych do wydajnego dowodzenia zgodności programów z wymaganiami oraz zdolnymi do współpracy z klientem w ramach uzgadniania tych wymagań.



Rysunek 3.2: Koncepcja budowy produktu wg. prof. Theodore Levitta.

- Zdominowanie terminologii weryfikacyjnej tak, aby dowodzenie poprawności programów było jednoznacznie kojarzone z firmą realizującą te usługi.

W następnej sekcji zostanie zdefiniowany wachlarz narzędzi marketingowych, które pozwolą zadebiutować produktowi na rynku zgodnie z założoną misją.

3.4. Kompozycja marketingowa

Aby zrealizować tak postawione cele, należy dobrać odpowiednią paletę instrumentów marketingowych, które zapewnią realizację zadań taktycznych. W niniejszej sekcji została zastosowana metoda tzw. czterech P:

Product – produkt i jego cechy.

Price – cena, rabaty i warunki płatności.

Place – strategia dystrybucji.

Promotion – ogół działań promocyjnych produktu.

Poniżej zostały omówione wszystkie cztery elementy z uwzględnieniem specyfiki usługi dowodzenia zgodności programów z wymaganiami.

3.4.1. Produkt

Zgodnie z koncepcją prof. Theodore Levitta, każdy produkt składa się z trzech poziomów zilustrowanych na rysunku 3.2:

Rdzeń produktu to podstawowe cechy samego produktu bądź usługi, w tym cechy funkcjonalne i właściwości użytkowe.

Rdzeniem usługi jest sam dowód poprawności. Dostarczany jest wraz ze szczegółowym opisem formatu zapisu, metodologii weryfikacji oraz z narzędziami automatyzującymi kontrolę dowodu.

Produkt rzeczywisty obejmuje wszystko to, co ma wpływ na postrzeganie produktu, tj. opakowanie, marka, materiał, cena i jakość, ale również obsługa personelu.

W przypadku formalnej weryfikacji oprogramowania sprzedawanym produktem jest nie tylko sam dowód, ale cały proces jego powstawania, w trakcie którego uściślone są wymagania oraz usuwane są błędy zarówno na poziomie projektu jak i implementacji.

Do cech produktu rzeczywistego zaliczamy również pakiet dodatkowych usług, obejmujących aktualizację dowodów dla kolejnych wersji oprogramowania, szkolenia dla programistów oraz całodobową obsługę sytuacji kryzysowych.

Produkt poszerzony zawiera w sobie obydwa poprzednie poziomy wraz ze wszystkimi dodatkowymi korzyściami dostarczanymi konsumentowi takimi jak dostawa, naprawy i konserwacja oraz usługi posprzedażowe.

Wyższa jakość zweryfikowanego programu i potwierdzenie zgodności z wymogami to nie wszystkie korzyści, jakie otrzymuje odbiorca usługi. Firma angażująca się w działania na rzecz poprawy jakości swoich produktów i bezpieczeństwa swoich użytkowników, jest lepiej postrzegana przez konsumentów, a przynależność do ekskluzywnego klubu producentów najwyższej jakości oprogramowania może się przekładać na lepsze wyniki sprzedaży.

3.4.2. Cena

Ponieważ obecnie żadna firma nie realizuje usług dowodzenia poprawności programów, brak jest doświadczenia koniecznego do oszacowania czasochłonności takiego projektu. W tej sytuacji, najrozsądniejszą polityką cenową jest opłata ryczałtowa za dzień pracy konsultanta.

3.4.3. Dystrybucja

Rozwój globalnych sieci telekomunikacyjnych umożliwia realizację usługi z dowolnego miejsca na ziemi. Jedyne obostrzenia mogą być związane z dostępem do kodu źródłowego projektów, które objęte są ograniczeniami eksportowymi i nie mogą być kopiowane na dyski komputerów znajdujących

się poza krajem, w którym zostały stworzone. Jednak szybki rozwój wirtualizacji i coraz większa przepustowość łączy telekomunikacyjnych umożliwia zdalną pracę również na komputerach znajdujących się „po drugiej stronie globu”.

3.4.4. Promocja

Promocja usługi dopiero wchodzącej na rynek jest niezwykle istotnym elementem strategii marketingowej, dlatego też na nią położony został największy nacisk. W przypadku weryfikacji oprogramowania promocja tej usługi wśród konsumentów służyć ma wywarceniu nacisku na producentach oprogramowania, aby dowodzili poprawności swoich programów. Poniżej zaprezentowane zostały najważniejsze elementy zapewniające osiągnięcie tego celu.

- Obecnie nie istnieją jeszcze żadne standardy zapisu dowodów poprawności oprogramowania. Normę taką należy stworzyć, udokumentować i opublikować. Jedną z najlepszych platform do podzielenia się taką informacją z resztą świata są konferencje naukowe związane z inżynierią oprogramowania.
- Przygotowanie dowodów poprawności jednego z popularnych programów, którego kod źródłowy jest publicznie dostępny (lista kilku wybranych pakietów wraz z dodatkowymi informacjami została zebrana w tabeli 3.1).
- Należy założyć, że podczas weryfikacji jednego z systemów uda się znaleźć istotne błędy bądź przeoczenia. Należy opublikować je na istniejącej od 1993 roku branżowej liście dyskusyjnej *BugTraq*, która zbiera tego typu informacje. Subskrybowana jest ona przez praktyków bezpieczeństwa oprogramowania na całym świecie. W przypadku błędów zgłaszanych przez innych uczestników dyskusji, należy również, w miarę możliwości, wskazywać, w jaki sposób zastosowanie formalnej weryfikacji oprogramowania mogło zapobiec wystąpieniu danej klasy problemów.
- Stworzenie strony internetowej z listą zweryfikowanych programów oraz narzędziami umożliwiającymi konsumentom sprawdzenie, czy zainstalowane przez nich wersje programów posiadają dowód zgodności z wymaganiami. Na stronie internetowej powinny się również znajdować materiały edukacyjne i szkoleniowe.
- Przygotowanie zastrzeżonego logotypu, którym producenci zweryfikowanych programów mogliby posługiwać się w celu promocji swoich systemów.

Linux:	System operacyjny ogólnego zastosowania
Udział w rynku:	od 1 do 5% w segmencie desktop, ok. 60% w segmencie serwerów
Kod źródłowy:	poza nielicznymi sterownikami w pełni dostępny
Kontrola:	Linux Foundation
Solaris:	Serwerowy system operacyjny
Udział w rynku:	ok. 9%
Kod źródłowy:	częściowo dostępny jako OpenSolaris
Kontrola:	Sun Microsystem (Oracle)
Android:	System operacyjny dla urządzeń przenośnych
Udział w rynku:	4%–6%
Kod źródłowy:	większość kodu dostępna publicznie
Kontrola:	Open Handset Alliance (Google)
Symbian:	System operacyjny dla urządzeń przenośnych
Udział w rynku:	ponad 45%
Kod źródłowy:	dostępny dla partnerów, plany pełnego upublicznienia w pierwszej połowie 2009
Kontrola:	Symbian (Nokia)
Apache:	Serwer HTTP
Udział w rynku:	ok. 60%
Kod źródłowy:	publiczny
Kontrola:	Apache Foundation
MySQL:	System zarządzania bazą danych
Udział w rynku:	ok. 25%
Kod źródłowy:	publiczny
Kontrola:	Sun Microsystem (Oracle)
SQLite:	System zarządzania bazą danych
Udział w rynku:	ok. 400 milionów kopii
Kod źródłowy:	publiczny
Kontrola:	Richard Hipp

Tablica 3.1: Lista wybranych popularnych pakietów oprogramowania z dostępnym kodem źródłowym.

- W dalszej perspektywie należałoby dążyć do rozszerzenia powszechnie obowiązujących norm jakościowych oprogramowania o dowodzenie ich poprawności. Do najważniejszych z nich należą:
 - *Common Criteria for Information Technology Security Evaluation* to obowiązujący standard ISO określający sposób oceniania jakości oprogramowania. Obecnie, nawet najwyższy poziom *Evaluation Assurance Level* (którego nie osiągnął nawet system operacyjny F-16) nie wymaga tego, aby kod źródłowy podlegał formalnej weryfikacji.
 - *Software Considerations in Airborne Systems and Equipment Certification* to standard, któremu musi podlegać oprogramowanie samolotów dopuszczonych do lotów przez europejskie i amerykańskie agencje lotnictwa cywilnego. Certyfikacja nie wymaga przeprowadzenia dowodu poprawności, a jedynie przygotowania zestawu testów, których wykonanie nie omija żadnego wiersza programu.

3.5. Monitoring ryzyka

Zarządzanie ryzykiem jest jedną z kluczowych funkcji administracji każdym przedsięwzięciem. Obejmuje ono szacowanie wartości parametru ryzyka oraz potencjalnych kosztów związanych z wystąpieniem niechcianego zdarzenia. Informacja taka pozwala na przeprowadzenie odpowiednich działań zapobiegawczych oraz na przygotowanie planów minimalizacji strat w sytuacji, gdyby, mimo wszystko, wystąpiło. Dla każdego ze zweryfikowanych projektów należy oszacować i stale aktualizować prawdopodobieństwo wystąpienia dwóch następujących zdarzeń:

1. Dowód poprawności może okazać się błędny. Jeśli błąd zostanie odkryty w metodologii weryfikacji, należy:
 - Przyznać się do błędu.
 - Oszacować skalę problemu i poprawić wszystkie dowody, które zostały dotknięte tym problemem.
 - Zadośćuczynić poszkodowanym.
 - Opracować plan, który nie dopuści do powtórzenia się tej klasy problemów w przyszłości.
2. W jednym ze zweryfikowanych programów zostanie znaleziony poważny błąd, który nie został objęty formalnymi wymaganiami. W tej

sytuacji należy skoordynować działania wraz z producentem dotkniętego systemu w celu jak najszybszego uzupełnienia dowodu. Dobrze przeprowadzona akcja może pozwolić na zacieśnienie relacji z klientem.

Tak opracowany plan postępowania w sytuacjach kryzysowych należy stale uzupełniać.

3.6. Podsumowanie

W rozdziale tym opisana została strategia wprowadzenia na rynek usługi formalnej weryfikacji oprogramowania. Opiera się ona na rzetelnym i uczciwym informowaniu zarówno konsumentów, jak i producentów oprogramowania o zagrożeniach związanych z błędami występującymi w programach komputerowych oraz na aktywnym przeciwdziałaniu i reagowaniu na zaistniałe sytuacje kryzysowe. Wszystkie działania, wykonywane w ramach przygotowanego planu, muszą być stale monitorowane, a sam plan dopasowywany do zmieniających się warunków panujących na rynku.

Bibliografia

- [1] Edsger W. Dijkstra. On the cruelty of really teaching computing science, XII 1988. W obiegu nieformalnym.
- [2] Gerard J. Holzmann i Margaret H. Smith. Automating software feature verification. *Bell Labs Technical Journal*, 5:72–87, 2000.
- [3] Francis Schneider, Steve M. Easterbrook, John R. Callahan, i Gerard J. Holzmann. Validating requirements for fault tolerant systems using model checking. W *Third IEEE Conference on Requirements Engineering*, strony 4–13. IEEE Computer Society, 1998.
- [4] Microsoft Corp. Static Driver Verifier.
- [5] Andrzej Wasylkowski, Andreas Zeller, i Christian Lindig. Detecting object usage anomalies. W *ESEC-FSE '07: Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, strony 35–44. ACM, New York, NY, USA, 2007.
- [6] George Shiffler. Forecast: PC Installed Base, Worldwide, 2004-2012. Raport techniczny, Gartner Inc., 2008.
- [7] BusinessWeek. The Web According to Ballmer, 2006.
- [8] Laurie F. Wurster. User Survey Analysis: Open-Source Software, Worldwide, 2008. Raport techniczny, Gartner Inc., 2008.
- [9] Bruce Schneier. Software makers should take responsibility. *The Guardian*, 2008.
- [10] Mark Methenitis. FTC could target EULAs, 2009.
- [11] Komisja Europejska. Prawa konsumenta: Komisja pragnie, aby konsumenci mogli korzystać z Internetu bez granic, 2009.

- [12] Michael Newman. Software errors cost U.S. economy \$59.5 billion annually. Raport techniczny, National Institute of Standards and Technology, Department of Commerce, 2002.
- [13] Edsger W. Dijkstra. Chapter I: Notes on structured programming. strony 1–82, 1972.
- [14] Mark Dowson. The Ariane 5 software failure. *SIGSOFT Softw. Eng. Notes*, 22(2):84, 1997.
- [15] Peter G. Neumann. Risks to the public in computers and related systems. *SIGSOFT Softw. Eng. Notes*, 29(5):13–18, 2004. ISSN 0163-5948.
- [16] John Herrman. 30GB Zunes Failing Everywhere, All At Once, 2008.
- [17] Michael E. Porter. *Strategia konkurencji. Metody analizy sektorów i konkurentów*. PWE, Warszawa, 1992.
- [18] Verisoft. Verisoft XT, Zusammenfassung, 2007.
- [19] John P. Desmond. Innovation Alive and Well. *Software Magazine*, 2008.