Paweł Zarzycki

Udostępnianie oprogramowania w modelu open source w branży elektroniki użytkowej

Zarządzanie Zmianami : zeszyty naukowe nr 4, 20-38

2010

Paweł Zarzycki*

Udostępnianie oprogramowania w modelu open source w branży elektroniki użytkowej

Streszczenie

W artykule omawiam problematykę udostępniania oprogramowania w modelu Open Source Software (OSS) jako narzędzia strategii biznesowej przedsiębiorstwa. Ważne jest pytanie, czy publiczne udostępnianie własności intelektualnej przedsiębiorstwa z branży elektroniki konsumenckiej może pozytywnie wpłynąć na jego przewagę konkurencyjną?
Doświadczenia firm z sektora ICT pozwalają postawić tezę, że udostępnienie elementów oprogramowania w modelu OSS pozwoli firmie uzyskać przewagę konkurencyjną. Istotne jest przy tym przyjęcie odpowiedniej strategii biznesowej.

W pracy przyjąłem indukcyjno-dedukcyjną metodę badań. Przeanalizowałem aktualny stan wiedzy teoretycznej, proponowane modele i teorie, aby dokonać próby ich ekstrapolacji do branży elektroniki konsumenckiej. Dokonałem również systematycznej analizy ważnych firm ICT, o różnych modelach biznesowych, które wykorzystują OSS. Analizy te pokazały, że:

- niezbędnym elementem skutecznego wykorzystania OSS jest zabezpieczenie istotnych źródeł przewagi konkurencyjnej;
- analizowany model ma szczególne zastosowanie w tych gałęziach przemysłu, w których istotne znaczenie ma innowacyjność.

Wskazałem również kluczowe czynniki sukcesu wpływające na przewagę konkurencyjną firmy.

Słowa kluczowe: open source, strategia, prawo autorskie, wartość niematerialna, model biznesowy.

Wstęp

Od kilkunastu lat obserwujemy gwałtowny rozwój oprogramowania w modelu open source (OSS) oraz społeczności tworzącej się wokół tego modelu. Idee i rozwiązania organizacyjne, pierwotnie powstałe na gruncie społeczności open source związanej z oprogramowaniem, inkorporowane są w innych dziedzinach twórczości intelektualnej, jak literatura, sztuka czy edukacja. Katalizatorem tego rozwoju jest ekspansja Internetu oraz szybki postęp w technologiach informatycznych. Jednocześnie obserwujemy, jak coraz więcej firm eksperymentuje z wykorzystaniem open source i często z powodzeniem wypracowuje efektywne modele biznesowe. Choć uznaje się, że ruch open source z założenia nie pracuje dla zysku, czy wręcz działa przeciw komercjalizacji know-how, to, jak pokazuje doświadczenie, może być w odpowiednich warunkach wykorzystany do budowania silnej pozycji rynkowej. W ostatniej de-

* Mgr/MBA Paweł Zarzycki — Cognitum, e-mail: p.zarzycki@versamind.com.

kadzie tym tematem zainteresowało się również wielu autorów akademickich. Poszukuje się nowych modeli teoretycznych, które pozwolą lepiej rozumieć i analizować obserwowalne zjawiska na tym polu, zarówno z dziedziny socjologii, teorii organizacji, ekonomii, jak i inżynierii oprogramowania. Jak zauważają różni badacze (por. Fitzgerald [2006], Chesbrough, Appleyard [2007]), modele i teorie sprawdzone w gospodarce opartej głównie na obrocie dobrami materialnymi zawodzą w analizie tworzących się nowych dziedzin gospodarki opartych przede wszystkim na własności intelektualnej i dobrach niematerialnych.

Branża elektroniki użytkowej (konsumenckiej) jest z jednej strony dziedziną gospodarki materialnej, jednak oprogramowanie (oraz towarzyszące mu inne wartości niematerialne) są istotnym elementem wpływającym na konkurencyjność w poszczególnych etapach łańcucha tworzonej wartości. Znaczenie oprogramowania stale rośnie wraz z dynamicznym postępem technologii, zwiększającymi się wymaganiami klientów i rozwojem rynków horyzontalnych. Pojawia się zatem pytanie, czy publiczne udostępnienie własności intelektualnej przedsiębiorstwa z branży elektroniki konsumenckiej może pozytywnie wpłynąć na jego przewagę konkurencyjną? Obiecujące doświadczenia firm z sektora ICT oraz postulaty wielu autorów, zarówno akademickich, jak i ze świata biznesu, pozwalają postawić tezę, że udostępnienie elementów oprogramowania w modelu open source pozwoli firmie uzyskać przewagę konkurencyjną. Istotne jest przy tym przyjęcie odpowiedniej strategii biznesowej przez przedsiębiorstwo, a strategia taka może być efektywnie wykorzystywana przez dowolne przedsiębiorstwo z branży.

Artykuł powstał na podstawie wyników badań w ramach mojej pracy dyplomowej MBA w Wyższej Szkole Zarządzania w Warszawie. W pracy tej przeanalizowałem strategie firm Microsoft, Google, LG oraz Samsung Electronics.

1. Oprogramowanie open source i jego historia

Oprogramowanie open source to prawdziwy fenomen ostatnich kilkunastu lat[1]. W wielkim skrócie open source to otwarte oprogramowanie, dostępne za darmo i dla każdego, tworzone bez wynagrodzenia przez społeczność programistów i użytkowników organizujących się w Internecie. Istotą takiego podejścia do wytwarzania oprogramowania jest jego pełna otwartość i dostępność, jednak z ważnym zastrzeżeniem — każde oprogramowanie pochodne (stworzone z wykorzystaniem produktu pierwotnego lub rozszerzające jego funkcjonalność) muszą być również w pełni otwarte i dostępne na tych samych zasadach. Nikt nie może również przywłaszczać sobie praw i sprzedawać wolnego oprogramowania (może za to sprzedawać dodatkowe usługi lub produkty).

Zastrzeżenie takie gwarantuje swobodny przepływ tworzonych wartości niematerialnych i dalszy rozwój wolnego

[1] Fenomen ten polega z jednej strony na zaangażowaniu kluczowych graczy na rynku IT w rozwój ruchu oprogramowania open source, z drugiej zaś na gwałtownym zainteresowaniu świata naukowego tym zjawiskiem. Jeszcze dziesięć lat temu nie było publikacji naukowych na temat open source, podczas gdy dziś wielu badaczy zajmuje się tym tematem na co dzień oraz prowadzone są projekty badawcze finansowane m.in. ze środków unijnych (por. Crowston i inni [2008], Giera [2004]).

oprogramowania dostępnego dla wszystkich użytkowników. Gwarantuje również, że nikt nie przejmie wolnego oprogramowania w sposób niezgodny z intencjami autora (dokładniej — autorowi przysługuje szereg roszczeń w przypadku naruszenia jego praw wynikających z prawnoautorskiej ochrony oraz warunków licencji).

Open source jest szczególnym przypadkiem całego segmentu produkcji oprogramowania. Wyróżnia je pozornie odwrotne podejście do zabezpieczania interesów i ochrony praw autorów. Pozorne, gdyż oprogramowanie open source mogło się rozwinąć właśnie dzięki dostępnym mechanizmom ochrony własności intelektualnej, w szczególności oprogramowania.

Historia open source zaczyna się na początku lat 80., kiedy to Richard Stallman założył Free Software Foundation. Stallman był idealistą — swoje oprogramowanie przekazywał razem z instrukcją: „Podziel się tym kodem z innymi użytkownikami. Naucz się na nim. Popraw go. A gdy skończysz, daj, proszę, coś z powrotem do naszej społeczności."[2]

Stallman i jego fundacja mają na swoim koncie takie projekty jak GNU[3], alternatywa dla systemu Unix, czy kompilator GCC[4]. Free Software Foundation stworzyła też pierwszą i jedną z najpopularniejszych licencji open source — GPL (GNU Public Licence). Na licencji tej udostępniany jest m.in. system Linux.

Licencja GNU GPL wymusza, aby każdy utwór pochodny był dystrybuowany tylko na takiej samej licencji. Określa się to jako efekt wirusowy. Jeśli w pewnym programie komputerowym znajdzie się komponent na licencji GNU GPL, to cały program musi być udostępniany na tej licencji.

Koncepcja Stallmana, została jakiś czas później zaadaptowana do obecnej postaci open source przez Erica Raymonda i Bruce'a Perensa. W 1998 r. założyli oni fundację Open Source Initiative (http://www.opensource.org/) oraz zaproponowali definicję oprogramowania otwartego — Open Source Definition. Dziś organizacja ta pełni główną rolę w promowaniu ruchu Open Source, m.in. certyfikuje licencje, które pretendują do miana open source. Obecnie (grudzień 2010 r.) OSI certyfikowała 67 licencji, wśród nich licencje takich firm, jak Microsoft, IBM, Lucent, NASA, Nokia czy Sun[5].

Na pierwszy rzut oka model open source wydaje się całkowicie odmienny od konwencjonalnej inżynierii oprogramowania. Na przykład w odróżnieniu od klasycznych teorii inżynierii oprogramowania dominuje w nim rozproszony model koordynacji, nie ma prawdziwego formalnego procesu projektowania, oceny ryzyka ani zdefiniowanych celów, nie ma bezpośredniej zachęty pieniężnej dla inżynierów ani nieformalnej koordynacji i kontroli [Fitzgerald 2005]. W istocie model ten dość dobrze odpowiada teorii inżynierii oprogramowania, co widać wyraźnie w tych projektach open source, które zyskały popularność i przyciągnęły do pracy zdolnych inżynierów. W projek-

[2] Cytat Stallmana zaczerpnięty z artykułu Erica Kidda dostępnego w Internecie: http://static.userland.com/userLandDiscussArchive/msg019844.html.
[3] GNU — System operacyjny, stanowiący darmową alternatywę dla systemu Unix (GNU — GNU's Not Unix).
[4] GCC — GNU Compiler Collection.
[5] Kompletna lista znajduje się na stronie: www.opensource.org/licenses/alphabetical.

tach tych można zauważyć dekompozycję systemu na moduły, staranną kontrolę wersji czy wzajemne sprawdzanie kodu przez programistów. Różnica polega na tym, że w podejściu open source te elementy niejako pojawiają się bez centralnego zarządzania czy ustalonych a priori procedur i procesów.

2. Własność intelektualna i jej ochrona

Pojęcie własności intelektualnej należy rozumieć przede wszystkim, jako ogół niematerialnych wytworów ludzkiego umysłu (intelektu), a także tych niematerialnych wytworów, które aby powstać, wymagały istotnych nakładów ich twórcy. Taką właśnie interpretację przyjmują współczesne prawne systemy ochrony własności intelektualnej. Istotą tej ochrony jest zabezpieczenie interesów twórcy dzieła bądź wynalazku (idei) lub inwestora, który poniósł ryzyko inwestycyjne związane ze stworzeniem dzieła, wobec potencjalnej konkurencji, która w przeciwnym razie łatwo mogłaby skopiować pomysł lub utwór, nie ponosząc żadnych nakładów na jego opracowanie (co wymaga dużej ilości pracy bądź szczególnych umiejętności)[6].

Ochrona własności intelektualnej ma zatem istotne znaczenie dla rozwoju technologicznego oraz cywilizacyjnego. Dzięki niej opłacalne stają się inwestycje w badania i rozwój czy tworzenie dzieł, które niezwykle łatwo jest kopiować i imitować[7]. Stanowi ona również prawne podstawy dla obrotu dziełami własności intelektualnej (np. przekazanie, sprzedaż, licencjonowanie).

Znaczenie własności intelektualnej we współczesnej gospodarce pokazują aktualne dane OECD[8]. Wydatki na badania i rozwój — czyli twórcze działania mające na celu podniesienie poziomu wiedzy, włączając w to wiedzę o człowieku, kulturze i społeczeństwie – zarówno badania podstawowe, jak i badania stosowane wynosiły w 2007 r. średnio 2,28% PKB w krajach OECD. Wydatki na oprogramowanie stanowią ponad połowę wydatków na ICT (łącznie oprogramowanie, sprzęt komputerowy i sprzęt telekomunikacyjny) w krajach rozwiniętych, takich jak USA, Szwecja, Finlandia, Francja[9].

We współczesnym świecie udział przemysłu opartego na własności intelektualnej jest ogromny. Dla przykładu w 2007 r. w USA firmy zajmujące się obrotem dobrami chronionymi prawem autorskim wygenerowały 1,52 bln dolarów (około 11,05% PKB)[10]. Dla porównania w tym samym roku w Korei Płd. udział tego segmentu rynku w PKB wynosił 8,67%, w Kanadzie ok. 4,5%, a w Rosji — ok. 6,06%[11]. Rynki oparte na własności intelektualnej stale się rozwijają.

Zagadnienie własności intelektualnej i jej ochrony jest bardzo złożone, dotyka często trudnych do zmierzenia elementów (np. prawnych zapisów o „istotnej części dzieła" czy „oczywistości rozwiązania"), ewoluowało na przestrzeni wieków

[6] Do lat 70. pojęcie ochrony własności intelektualnej utożsamiano głównie z ochroną praw autorskich oraz prawami pokrewnymi [Jacob 2004, s. 3].
[7] Łatwość ta jest de facto wynikiem takiego właśnie rozwoju technologii.
[8] Przytoczone dane pochodzą z raportu OECD Factbook 2010 [OECD 2010].
[9] Ibid.
[10] Dane pochodzą z raportu IIPA 2003-2007 [IPPA 2009].
[11] Dane pochodzą z WIPO Economic & Contribution Mapping [WIPO 2010].

wraz z rozwojem technologii. Widać to szczególnie wyraźnie w obszarze oprogramowania.

Ważnym zadaniem ochrony własności intelektualnej jest stymulowanie innowacyjności. Robert Blasi [2005] trafnie streścił problem, przed jakim staje każdy innowator:

- wymyśl produkt lub usługę,
- alokuj zasoby do jego opracowania i stworzenia,
- wyedukuj rynek,
- a na koniec, jeśli ci się powiodło, konkurencja skopiuje twój pomysł i zaoferuje go taniej.

Ten problem jest wyraźnie widoczny w przemyśle komputerowym, w którym kopiowanie jest technologicznie niezwykle tanie, proste i efektywne. Wymagana zatem jest właściwa ochrona. Ochrona programów komputerowych realizowana jest przede wszystkim na gruncie prawa autorskiego. W USA (w odróżnieniu od krajów Europy) dopuszcza się również w pewnym zakresie ochronę patentową programów komputerowych. Na gruncie prawa autorskiego zbudowany jest zwłaszcza model udostępniania open source, o czym piszę w dalszej części opracowania. Co warto zauważyć, to fakt, że wbrew powszechnemu przekonaniu licencjonowanie open source opiera się w całości właśnie na ochronie praw autorskich, a nie próbuje mu przeciwdziałać.

Choć prawo autorskie określane jest jako „wyłączne prawo" przyznawane autorowi dzieła, to w istocie polega ono na określaniu działań zabronionych dla podmiotów trzecich, takich jak kopiowanie utworu bądź jego fragmentów. Naruszenie tych zakazów pozwala dochodzić autorowi roszczeń od naruszającego poprzez zaniechanie tych działań, a także rekompensatę poniesionych strat lub utraconych korzyści [Jacob 2004, ss. 4-5]. Roszczeń tych dochodzi się w większości przypadków na gruncie prawa cywilnego. W pewnych sytuacjach stosuje się również prawo karne (m.in. kopiowanie płyt CD/DVD, podrabianie perfum i markowej odzieży). Ma to szczególne znaczenie dla oszacowania faktycznych kosztów ochrony, jakie ponosi autor, aby dochodzić swoich praw. Są to nie tylko koszty postępowania sądowego, ale także monitorowania rynku, aby wychwycić nieuczciwych konkurentów. W praktyce większość spraw załatwianych jest na drodze porozumienia stron (por. Jacob [2004, s. 6], Mossoff [2001]).

Istnieją co najmniej dwa powody, dla których oprogramowanie otrzymało prawnoautorską ochronę na wzór utwór artystycznych. Po pierwsze, wraz z pojawieniem się oprogramowania „z pudełka" na rynku konsumenckim, wystąpił problem piractwa komputerowego, bardzo podobny np. do piractwa fonograficznego czy piractwa wideo. Podobnie wyglądała też dystrybucja obu rodzajów dóbr i przeznaczenie — program komputerowy można kupić w sklepie, zapakowany i dostarczony na nośniku typu płyta CD, którego można używać w domu lub w biurze i robić kopie zapasowe na swoje potrzeby, natomiast intuicyjnie za niedozwolone uznaje się robienie dalszych kopii i odsprzedawanie ich[12]. W tym sen-

[12] Należy zaznaczyć, że sprzedaż oprogramowania „z pudełka", de facto sprzedaż licencji na jego używanie w ograniczonym zakresie, to tylko pewien fragment całego rynku oprogramowania.

sie przyznana ochrona prawna byłaby na wzór wykonań utworów, a nie utworów autorskich sensu stricte. Prawo autorskie wyraźnie rozróżnia te kategorie, m.in. przyznając inny okres ochrony dla utworu i wykonania[13].

Po drugie, oprogramowanie jest tworzone w pierwotnej postaci przez programistów, którzy piszą kod źródłowy, tekst zrozumiały dla człowieka w języku programowania, opis algorytmu programu komputerowego. Tekst ten jest następnie automatycznie tłumaczony na język maszynowy danego komputera (kompilowany). Stąd pojawiła się metafora tekstu programu komputerowego jako utworu literackiego. I chociaż programista tworzący treść programu działa w bardzo sztywnych ramach języka programowania, bardziej przypominającego wyrażenia matematyczne niż prozę literacką, to sam proces jest de facto twórczy, o czym świadczyć może wciąż brak pełnej automatyzacji tego procesu. Nie istnieje system, który pisałby programy komputerowe bez udziału człowieka.

Tak więc prawnoautorska ochrona oprogramowania jest powszechnie przyjętym rozwiązaniem. Zgodnie z polską ustawą, zharmonizowaną z dyrektywą Komisji Europejskiej, programy komputerowe podlegają ochronie jak utwory literackie (art. 74 pr. aut.). Przedmiotem ochrony są programy komputerowe w dowolnej postaci (niezależnie od formy ich wyrażenia), zarówno jako tekst źródłowy, jak i kod maszynowy. Taka sama ochrona przysługuje oprogramowaniu wydrukowanemu w gazecie, dostarczonemu na płycie CD czy zakupionemu z gotowym urządzeniem. W ten sam sposób chronione są również interfejsy. Przedmiotem ochrony nie są idee, pomysły, algorytmy, zasady, procedury. Ochrona przysługuje natomiast ich realizacjom [Barta, Markiewicz 2002; Barta, Markiewicz 2005].

3. Teoretyczny model biznesowy firmy open source

Analiza firmy udostępniającej oprogramowanie może być przeprowadzona na podstawie pewnego analitycznego modelu przedsiębiorstwa. Pierwszy taki teoretyczny model zaproponował Elad Harison [2008]. Jego model opisuje firmę produkującą oprogramowanie, która podejmuje decyzje, czy jego części udostępnić jako open source. Model Harisona przyjmuje wiele uproszczeń, m.in. brak ryzyka związanego z tworzeniem nowej funkcjonalności, zarówno wewnątrz firmy, jak i przez społeczność open source. Dużym uproszczeniem jego modelu jest również założenie, że firma zarabia wyłącznie na zamkniętych elementach swojego oprogramowania (jako że otwarte publicznie elementy mogą być bardzo tanio i szybko wykorzystane przez konkurencję, nie są źródłem przewagi konkurencyjnej). W praktyce elementy otwarte publicznie mogą również generować dla firmy wiele korzyści. W modelu tym Harison zakłada, że każda firma stale rozwija technologicznie swoje produkty, inwestując w badania i rozwój (B&R). Zwiększenie jakości technologicznej rozwijanego oprogramowania oznaczone jest jako R w przytaczanym modelu[14].

[13] Zgodnie z polską ustawą dla wykonań i nagrań okres ochronny wynosi 50 lat, a dla utworów literackich 70 lat.
[14] Harison oparł swój model na modelu zmian technicznych (ang. *model of technical change*) Nordhausa z 1969 r. oraz na pojęciu „drabiny jakości" Grossmana z 1991 r. (por. Harison [2008, ss. 94-101]).

Harison rozwija swój model w dwóch krokach: model firmy krótkowzrocznej, która jest zorientowana na krótkoterminowy zysk (w pojedynczym okresie), oraz firmy dalekowzrocznej, która chce maksymalizować zysk długoterminowo, realizując swoją strategię konsekwentnie w kolejnych okresach. Pierwszy model rozważa zyski i straty w pojedynczym okresie, natomiast drugi jest jego transpozycją na *n* kolejnych okresów. Dla zwięzłości wywodu w artykule pomijam opis modelu firmy dalekowzrocznej.

3.1. Model firmy krótkowzrocznej

Model opisuje pojedynczą firmę. Jako że przyjęto brak ryzyka związanego z budową nowych funkcjonalności, rozwój ich wewnątrz firmy może zostać w całości zastąpiony rozwojem przez społeczność open source, jeśli kod źródłowy danej funkcjonalności zostanie udostępniony. Firma podejmuje decyzję, jak dużą ilość funkcjonalności otworzyć publicznie, aby były one rozwijane przez społeczność open source, a jaką część pozostawić zamkniętą. Celem firmy jest maksymalizowanie zysku. Parametr:

$$\alpha \in [0,1]$$

to stopień otwartości przedsiębiorstwa: $\alpha = 0$ oznacza, że żaden element nie jest otwarty publicznie; $\alpha = 1$ oznacza, że wszystkie funkcjonalności są otwarte publicznie.

Koszt tworzenia nowej funkcjonalności samodzielnie (funkcjonalności zamkniętej) to dla firmy $\bar{c}$, natomiast koszt tworzenia nowej funkcjonalności przez społeczność open source (funkcjonalność otwarta) to dla firmy c_1. Ponieważ społeczność open source wykonuje większość zadań bez wynagrodzenia z firmy (niemniej firma nadal ponosi pewne koszty, np. związane z koordynacją i integracją), to $0 < c_1 < \bar{c}$ [15].

Zatem wydatki firmy na B&R to suma wydatków na samodzielne rozwijanie nowych funkcjonalności, $\bar{c} \cdot (1-\alpha) \cdot R$ oraz wydatków na rozwijanie funkcjonalności w modelu open source, $c_1 \cdot \alpha \cdot R$.

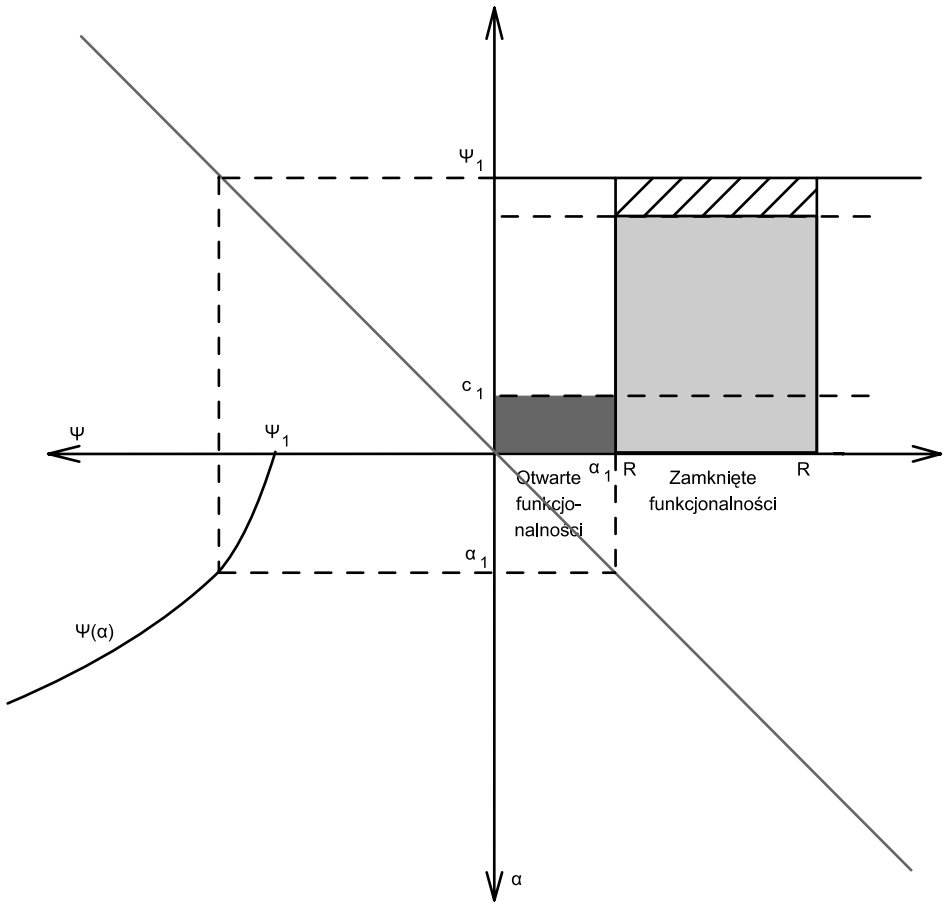
Dochód uzyskany z jednej funkcjonalności jest funkcją poziomu otwartości, $\Psi(\alpha)$, zgodnie z założeniem, że otwarte funkcjonalności mogą być łatwo wykorzystane przez konkurencję, a więc firma nie może generować z nich dochodów. Przychody są generowane tylko z tych funkcjonalności, które są dystrybuowane jako własnościowe (zamknięte), $(1 - \alpha)R$. Zysk dla firmy w pojedynczym okresie to różnica między wygenerowanymi przychodami a kosztami tworzenia nowych funkcjonalności, zamkniętych i otwartych.

Firma maksymalizuje zyski w pojedynczym okresie, π, szukając optymalnego podziału między rozwojem nowych funkcjonalności samodzielnie a rozwojem ich jako open source, uwzględniając przychody, jakie może wygenerować: (1.1)

$$\pi = (1-\alpha)\Psi(\alpha)R - [\bar{c}(1-\alpha)R + c_1\alpha R] = (1-\alpha)\Psi(\alpha)R - [\bar{c} - \alpha(\bar{c} - c_1)] \cdot R$$

[15] Harison pominął dolne ograniczenie c_1.

Rysunek 1. Model dla pojedynczego okresu



Uwaga: dla zadanego R zmiany α powodują zmianę proporcji R, z których firma generuje zyski, a także $\Psi(\alpha)$, dochodowość funkcjonalności.

Źródło: Harison [2008, s. 98].

Rysunek 1 ilustruje zależności pomiędzy składnikami równania. Dla zadanego przyrostu technicznej jakości produktu R (oś odciętych w I ćwiartce wykresu) firma podejmuje decyzję o wybranym poziomie otwartości swojego oprogramowania, α_1. α_1 jest argumentem funkcji dochodu $\Psi(\alpha)$, która jest pokazana w III ćwiartce wykresu. Dochodowość Ψ_1 określa zyski, które, jako że są generowane tylko z tej części funkcjonalności, która pozostaje zamknięta, wyraża się jako $\Psi_1 (1 - \alpha_1)R$ (pola jasnoszare i zakreskowane).

Poniesione koszty to suma kosztów budowy otwartych funkcjonalności (pole ciemnoszare) oraz zamkniętych funkcjonalności (pole jasnoszare). Końcowy zysk to różnica między przychodem

z zamkniętej części oprogramowania a kosztami obu rodzajów funkcjonalności oprogramowania. Zwiększanie wartości α może zwiększać dochodowość pojedynczej funkcjonalności, ale zmniejsza udział tej części oprogramowania, z której firma generuje zyski.

Harison wymienia kilka efektów, jakie przynosi przyrostowa zmiana współczynnika otwartości α na koszty i zyski:

1) Przychody maleją o

$$\Psi(\alpha)\cdot R-(1-\alpha)\frac{d\Psi(\alpha)}{d\alpha}\cdot R$$

ponieważ zwiększa się udział niedochodowej części oprogramowania,

2) koszty części open source rosną o
$c_1\cdot d\alpha\cdot R$

3) koszty części zamkniętej maleją o
$\bar{c}\cdot d\alpha\cdot R$

Na rozwiązanie postawionego problemu, maksymalizacji zysku duży wpływ ma postać zależności dochodowości od otwartości $\Psi(\alpha)$.
Harison różniczkuje równanie 1.1, by pokazać dynamikę przyrostu zysków od współczynnika otwartości:
(1.2)

$$\frac{d\pi}{d\alpha}=-\Psi(\alpha)\cdot R+(1-\alpha)\frac{d\Psi(\alpha)}{d\alpha}\cdot R+(\bar{c}-c_1)\cdot R$$

Następnie podaje trzy propozycje wariantów funkcji dochodowości od otwartości[16]:

Propozycja 1. Jeśli przychody per moduł są stałe ($\Psi(\alpha) = \Psi$), optymalną strategią jest zachować wszystkie cechy oprogramowania jako własnościowe[17].

Propozycja 2. Jeśli funkcja przychodów spełnia warunki:

$$\begin{cases}\dfrac{d\Psi(\alpha)}{d\alpha}>\left[\dfrac{1}{1-\alpha}\left(\Psi(\alpha)-(\bar{c}-c_1)\right)\right] & \text{dla } \alpha=0\\ \Psi(1)>(\bar{c}-c_1)\end{cases}$$

to $0 < \alpha^* < 1$.

Propozycja 3. Jeśli $\Psi(\alpha)$ jest ciągła i

$$\Psi(\alpha^*)\leqslant(\bar{c}-c_1)+\frac{c_1}{1-\alpha^*}$$

to firma działająca przy otwartości $\alpha = \alpha^*$ będzie zwiększać (zmniejszać) poziom otwartości swojego oprogramowania.

3.2. Analiza modelu firmy krótkowzrocznej

Jak zatem wynika z przedstawionego wyżej opisu, jeśli w przypadku danej firmy dochodowość nie zależy od stopnia jej otwartości, to optymalną strategią dla firmy będzie zachowanie całości

[16] Dla przejrzystości pomijamy tu całą dyskusję z oryginału (por. Harison [2008, ss. 97-98]).

[17] Dowód: z równania 1.2: $\frac{d\pi}{d\alpha} < 0$, gdy $\frac{d\Psi(\alpha)}{d\alpha} = 0$, zatem $\alpha^* = 0$.

oprogramowania jako zamkniętego. Innymi słowy, jeśli udostępnienie żadnej z funkcjonalności nie przynosi firmie dodatkowych korzyści, a jednocześnie przestałaby ona generować z niej zyski (zgodnie z założeniem, że zarabia tylko na własnościowych), to nieopłacalne jest dla niej jej udostępnianie. W efekcie żadna z części oprogramowania nie zostaje otwarta.

Jeśli jednak udostępnianie kolejnych funkcjonalności (co odpowiada zwiększaniu otwartości α) powoduje znaczny wzrost dochodowości, to firmie będzie się opłacać otworzyć część funkcjonalności (jednak nie wszystkie). Wzrost ten musi być znaczny, gdyż zbyt powolny nie będzie rekompensował utraconych zysków, możliwych do realizacji, gdyby wszystkie funkcjonalności pozostały zamknięte.

Taka analiza wskazuje na dwa istotne warunki, jakie muszą być spełnione, aby udostępnianie open source faktycznie zwiększało rentowność firmy:
1) Firma musi zachować część funkcjonalności jako zamkniętych, aby generować na nich zyski.
2) Ponadto polityka udostępniania funkcjonalności otwartych musi w istotny sposób wpływać na dochodowość pozostałych, zamkniętych części.

Przykładem może być firma, która produkuje systemy operacyjne oraz aplikacje dla swojej platformy. Jeśli teraz:

- udostępni interfejsy swojego systemu (API[18]), aby ułatwić tworzenie nowych aplikacji na tę platformę, a jednocześnie uzyskać zwrotną informację od społeczności programistów, aby ją udoskonalać,
- udostępni część aplikacji użytkowych, które nie mają istotnego wpływu na jej biznes, za to przyspieszą powstawanie nowych aplikacji w społeczności, gdyż gotowy kod będzie na różne sposoby udoskonalany i przerabiany,
- udostępni darmowe narzędzia wspomagające tworzenie oprogramowania na jej platformę,

to w efekcie:

- nastąpi szybki wzrost liczby dostępnych, darmowych aplikacji na daną platformę,
- aplikacje te będą wysokiej jakości, intensywnie przetestowane,
- poszerzona współpraca ze społecznością podniesie jakość samej platformy,
- bezpośrednia konkurencja (firmy tworzące konkurencyjne systemy operacyjne) nie będzie mogła wykorzystać łatwo udostępnionego oprogramowania,
- platforma stanie się o wiele bardziej atrakcyjna dla klientów dzięki temu, że jest stabilniejsza i istnieje dla niej dużo dobrych darmowych aplikacji. Zatem firma będzie stale powiększać udział w rynku z jej podstawowym produktem.

Taką właśnie strategię przyjęła np. firma Microsoft, o której piszę w dalszej części.

Wspomniany prosty model Harisona firmy wykorzystującej open source pokazuje problem pewnej trudności opisu zjawiska, jakie zaobserwowano już kilka lat temu — wiele firm na świecie, zarówno małych, średnich przedsiębiorstw, jak i prawdziwych gigantów przemysłu

[18] Ang. *Application Programming Interface* — interfejs programowania aplikacji, powszechnie używany termin na określenie interfejsów dedykowanych współpracy z innymi programami.

informatycznego, z powodzeniem nauczyło się wykorzystywać open source w biznesie.

Niektórzy autorzy postulują potrzebę stworzenia nowych modeli strategii biznesowej, pozwalających opisać obserwowane zjawisko wykorzystania otwartego oprogramowania. Obserwacje te często są przeciwstawiane dotychczasowym modelom strategicznym, które zakładają przede wszystkim wypracowanie i ochronę swojej pozycji na konkurencyjnym rynku, m.in. poprzez tworzenie wysokich barier dla konkurentów i współzawodnictwo, a nie otwartość i dzielenie się wiedzą. Natomiast, jak można zaobserwować, wykorzystanie open source, a więc dzielenie się know-how, otwartość i współpraca, pozwala firmom na wspólne tworzenie wartości i stymulowanie innowacji [Chesbrough, Appleyard 2007].

4. Modele strategiczne

Chesbrough i Appleyard [2007] zaproponowali nowe podejście do analizy strategicznej, bazując na wcześniejszych pracach Chandlera, Ansoffa, Portera i innych, nazywanych „otwartą strategią" (ang. *open strategy*). Strategia ta łączy tradycyjne modele strategii biznesowych z potencjałem, jaki jest przypisywany open source. Otwartość i innowacyjność ma być ukierunkowana na powiększenie wartość generowanej dla przedsiębiorstwa. Strategia ta wprowadza jednocześnie nowe modele biznesowe bazujące na wykorzystaniu społeczności open source i pozyskaniu również w ten sposób wiedzy do organizacji z zewnątrz. Istotne jest przy tym, aby ze wspólnie wypracowywanej wartości, firma mogła zatrzymać jej część dla siebie. Efektywna strategia ma pozwalać na utrzymanie równowagi między wartością wspólnie tworzoną a wartością zatrzymaną przez przedsiębiorstwo. Autorzy wskazują przy tym, że ten model strategii jest szczególnie istotny dla firm, które chcą dominować na rynku dzięki przewadze innowacyjności swoich produktów bądź usług.

Konstruując swoją propozycję otwartej strategii, Chesbrough i Appleyard [2007] wiele uwagi poświęcili modelowi pięciu sił Portera. Fitzgerald [2006] za to podzielił rozwój ruchu open source na dwa etapy, pierwszy (FOSS, ang. *Free Open Source Software*) skoncentrowany na wolnościowym aspekcie tworzonego oprogramowania oraz drugi (OSS 2.0, ang. *Open Source Software* 2.0), w którym znacznie więcej uwagi poświęcano komercjalnemu wykorzystaniu open source. Zarówno w pierwszym, jak i drugim istnieją dwa podstawowe modele biznesowe generowania przychodów: poprzez tworzenie wartości dodanej oraz penetrowania lub tworzenia rynku. Cechą drugiego etapu jest powstawanie nowych licencji przemysłowych upraszczających komercjalizację oprogramowania open source oraz powstanie nowych modeli biznesowych, takich jak zwielokrotnianie produktywności czy wykorzystywanie efektu reputacji.

5. Strategie wybranych firm ICT

W opracowaniu analizuję firmy z sektora ICT — Microsoft, Google, LG oraz Samsung Electronics, z których każda ma inny model biznesowy i inne doświadczenia z wykorzystaniem open source. Microsoft długi czas ignorował ruch open source, a w pewnym momencie był wręcz uznawany za jego wroga, społeczność open source zaś nieraz otwarcie występowała przeciw firmie, np. deklarując chęć stworzenia produktów, które miały odebrać firmie pozycję na rynku. Obec-

nie firma ściśle współpracuje ze społecznością open source i sporo inwestuje w jej rozwój.

Google jest najmłodszą firmą od początku identyfikowaną z ruchem open source i od początku działającą w przestrzeni Internetu. Ma odmienny model biznesowy, gdzie indziej niż Microsoft generuje zyski oraz efektywnie wykorzystuje open source, penetrując istniejące i tworząc nowe rynki oraz zabezpieczając własne źródła przewagi konkurencyjnej.

LG jest firmą o profilu najbardziej zbliżonym do firmy Samsung Electronics. Jest też bezpośrednim konkurentem Samsunga. Obie działają na rynkach elektroniki użytkowej, na których istotnym czynnikiem przewagi konkurencyjnej jest zaplecze produkcyjne. Jednak bardzo wyraźnie branża elektroniki użytkowej podąża w ślad za firmami softwarowymi, adaptując wypracowane przez nie rozwiązania tworzenia oprogramowania.

Przyglądając się analizowanym firmom i wypracowanym przez nie praktycznym modelom biznesowym, należy wyciągnąć podobne wnioski, jakie wynikały z modelu teoretycznego firmy open source: aby efektywnie wykorzystać open source, firma musi zabezpieczyć swoje podstawowe źródła przewagi konkurencyjnej — produkty lub usługi, z których generuje główne zyski, oraz tak skonstruować politykę udostępniania open source, aby wywoływała efekt dźwigni dla sprzedaży i rentowności podstawowych produktów lub usług.

Zarówno Microsoft, jak i Google pielęgnują i chronią swoje podstawowe źródła zysków — kody systemów operacyjnych Windows w przypadku pierwszej firmy oraz zaawansowany mechanizm indeksowania stron WWW i szybkiego wyszukiwania w przypadku drugiej. Tworzenie i stymulowanie społeczności open source przynosi obu firmom wiele złożonych korzyści, które następnie pośrednio przekładają się na zwiększenie dochodów przedsiębiorstwa w dłuższej perspektywie czasowej.

W przypadku Microsoft obserwujemy wyraźną zmianę paradygmatu w podejściu do otwartości. Początkowo firma prezentowała całkowicie zamknięte podejście do zarządzania swoją własnością intelektualną. Prezentowała stanowisko, że ruch open source przyczynia się jedynie do erozji rynku oprogramowania, a porównać by go można do ruchów anarchistycznych w polityce. Animozje te odwzajemniali uczestnicy ruchu open source, często wyraźnie je artykułując, skąd wzięło się powszechne mniemanie o antybiznesowym i niekomercyjnym wykorzystaniu open source. Jednak pozytywne doświadczenia innych firm oraz z pewnością obserwacja rozwoju samego ruchu open source, spowodowały, że firma zmieniła podejście. Dzisiaj Microsoft oficjalnie podejmuje wiele inicjatyw i publikuje znaczne ilości kodu w modelu open source. Wśród korzyści, jakie odnosi w ten sposób, należy wymienić budowanie silniejszych relacji z partnerami, promocję własnych rozwiązań i technologii, budowanie innowacyjnej marki, zwiększanie produktywności za pomocą zewnętrznych zasobów i przede wszystkim penetracja rynku i zabezpieczanie swojej pozycji (por. Cusumano & Selby [1998], Microsoft [2010]).

Google od początku wykorzystywał i promował rozwiązania open source. Być może dlatego, że firma jest stosunkowo młoda. Jej powstanie przypada na okres, gdy Microsoft dostrzegł, że zaniedbał rynek, jaki właśnie się tworzy — Internet. Również w tym czasie Microsoft zaczął

Rysunek 2. Łańcuch wartości firmy wykorzystującej open source

Źródło: Opracowanie własne na podstawie Porter [2006, s. 65].

dostrzegać, że open source trudno ignorować, jeszcze trudniej zwalczać, natomiast można go wykorzystać [Google 2010].

Google wykorzystuje open source na dwa sposoby — po pierwsze na bazie gotowego otwartego oprogramowania zbudował własne utajone rozwiązania. Nie dystrybuuje tego kodu dalej ani nie sprzedaje jako produktu, więc nie ma problemu z kompatybilnością z licencjami open source (głównie GPL). Warto wspomnieć, że Microsoft ujawnił fakt, że w kodach swojego systemu Windows NT wykorzystuje gotowe komponenty dostępne na licencji BSD (co oczywiście nie jest naruszeniem tej licencji). Drugi sposób, to otwieranie własnego kodu (często zbudowanego z wykorzystaniem gotowych komponentów, np. Linuksa) nowych aplikacji, niezależnych od wewnętrznych systemów firmy. Aplikacje te są udostępniane publicznie całej społeczności. Dzięki temu firma buduje ogromny rynek wokół swojej platformy i podstawowych usług, jakie oferuje. Docelowo każda informacja w Internecie ma przechodzić przez serwery Google. Dałoby to w oczywisty sposób ogromną przewagę firmie na jej podstawowym rynku — sprzedaży reklamy i informacji marketingowej.

6. Analiza strategii firmy udostępniającej OSS

Jak wykazałem w poprzednich punktach, wykorzystanie open source ma ogromny wpływ na działalność operacyjną przedsiębiorstwa. Można pokazać ten efekt za pomocą modelu łańcucha wartości Portera [2006]. Poniżej przedstawiam model łańcucha wartości dla typowej firmy udostępniającej OSS (nie tylko z branży elektroniki użytkowej).

Na schemacie zaznaczono wpływ wykorzystania open source na poszczególne elementy łańcucha wartości:

- infrastruktura firmy — wpływ ujemny: 1) firma musi dodatkowo rozbudować i utrzymywać swoją infrastrukturę teleinformatyczną, aby zapewnić prowadzenie projektów open source w rozproszonym środowisku internetowym; 2) firma powinna również rozbudować swoją infrastrukturę o narzędzia wspomagające monitorowanie potencjalnych naruszeń własnych praw autorskich i patentowych przez

konkurencję oraz ewaluacje wprowadzanych z zewnątrz rozwiązań;
- zarządzanie zasobami ludzkimi — wpływ dodatni: 1) firma współpracując ze społecznością open source, buduje markę innowacyjnego pracodawcy, jest to jeden ze sposobów na tworzenie tzw. employer branding; 2) jednocześnie firma ma dostęp do bazy talentów — programistów pracujących przy projekcie z całego świata, baza taka jest aktualna i wiarygodna, a wyniki pracy uczestników projektu są oceniane na bieżąco;
- rozwój technologii — wpływ dodatni: 1) firma pozyskuje nowe pomysły i rozwiązania z zewnątrz, wprowadzane są one często przez użytkowników końcowego produktu, a więc odpowiadają na faktyczne zapotrzebowanie klientów; 2) dodatkowo otwarte rozwiązanie jest intensywniej testowane, w różnych konfiguracjach, jakie spotykane są u klienta, a więc finalny produkt jest bardziej dojrzały i stabilny;
- zaopatrzenie — wpływ dodatni: publikowanie produktu jako otwartego pozwala na wykorzystanie dostępnych, gotowych modułów ze świata open source (choć nie wszystkie licencje są kompatybilne ze sobą); dotyczy to komponentów oprogramowania, a także takich elementów, jak grafika, zdjęcia, szablony (dostępnych na podobnych licencjach, np. Creative Commons);
- logistyka wewnętrzna — wpływ ujemny: firma, prowadząc logistykę wewnętrzną, musi podejmować dodatkowy wysiłek na synchronizację działań wewnętrznych z informacją dostępną publicznie w ramach projektów open source, musi dbać o rozbudowane systemy kontroli wersji, repozytoria kodu oraz aktualność baz danych projektów;
- działania operacyjne — wpływ mieszany: 1) z jednej strony firma uzyskuje silne wsparcie z zewnątrz przy produkcji, testowaniu i dystrybucji; 2) z drugiej strony musi podejmować dodatkowe działa związane z koordynacją projektu oraz pilnowaniem potencjalnych naruszeń przez konkurencję;
- logistyka zewnętrzna — wpływ dodatni: publikowanie oprogramowania znacznie upraszcza logistykę zewnętrzną, oprogramowanie takie większość klientów może po prostu ściągnąć z Internetu, nie trzeba rozwijać specjalnych zabezpieczeń przed kopiowaniem ani utrzymywać systemów ewidencji aktywnych kopii oprogramowania;
- marketing i sprzedaż — wpływ dodatni: firma bez dodatkowych nakładów uzyskuje dodatkowy kanał promocyjny oraz generujący leady sprzedażowe; klienci pozyskujący samodzielnie oprogramowanie z Internetu sami poszukują usług dodanych (wsparcia technicznego, rozszerzonych wersji produktów) na stronie producenta;
- serwis — wpływ dodatni: w rozproszonym środowisku łatwiej znaleźć odpowiednie zasoby, gdy pojawia się problem, firma nie musi szukać za każdym razem, gdy powstaje sytuacja awaryjna, zespołu programistów wewnątrz — może ich pozyskać na zewnątrz, w ramach społeczności open source; również czas reakcji jest znacznie krótszy, a użytkownicy mają większą tendencję do bezpośredniej współpracy z twórcą danego fragmentu kodu niż w klasycznym modelu zamkniętego oprogramowania.

7. Propozycja strategii z wykorzystaniem open source dla firmy z branży elektroniki konsumenckiej

7.1. Uwarunkowania i cele strategii

Dotychczasowe rozważania i analizy wskazywały, że odpowiednie wykorzystanie open source może być elementem bardzo efektywnej strategii biznesowej. Odpowiednie, to znaczy przy zachowaniu dwóch niezbędnych warunków:

- firma musi zabezpieczyć inne źródła przewagi konkurencyjnej, które pozwalają generować jej zyski w ramach podstawowego biznesu,
- udostępniany kod źródłowy w modelu open source musi przyczyniać się do zwiększania rentowności podstawowego biznesu.

Pierwszy warunek może być zaspokojony w dwojaki sposób. Po pierwsze, firma najczęściej ma silne źródło przewagi konkurencyjnej w postaci zasobów i infrastruktury produkcyjnej — fabryk lub trwałych relacji z producentami, którym zleca wytwarzanie swoich produktów. Ten element nie będzie zagrożony w przypadku udostępnienia publicznie części oprogramowania jako open source. Po drugie udostępnianie publicznie kodów źródłowych nie powinno dotyczyć tych elementów oprogramowania, które w istotny sposób wpływają na wyróżnienie produktu na tle konkurencji, a jednocześnie są trudne do skopiowania przez konkurentów bez znajomości kodu źródłowego. Takim elementem są np. kody sterowników urządzeń (ang. *drivers*).
Drugi warunek będzie zachowany wtedy, gdy publicznie udostępniony projekt open source będzie się przyczyniać do większej atrakcyjności podstawowych produktów firmy, dzięki czemu będą one chętniej kupowane przez klientów. Taki wpływ miałoby z pewnością podnoszenie jakości platformy wbudowanego oprogramowania w urządzenie, zaawansowanie techniczne tej platformy, duża liczba oczekiwanych przez klientów funkcjonalności oraz duża liczba aplikacji współpracujących z urządzeniem.

Można więc sformułować następujące cele strategii wykorzystania open source:

- stałe podnoszenie jakości platformy oprogramowania, poprzez intensywne testowanie w różnych środowiskach i konfiguracjach systemowo-sprzętowych,
- rozwijanie nowych funkcjonalności produktowych, zbieranie ocen i propozycji bezpośrednio od użytkowników,
- kreowanie marki innowacyjnego przedsiębiorstwa tworzącego rozwiązania przyszłości,
- pozyskanie talentów ze środowiska programistów — podnoszenie produktywności i employer branding,
- zabezpieczenie kluczowych elementów oprogramowania wyróżniających produkt na tle konkurencji, a jednocześnie trudnych do skopiowania bez znajomości kodu źródłowego.

Aby zrealizować tak postawione cele, firma powinna zewidencjonować oprogramowanie tworzone wewnętrznie, zarówno oprogramowanie dedykowane do produkcji, jak i oprogramowanie wspomagające proces wytwórczy (narzędzia programistyczne, narzędzia do testowania, narzędzia wspierające proces na linii produkcyjnej), następnie zidentyfikować te komponenty, które powinny stanowić tajemnicę firmy, oraz te, które można udostępnić publicznie. Będzie to miało w oczywisty sposób wpływ na dekom-

pozycję i architekturę systemu, konieczne może okazać się przeprojektowanie istniejącego systemu.

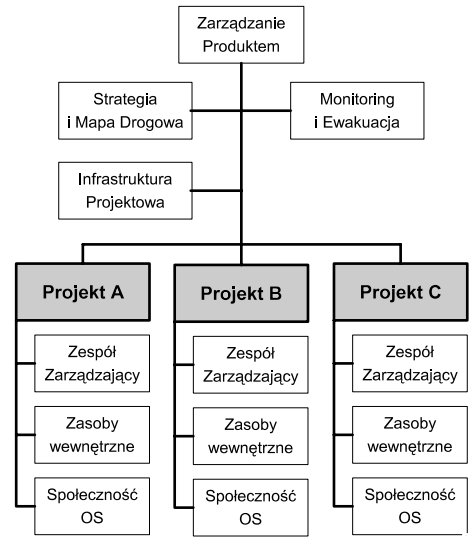
7.2. Zarządzanie projektem open source w przedsiębiorstwie

Aby zrealizować przedstawioną wyżej strategię rozwoju z wykorzystaniem open source, firma powinna wprowadzić zmiany organizacyjne w dziale badawczo-rozwojowym, który zajmuje się oprogramowaniem. Schemat organizacyjny proponowanej struktury przedstawia rys. 3. Na szczycie takiej struktury stoi zespół zarządzania produktem, który opiekuje się produktem pod kątem ogólnie przyjętej strategii. Zespół nakreśla długoterminowy kierunek rozwoju produktu, zwany potocznie mapą drogową, oraz ustala przyjętą strategię. Wsparciem zespołu zarządzania produktem jest zespół monitoringu i ewaluacji, który zajmuje się monitorowaniem ewentualnych naruszeń praw autorskich i patentowych przez konkurencję oraz ewaluacją wprowadzanego kodu pod kątem potencjalnych zagrożeń, np. naruszania cudzych praw. Na poziomie całego produktu utrzymywana jest również niezbędna infrastruktura projektowa, z wykorzystaniem której prowadzone są poszczególne projekty open source.

Następnie każdy z projektów open source jest prowadzony przez własny zespół zarządzający, który dba o to, aby kierunek jego rozwoju wpisywał się w ogólnie przyjęty kierunek rozwoju całego produktu, oraz o własne zasoby firmy, jak programiści, testerzy, projektanci. Do zadań zespołu zarządzającego projektem należy też promocja projektu w środowisku open source oraz pozyskiwanie najlepszych talentów do współpracy.

Aby obniżyć koszty prowadzenia projektów, a jednocześnie bardziej je uwiarygodnić i promować, firma może wykorzystać infrastrukturę projektową oferowaną za darmo przez partnerów, np. Microsoft CodePlex czy Google Code.

Rysunek 3. Schemat organizacyjny zarządzania projektami open source w przedsiębiorstwie



Źródło: opracowanie własne.

Bardzo ważny jest również wybór licencji, na której będzie udostępniany kod źródłowy. Odpowiednim wyborem może być licencja GPL, gdyż gwarantuje, że wszystkie prace pochodne będą dostępne zwrotnie dla społeczności na tych samych zasadach. Jednocześnie pozwoli to włączać do systemu gotowe komponenty dostępne publicznie zarówno na licencji GPL, jak i LGPL oraz innych, kompatybilnych licencjach.

7.3. Kontrola realizacji strategii

Realizacja strategii open source powinna być przez firmę monitorowana w celu kontroli jej skuteczności i być

może wprowadzania zmian. Poniżej wymieniam propozycje niektórych możliwych mierników:

- ile projektów open source jest realizowanych z wykorzystaniem udostępnionego kodu?
- ilu programistów ze społeczności open source angażuje się w te projekty? Jaka jest ich rotacja w projekcie?
- ile powstało linii kodu w poszczególnych projektach open source? Ile linii kodu we wszystkich projektach łącznie? Jak zmieniają się te liczby w czasie?
- ile firm trzecich i w ilu produktach wykorzystuje udostępnione komponenty bądź prace pochodne?
- ile zidentyfikowano naruszeń praw autorskich przez firmy trzecie?
- jak zmieniają się wyniki testów Q&A (testów jakości produktu)? Jak zmienia się poziom satysfakcji klientów?

Wskaźniki takie, zebrane i zestawione w stosunku rok do roku, będą stanowiły dla zarządu firmy miarodajne narzędzie kontroli realizacji i skuteczności omówionej strategii.

Ostatecznie spodziewane efekty, jakie powinna przynieść tak wdrożona strategia to:

- obniżenie kosztów produkcji oprogramowania i wzrost produktywności,
- podniesienie jakości i stabilności oprogramowania w urządzeniach,
- powstawanie nowych aplikacji dla użytkowników,
- rozwój nowych funkcjonalności oczekiwanych na rynku,
- wzrost popularności produktów i samej platformy,
- kojarzenie marki jako wysoce innowacyjnej.

A zatem firma uzyska większą przewagę konkurencyjną.

Podsumowanie

W artykule przeanalizowałem aktualny stan wiedzy teoretycznej, proponowane modele i teorie open source oraz przykłady firm wykorzystujących open source. Pokazałem, jaki wpływ ma włączenie open source do strategii firmy na poszczególne etapy łańcucha wartości. Najważniejsze wyniki przeprowadzonych rozważań i analiz są następujące:

- niezbędnym elementem skutecznego wykorzystania open source jest zachowanie innych źródeł przewagi konkurencyjnej (innych od rozwiązań w udostępnianym publicznie kodzie),
- open source ma szczególne zastosowanie w tych gałęziach przemysłu, w których duże znaczenie ma innowacyjność,
- aby w pełni opisać zjawisko i jego wpływ na przedsiębiorstwo, należy uwzględnić wiele czynników i zależności w całym ekosystemie danego biznesu,
- open source nie jest panaceum, jednak w odpowiednich warunkach może istotnie wpływać na pozycję konkurencyjną firmy.

Wskazałem również istotne warunki, jakie muszą być spełnione, aby wykorzystanie open source przynosiło dodatnie efekty:

- firma musi zabezpieczyć inne źródła przewagi konkurencyjnej, które pozwalają generować jej zyski w ramach podstawowego biznesu,
- udostępniany kod źródłowy w modelu open source musi przyczyniać się do zwiększania rentowności podstawowego biznesu.

Zagadnienie oprogramowania tworzonego w modelu open source, a zwłaszcza wykorzystanie go w strategii biznesowej

przedsiębiorstwa jest wciąż gałęzią nauki intensywnie rozwijającą się. Niezbędnych jest wiele badań, aby wnikliwie opisać i przeanalizować to zjawisko. Niektóre z możliwych kierunków kolejnych badań i analiz to:

- jak mierzyć produktywność społeczności open source? Jakimi narzędziami można je porównywać do produktywności inżynierów zatrudnianych na klasycznych zasadach?
- jak modelować zjawisko inercji i masy krytycznej w projektach open source? Jakie wielkości projektów są optymalne?
- planowanie finansowe w projektach open source — jak szacować nakłady i zyski w takich projektach? Jak porównywać je z tradycyjnym modelem na płaszczyźnie finansowej?
- jak efektywnie zarządzać projektami open source? Jak optymalnie balansować między pełną swobodą a wyznaczaniem sztywnych ról i celów?
- modelowanie i symulacja społeczności open source. Próby predykcji zachowania takich społeczności. Prognozowanie projektów open source.

Bibliografia

Barta J., Markiewicz R. [2002], *Ustawa o ochronie baz danych. Komentarz*, Warszawa, ABC.

Barta J., Markiewicz R. [2005], *Oprogramowanie open source w świetle prawa. Między własnością a wolnością*, Kraków, Zakamycze.

Blasi R., Brown S. [2005], *Getting To Market: Intellectual Property*, Retrieved Feb 18, 2010, from http://web.mit.edu/e-club/www/presentations/ip.pdf.

Chesbrough H., Appleyard M. [2007], *Open Innovation and Strategy*, "California Management Review", Vol. 50, No. 1.

Crowston K., Wei K., Howison J., Wiggins A. [2008], *Free/Libre Open Source Software Development: What We Know and What We Do Not Know*, Technical report, Syracuse University FLOSS Project.

Cusumano M., Selby R. [1998], *Microsoft Secrets, (Paperback Edition ed.)*, New York: Touchstone.

Fitzgerald B. [2005], *Has Open Source Software a Future? Perspectives on Free and Open Source Software*, MIT Press.

Fitzgerald B. [2006], *The Transformation of Open Source Software*, "Management Information Systems Quarterly", Vol. 30, No. 3.

Giera J. [2004], *The Costs And Risks Of Open Source, Forrester Research*, Forrester Research.

Google Inc. [2010], *Corporate Information*. Pobrano 8 maja 2010 r. z lokalizacji www.google.com/corporate/.

Harison E. [2008], *Intellectual Property Rights, Innovation and Software Technologies*, Edward Elgar Publishing.

Idris K. [2003], *Intellectual property*, WIPO.

IIPA [2009], *Copyright industries in the U.S. economy: The 2003-2007 Report*, International Intellectual Property Alliance.

Jacob S.R. [2004], *A Guidebook To Intellectual Property (5th Edition ed.)*, London, Sweet & Maxwell.

Komisja Europejska [2001], *Dyrektywa 2001/29/WE Parlamentu Europejskiego i Rady z dnia 22 maja 2001 r. w sprawie harmonizacji niektórych aspektów praw autorskich i pokrewnych w społeczeństwie informacyjnym*. Pobrano 22 lutego 2010 r. z lokalizacji http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=DD:17:01:32001L0029:PL:PDF.

Microsoft [2009], *Participation in a World of Choice; Perspectives on Open Source and Microsoft*, Microsoft Corporation.

Microsoft [2010], *Microsoft Open Source Website*. Pobrano 12 maja 2010 r. z lokalizacji www.microsoft.com/opensource/.

Mossoff A. [2001], *Rethinking the Development of Patents: An Intellectual History, 1550-1800*, "Hastings Law Journal", Vol. 52.

OECD [2010], *OECD Factbook 2010*, OECD Publishing.

OSI [2010], *Open Source Initiative*. Pobrano 15 grudnia 2010 r. z lokalizacji www.opensource.org/.

Parlament Europejski [1996], *Dyrektywa 96/9 Parlamentu Europejskiego z 11 marca 1996 r. o ochronie baz danych*.

Porter M., [2006], *Przewaga konkurencyjna. Osiąganie i utrzymywanie lepszych wyników*, Gliwice, Helion.

Raymond E. [1999], *The Cathedral and the Bazaar; Musings on Linux and Open Source by an Accidental Revolutionary*, Sabastopol, CA, O'Reilly.

Sejm RP [2001], *Ustawa z 27 lipca 2001 o ochronie baz danych*.

WIPO [2010], *Economic Contribution & Mapping*. Pobrano 14 grudnia 2010 r. z lokalizacji www.wipo.int/ip-development/en/creative_industry/economic_contribution.html.

Publishing Open Source Software in the Consumer Electronics Industry

Summary

The article discusses the problems of publishing software with Open Source Software [OSS] model as a tool for the company's business strategy. The essential question is whether the public disclosure of intellectual property of the company in the consumer electronics can boost its long-term results?
The experience of companies in the ICT sector and the stipulations of many authors lead to express the proposition that the publishing of software components with the OSS model will allow the company to gain competitive advantage. It is important at the same time to adopt an appropriate business strategy.
The study adopted an inductive-deductive method of research. Current state of theoretical knowledge was analysed as well as the proposed models and theories to make the attempt to extrapolate the field of consumer electronics. A systematic analysis of important ICT companies with different business models that use OSS has been performed. Those studies showed that:

- it is essential to the efficient use of OSS to protect the important sources of competitive advantage;
- analysed model is particularly applicable in those industries where it is important to innovation.

In addition key success factors influencing the company's competitive position has been identified.

Key words: open source, strategy, copyright, intangible asset, business model