Prawna ochrona oprogramowania Open Source

Krzysztof Siewicz[*]

Spis treści

Zamiast wstępu

Program komputerowy jest algorytmem, czyli zestawem instrukcji, którego przeznaczeniem jest dokonanie określonego zadania, wyrażonym w języku programowania zrozumiałym przez komputer i przez niego wykonywanym. Definicja powyższa nie jest definicją prawniczą, a rozumienie programu komputerowego w różnych systemach prawa jest rozmaite. Niektóre z nich nie zawierają definicji w ogóle, przez co zmuszają organy stosujące prawo do polegania na ustaleniach nauki informatyki w jej danym stadium rozwoju. Tak jest w przypadku prawa wspólnotowego i wielu systemów prawnych państw członkowskich. Zdarzają się jednak dość obszerne definicje prawnicze. Jedną z nich zawiera amerykański Copyright Act z 1976 roku po nowelizacji z roku 1980, stanowiący, że program komputerowy to "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result."[1]

Upraszczając, programy komputerowe mogą być wyrażone na dwa sposoby: w

[1] Copyright Act of United States of America Sec. 101, 17 U.S.C. Sec. 101 (1976 z późń. zm.).

kodzie źródłowym (source code) i wynikowym (object code). Kod źródłowy programu to jego zapis w jednym ze zrozumiałych dla człowieka języków programowania, takich jak na przykład C lub Java. Z uwagi na jego zrozumiałość, kod źródłowy jest wykorzystywany do tworzenia i zmieniania programów (debugowania, tworzenia nowych wersji). Kod wynikowy to tłumaczenie kodu źródłowego na zestaw instrukcji bezpośrednio realizowanych przez komputer, które są praktycznie niemożliwe do odcyfrowania przez człowieka. Komputery jednak bezpośrednio "rozumieją" jedynie kody wynikowe, a zatem każdy program musi zostać ostatecznie w ten sposób wyrażony. Tłumaczenie z kodu źródłowego na wynikowy jest określane jako "kompilacja" i dokonują go specjalne programy. Co istotne, po kompilacji możliwe jest rozpowszechnianie tylko kodu wynikowego, gdyż wystarcza on do uruchamiania programu w komputerze.

Program komputerowy, wyrażony w jednej z powyższych postaci wraz z pewnymi dodatkowymi elementami składa się na oprogramowanie (software). Te elementy można ogólnie podzielić na następujące kategorie: (1) narzędzia deweloperskie; (2) materiały przygotowawcze; (3) dane wejściowe; (4) wynik programu; (5) dodatkowe materiały (instrukcje, moduły pomocy); (6) interfejsy (w tym graficzne tzw. "look and feel" programu).[2] Tak jak kody źródłowe, narzędzia deweloperskie i materiały przygotowawcze mają znaczenie jedynie w trakcie tworzenia programu i zwykle nie są przedmiotem obrotu. Gotowy produkt oferowany na rynku użytkownikom końcowym stanowi zatem oprogramowanie składające się z kodu wynikowego i ostatnich czterech z wyżej wymienionych elementów.

Rynek oprogramowania

Mimo, że oprogramowanie rozprowadzane jest często jak zwykłe towary, różni się ono znacznie od rzeczy ruchomych. Przede wszystkim, jest ono informacją w czystej postaci, co jest prawdopodobnie najbardziej rzucającą się w oczy różnicą. Ponadto, oprogramowanie nie jest zwykle gotowym, ostatecznym i działającym bez zarzutu produktem końcowym. Wszyscy wiedzą, że każdy program komputerowy zawiera błędy (bugs), wynikające z praktycznej niemożliwości przewidzenia wszystkich stanów zachodzących w komputerze podczas uruchamiania programu, nawet jeżeli jest ono przygotowywane przez najbardziej

[2] Por. DAVID BAINBRIDGE, SOFTWARE COPYRIGHT LAW, 1-3 (Butterworths, London, Edinburgh, Dublin, 4th ed., 1999).

starannych autorów. Zwykle po wypuszczeniu oprogramowania na rynek i stopniowym ujawnianiu się błędów, opracowywane są dodatkowe pliki (patches) naprawiające te błędy. Jest także powszechnie przyjęte opracowywanie nowych wersji oprogramowania (upgrades), zwiększających jego funkcjonalność. Ciągły rozwój jest powodowany przez gwałtowny postęp w całej dziedzinie technologii informatycznych oraz ogólnym ideologicznym nastawieniem w tej dziedzinie przemysłu, które nie stawia za cel opracowanie dzieła skończonego, lecz raczej ewolucyjne i wieczne udoskonalanie. W wyniku tego "życie" konkretnego programu jest dość krótkie, zwłaszcza w porównaniu do terminów ochronnych ustanawianych przez prawo autorskie lub patentowe.

Rynek oprogramowania charakteryzuje się tak zwanymi efektami sieciowymi (network effects). Oznacza to, że wartość danego oprogramowania postrzegana przez użytkowników zwiększa się za każdym razem, gdy zdobywa ono nowych użytkowników. Powodowane jest to głównie przyczynami natury technicznej, jak wymogi kompatybilności i współpracy różnego oprogramowania ze sobą i ze sprzętem. Duże znaczenie odgrywa też zwykłe przyzwyczajenie użytkowników do konkretnych rozwiązań (choćby tak prozaicznych, jak kolejność poleceń w menu).[3] Wraz ze wzrostem liczby użytkowników rosną także koszty zmiany oprogramowania na inne, co ostatecznie może uniemożliwić porzucenie korzystania z konkretnego produktu, nawet jeżeli inne byłyby obiektywnie lepsze. Efekty sieciowe oznaczają po prostu to, że taniej jest korzystać z oprogramowania, z którego korzystają wszyscy inni.

Programy komputerowe, podobnie jak inne dobra intelektualne, cechują się relatywnie wysokimi kosztami produkcji. Wymaga ona dużo czasu, wielokrotnych testów i zmian, a często też konsultacji ekspertów z różnych dziedzin. Jednak dzięki technologii cyfrowej i dlatego, że programy są ze swej natury cyfrowe, po ich stworzeniu mogą zostać powielone niezwykle szybko i praktycznie bez jakichkolwiek dodatkowych kosztów. Do kopiowania oprogramowania nie jest potrzebna żadna specjalistyczna wiedza lub urządzenia. Jest jeszcze bardziej istotne, że kopie programów w ogóle nie różnią się od oryginału, czego nie można powiedzieć, na przykład o kopiach obrazów i innych dzieł tradycyjnie objętych ochroną prawa własności intelektualnej. Oznacza to, że kopiowanie umożliwia korzystanie z

[3] F. Warren-Boulton *et. al.*, *Economics of intellectual property protection for software: The proper role for copyright*, 3 no. 2 STANDARD VIEW 68, June 1995.

tego samego programu przez dowolną ilość osób, bez konieczności ograniczenia korzystania przez właściciela oryginału. To, że programy pisane z wielkim nakładem pracy i kosztów mogą być następnie używane bez jakichkolwiek opłat przez nieautoryzowanych "gapowiczów" i to w dodatku w jakości nieodbiegającej od tej, którą mogą zaoferować ich autorzy, w oczywisty sposób ma duży wpływ na kształt systemu prawnej ochrony oprogramowania.

System ten składa się głównie z prawa autorskiego. Ostatnio coraz większa uwaga przemysłu oprogramowania kieruje się w stronę ochrony patentowej. Jednak licencjonowanie, stanowiące połączenie instytucji prawa autorskiego i prawa zobowiązań, jest kluczowym elementem tego systemu.

Programy a prawo autorskie

Podstawową korzyścią ochrony prawnoautorskiej jest zabezpieczenie przed nieautoryzowanym kopiowaniem. Ochrona ta przynależy utworom, w rozumieniu prawa autorskiego i nie od samego początku było oczywiste, że programy komputerowe mogą być za utwory uznane. Tradycyjne przejawy twórczości ludzkiej objęte prawem autorskim nie mają, w przeciwieństwie do programów, użytkowego charakteru.[4] Utwory zwykle służą międzyludzkiej komunikacji, czego również nie można powiedzieć o programach.[5] Programy są po prostu przepisem na rozwiązanie konkretnego zadania, nie jest ich celem dostarczanie emocji estetycznych, ani nawet edukacja odbiorcy. Ich odbiorcami nie są bowiem ludzie, a komputery, nawet jeżeli ich działanie może być analizowane przez informatyków. Programy komputerowe określane są poprawnie jako "maszyny" przez takich amerykańskich autorów jak Pamela Samuelson, co wskazuje na rzeczywiste źródło ich wartości – konkretne zachowanie.[6] Ten sam rezultat, zachowanie programu, może zostać osiągnięte przez rozmaite sposoby wyrażenia instrukcji, różne zapisy programu. Oznacza to, że konkretna forma wyrażenia programu nie jest tak bardzo wartościowa, jak to ma miejsce w przypadku innych rodzajów dóbr objętych ochroną prawnoautorską.

Te i inne cechy szczególne oprogramowania mogą być brane pod uwagę w dyskusji,

---

4 JOSEPH DREXL, WHAT IS PROTECTED IN A COMPUTER PROGRAM?: COPYRIGHT PROTECTION IN THE UNITED STATES AND EUROPE, 9-12 (VCH Verlagsgesellschaft mbH, Weinheim, New York, 1994).

5 *Id.*

6 Pamela Samuelson *et. al., A Manifesto Concerning The Legal Protection Of Computer Programs*, 94 COLUMBIA LAW REVIEW, December 1994, at 2308, 2316 *et seq.*

czy prawo autorskie jest rzeczywiście właściwe dla jego ochrony. Lecz ani cel utworu, ani źródło jego wartości nie jest kryterium nadawania ochrony prawnoautorskiej. Ponadto, istnieją przykłady wielu utworów użytkowych cieszących się tą ochroną na długo przed pojawieniem się programów komputerowych. Nie wdając się w prawnicze dyskusje, można ogólnie stwierdzić, że podstawowym warunkiem uznania za utwór jest jego oryginalność, rozumiana jako indywidualny wkład twórczy autora. Systemy prawne wymagają obecnie minimalnego wkładu twórczego dla przyznania ochrony. Jest tak zwłaszcza w przypadku prawa amerykańskiego, a i standard wymagany przez Dyrektywę o prawnej ochronie programów komputerowych[7] jest postrzegany jako dość niski. Niezależnie od tego, ochrona prawnoautorska dla programów wynika z takich umów międzynarodowych jak TRIPS[8] i, za jej pośrednictwem, Konwencja berneńska[9]. Jest ona wymagana także przez WIPO Copyright Treaty[10]. Oczywiście, bezpośrednią podstawę ochrony stanowią ustawy, takie jak amerykański Copyright Act[11], po nowelizacji w 1980 roku.

Jak już zostało to zasygnalizowane, oprogramowanie składa się z różnorodnych elementów. Dla określenia ich indywidualnej ochrony prawnoautorskiej zwykle odróżnia się elementy o charakterze tekstowym od pozostałych. Te pierwsze, czyli instrukcje zapisane w kodzie źródłowym lub wynikowym są obecnie uznawane za podlegające ochronie bez większych przeszkód, chociaż toczyła się dyskusja czy kody wynikowe tworzone nie przez człowieka, lecz przez kompilatory mogą być kwalifikowane jako utwory. Pozostałe elementy, takie jak struktura programu (tak zwane SSO – structure, sequence and organization), wykresy (flow charts), struktura menu, interfejsy, są badane przez organy stosujące prawo czy stanowią twórcze wyrażenie pewnej abstrakcyjnej idei, czy też stanowią jedynie taką właśnie ideę.[12] Udzielanie ochrony jedynie konkretnym twórczym sposobom wyrażenia idei, a nie samym abstrakcyjnym ideom jest bowiem od dawna główną zasadą prawa autorskiego. Jest to jednak kryterium tak płynne, że nie należy się kiedykolwiek spodziewać jego sprecyzowania wobec jakiegokolwiek rodzaju utworów. Wobec programów komputerowych jest ono

---

7 91/250/EEC, OJ L122, 17/05/1991, P.42.

8 WTO Agreement on Trade-Related Aspects of Intellectual Property Rights (TRIPS) (1994).

9 Konwencja berneńska o ochronie dzieł literackich i artystycznych (1886, Akt paryski 1971)

10 WIPO Copyright Treaty (1996).

11 Patrz przyp. 1.

12 DIANE ROWLAND, ELIZABETH MACDONALD, INFORMATION TECHNOLOGY LAW, 29 (Cavendish Publishing Ltd, London, Sydney, 2nd ed., 1997).

stosowane dość rygorystycznie i ogólnie rzecz biorąc, elementy programów poza jego kodem źródłowym i wynikowym bardzo rzadko mogą korzystać z ochrony. Oczywiście nie dotyczy to tych elementów, które są samodzielnymi utworami, jak na przykład graficzne prezentacje, muzyka, fabuła gry komputerowej.

Nie jest konieczne zagłębianie się w dyskusję o zakresie prawnoautorskiej ochrony, gdyż nie jest to zagadnienie specyficzne dla oprogramowania Open Source. Ważne jest jednak wyciągnięcie wniosku, że ochronę prawa autorskiego dość łatwo jest obejść, gdyż obejmuje ona głównie zapis instrukcji programu (kod źródłowy lub wynikowy), a nie konkretny jego rezultat. Taki sam rezultat może zostać osiągnięty za pomocą innych instrukcji, przed napisaniem których prawo autorskie nie jest już w stanie nikogo powstrzymać. Innymi słowy, ochrona prawnoautorska nie obejmuje idei programu i jego zachowania, będących głównym źródłem jego rynkowej wartości. Dodatkowo, zwraca się ona przeciwko kopiowaniu utworu, ale już nie wobec tak zwanej niezależnej twórczości. Oznacza to, że informatyk pozbawiony jakiegokolwiek kontaktu z kodem źródłowym innego programu, który napisze dokładnie taki sam program ma pełne prawo do jego kopiowania i rozpowszechniania.

Programy a prawo patentowe

To niecałkowite dopasowanie ochrony prawnoautorskiej do oprogramowania jest główną przyczyną, dla której producenci coraz częściej poszukują ochrony patentowej. Prawo patentowe chroni bowiem uprawnionego z patentu nie tylko przed konkurencją produktów podobnych czy ekwiwalentnych, ale zabrania także niezależnego wytwarzania opatentowanego przedmiotu lub procesu.[13] Na pierwszy rzut oka, porównanie ochrony prawnoautorskiej z patentową wydaje się nie mieć większego sensu z uwagi na inny przedmiot tej ochrony. Prawo autorskie, jak już opisano wyżej, chroni bowiem konkretne wyrażenie, formę utworu, podczas gdy patent rozciąga się ogólnie rzecz biorąc, na praktyczne zastosowanie innowacyjnego pomysłu przybierające formę materialnego produktu lub technicznego procesu. Jednak oprogramowanie z uwagi na swoją specyfikę może dość często korzystać z ochrony patentowej w sposób pośredni. Zwykle jest ono przedstawiane jako część wynalazku, odpowiedzialna za prawidłowe funkcjonowanie materialnej maszyny lub przebieg

[13] KENNETH NICHOLS, INVENTING SOFTWARE: THE RISE OF "COMPUTER-RELATED" PATENTS , 3 (Quorum Books, Westport, 1998).

procesu o technicznym charakterze.

Kryteria przyznawania ochrony patentowej oprogramowaniu są jeszcze bardziej płynne niż te wymagane przez prawo autorskie. Ich stosowanie budzi zatem tym większe kontrowersje. Dyskusja dotyczy przede wszystkim aspektu celowościowego, czyli teoretyczno-ideologicznych podstaw dla rozciągania ochrony patentowej na technologie informatyczne, a dopiero później techniczno-prawnych kwestii zastosowania konkretnych przepisów prawa. W dyskusji teoretycznej można wyróżnić bardzo dużo różnych stanowisk. Natomiast same przepisy prawa nie dają także wystarczających wskazówek. Oczywiście, dla uznania za wynalazek konieczne jest wypełnienie ogólnych wymogów patentowalności jak nowość, poziom wynalazczy i możliwość przemysłowego zastosowania. Jednak kryteria te odnoszą się jedynie do czegoś, co zalicza się do przedmiotu prawa patentowego, a nie we wszystkich systemach prawnych jest wystarczająco jasne czy przedmiot ten obejmuje oprogramowanie. TRIPS na przykład nakłada obowiązek na sygnatariuszy przyznawania patentów "in all fields of technology"[14], a oprogramowanie nie jest nigdzie wyraźnie wyłączone. Orzecznictwo amerykańskie (USA jest jednym z sygnatariuszy TRIPS) zezwala na patentowanie oprogramowania i metod prowadzenia działalności gospodarczej. Punktem granicznym jest tu nie tyle istnienie wynalazku jako rzeczy materialnej lub doprowadzenie przez niego do przekształcenia rzeczy materialnej, lecz jego "użyteczność".[15] Zdaniem sądów amerykańskich abstrakcyjne idee, które nie są użyteczne nie są również patentowalne.[16] W efekcie takiej interpretacji, w Stanach w roku 2000 obowiązywało już 40000 patentów na oprogramowanie, a liczba ta wzrasta rocznie o kilka tysięcy.[17]

W Europie, gdzie wiele państw, jak i sama Wspólnota Europejska, również należy do systemu WTO-TRIPS, kwestia patentowalności oprogramowania regulowana jest przez Europejską Konwencje Patentową (EPC).[18] Konwencja ta wyłącza spod swego zakresu programy komputerowe "jako takie".[19] Interpretacja tego przepisu przez Europejski Urząd Patentowy (EPO) nie wydaje się jednak być zbyt restryktywna, jako że do tej pory (2004)

[14] TRIPS Art. 27.

[15] *State Street Bank & Trust v. Signature Financial Services*, 149 F.3d 1368 (Fed. Cir. 1998), cert. denied, 119 S. Ct. 851 (U.S. Jan 11, 1999).

[16] 149 F.3d 1368, 1373.

[17] Peter Toren, *Software and Business Methods are Patentable in the US (Get over it)*, PATENT WORLD, September 2000, at 7, 8.

[18] Europejska Konwencja Patentowa (1973).

[19] EPC Art. 52.

urząd ten przyznał 30000 patentów na oprogramowanie a rocznie przyznaje takich patentów 3000.[20] Kluczowym elementem badanym przez EPO jest "efekt techniczny" oprogramowania i nadaje on patent, jeżeli zastrzeżenie patentowe jako całość odnosi się do czegoś więcej niż tylko niepatentowalne metody matematyczne, czynności umysłowe lub metody prowadzenia działalności gospodarczej.[21] Na forum Unii Europejskiej toczy się obecnie gorąca dyskusja wokół projektu Dyrektywy o patentowaniu wynalazków realizowanych komputerowo[22], przedstawianej jako próba określenia dokładnych zasad patentowalności oprogramowania wobec niejasnej praktyki EPO i państw członkowskich. Zdaniem innych, jak na przykład Foundation for Free Information Infrastructure (FFII), jest ona po prostu dążeniem do usankcjonowania tej praktyki.

Umowy licencyjne i prawo zobowiązań

Opis systemu prawnej ochrony oprogramowania musi być jeszcze uzupełniony o licencje. Licencja jest prywatnoprawną umową, w której osoba uprawniona z praw autorskich do utworu (licencjodawca) zezwala na korzystanie z nich osobie, której takie prawa nie przysługują (licencjobiorcy). Licencje stanowią przeważającą formę wykorzystywaną przy obrocie oprogramowaniem, podczas gdy obrót bardziej tradycyjnymi przedmiotami prawa autorskiego, jak na przykład książkami, odbywa się zwykle za pomocą jedynie umowy sprzedaży. Ściśle rzecz biorąc, jest to stan postulowany przez dystrybutorów oprogramowania. Z prawnego punktu widzenia bowiem, określenie, czy w danym przypadku dochodzi do sprzedaży, czy do udzielenia licencji, czy też do zawarcia obu tych umów jednocześnie nastręcza niejednokrotnie wielu problemów i rodzi poważne konsekwencje co do zakresu praw przysługujących użytkownikowi.

Licencje dążą do określenia praw i obowiązków użytkownika domyślnie regulowanych przez prawo autorskie, o czym dalej, ale też jak wszystkie inne umowy cywilnoprawne podlegają ogólnym regułom prawa zobowiązań. Z tego punktu widzenia przede wszystkim zwraca uwagę sposób, w jaki dochodzi do zawierania tych umów. W masowym obrocie, oprogramowanie oferowane jest użytkownikom końcowym zwykle w

[20] Foundation for a Free Information Infrastructure, *Software Patents in Europe: A Short Overview*, http://swpat.ffii.org/lisri/intro/index.en.html.

[21] Yannis Skulikaris, *Software-Related Inventions and Business-Related Inventions; A review of practice and case law in US and Europe*, PATENT WORLD, February 2001 at 26, 28.

[22] 6580/02 PI 10 CODEC 242.

postaci tak zwanych umów celofanowych (shrink-wrap licenses), czyli w pudełkach owiniętych folią z nadrukowaną informacją, że szczegółowe postanowienia licencji znajdują się w środku, a otwarcie pudełka i korzystanie z oprogramowania stanowi zgodę na prawa i obowiązki z nich wynikające. Coraz bardziej popularnym staje się obrót oprogramowaniem on-line. Przybiera on postać umów click-wrap, których postanowienia prezentowane są przed pobraniem programu lub jego instalacją, a użytkownik wyraża na nie zgodę przez kliknięcie w pole "Zgadzam się". Przedsiębiorcy Internetowi często również przedstawiają użytkownikom umowy w formie browse-wrap, które łączą zgodę na ich postanowienia z faktem przeglądania stron www i zwykle przybierają formę warunków użytkowania strony (terms and conditions). Umowy licencyjne stosowane w obrocie Open Source Software zawierają postanowienia, które za zgodę na poddanie się im uważają fakt korzystania przez użytkownika z uprawnień przez nie regulowanych (czyli kopiowanie, modyfikacja i rozpowszechnianie programu) i w związku z tym określane są w literaturze anglojęzycznej jako "use-based".

Kontrakty omawiane powyżej zaliczane są przez naukę prawa zobowiązań do umów adhezyjnych, czyli takich, których zawarcie następuje przez przystąpienie w formie, w jakiej jest ona przygotowana przez przedsiębiorcę, bez możliwości negocjacji. Systemy prawne w rozmaity sposób regulują ich ważność i egzekwowalność, co do zasady wymagają jednak odpowiedniego poinformowania konsumenta o umowie, istnienia możliwości wglądu w jej postanowienia, a przede wszystkim zakazują stosowania niektórych technik (np. drobnego druku) lub konkretnych klauzul zwanych abuzywnymi.

Teoretyczne podstawy prawnej ochrony własności intelektualnej

Przeznaczeniem umów licencyjnych jest regulacja praw i obowiązków należących do domeny prawa własności intelektualnej. System ten stworzony jest do realizacji konkretnych celów i wynika z pewnych teoretycznych założeń. Uważa się, że podstawy te w prawie amerykańskim różnią się od tych, na których oparte zostały regulacje kontynentalne. Punktem wyjścia prawa amerykańskiego jest tak zwana "incentive theory", nakazująca w ochronie autorów i wynalazców widzieć zachętę dla ich twórczości.[23] Monopole tworzone na ich rzecz przez prawo dają im możliwość czerpania zysków, a jednocześnie dostarczają społeczeństwu

[23] DREXL, przyp. 4.

dóbr niezbędnych dla jego kulturowego i cywilizacyjnego rozwoju. Ostatecznym celem systemu amerykańskiego jest postęp rozumiany jako dobro publiczne. Systemy kontynentalne akcentują raczej konieczność ochrony uprawnionych z praw własności intelektualnej i dlatego na przykład wyposażone są w specyficzne instytucje chroniące osobisty związek autora z dziełem. Jednak nawet i europejscy autorzy, oceniając skuteczność systemu odwołują się często do amerykańskiej koncepcji wyważania publicznego i prywatnego interesu w stosunku do własności intelektualnej.

Koncepcja ta oparta jest na następujących przesłankach. Społeczeństwo jest jak najbardziej zainteresowane stałą podażą innowacji (postępu), która składa się z wiedzy i form jej wyrażania lub praktycznego zastosowania. Wiedza w sensie nauki ekonomi jest dobrem publicznym, co oznacza, że ilość jej konsumentów pozostaje bez wpływu na jej zużycie oraz, że nie jest łatwe wykluczanie od konsumpcji poszczególnych jednostek (np. różnicowanie pomiędzy płacącymi za dobro i "gapowiczami"). Uważa się, że mechanizmy rynkowe same nie doprowadziłyby do powstania wystarczającej podaży wiedzy, gdyż w niedługim czasie wszyscy konsumenci zostaliby "gapowiczami" i żaden z producentów nie byłby w stanie osiągać zysku.

Prawo własności intelektualnej jest jednym z przykładów administracyjnej ingerencji na rynku wiedzy. Innymi sposobami są subsydia lub bezpośrednie zamówienia rządowe, których skuteczność uważa się jednak generalnie za mniejszą. Ingerencje te mają za zadanie zbilansować prywatną kontrolę nad innowacjami i wiedzą z publicznym wolnym dostępem do nich. Z ekonomicznego punktu widzenia regulowanym rynkiem jest rynek wiedzy, a nie towarów produkowanych z wykorzystaniem tej wiedzy i należy o tym pamiętać przy ocenianiu tego systemu. Innymi słowy, nie liczy się ilość towarów i usług dostępnych do konsumpcji produkowanych dzięki ingerencji na rynku, lecz zasób wiedzy, który dzięki temu trafia do publicznej wiadomości.[24]

Praktyczne wykorzystanie prawa w modelu proprietary

Wszystkie powyżej opisane uwarunkowania wywierają swoje piętno na prawną ochronę oprogramowania, które pod pewnymi warunkami może korzystać z dobrodziejstw

[24] A. Samuel Oddi, *An Uneasier Case for Copyright than for Patent Protection of Computer Programs*, 72 NEBRASKA LAW REVIEW 351, 368 (1993).

prawa autorskiego lub patentowego. Jednak żaden z tych reżimów nie jest "uszyty na miarę" oprogramowania. Program komputerowy "jako taki" nie może na przykład zostać opatentowany w Europie. Prawo autorskie natomiast nie obejmuje idei, niezbyt mocno chroni SSO programu, zezwala na ograniczoną dekompilację, a przede wszystkim na niezależną twórczość. Nie będzie zatem wystarczająco chroniło właściciela programu przed odebraniem mu źródła jego wartości, jakim jest rezultat i zachowanie programu. W związku z tym, oprócz dążenia przedsiębiorców działających na rynku oprogramowania do zwiększenia dostępności ochrony patentowej, dawno już sięgnęli oni po siłę, jaką w kontaktach z konsumentami daje prawo zobowiązań. Mówiąc wprost, licencje na oprogramowanie wykorzystywane są w celu odebrania użytkownikom praw, które przysługują im z mocy ustaw prawnoautorskich, a wynikają z takich instytucji jak wyczerpanie prawa czy dozwolony użytek.[25]

Często podaje się, że rozpowszechnianie oprogramowania z pomocą licencji jest konieczne, gdyż nie jest możliwe uruchamianie programu komputerowego bez naruszenia wyłącznych uprawnień autora do kopiowania. Rzeczywiście, w odróżnieniu do bardziej tradycyjnych przedmiotów objętych ochroną prawnoautorską, zawsze gdy dochodzi do skorzystania z programu, musi on zostać skopiowany (np. z nośnika do pamięci RAM, a później instrukcja po instrukcji do procesora). Jednak nie może być to fakt decydujący, zwłaszcza obecnie, gdy większość ustawodawców dawno uzupełniła prawo autorskie o przepisy zezwalające nabywcom legalnego oprogramowania na takie działania. Nawet wobec braku takich przepisów można by pokusić się o przychylniejszą interpretację choćby instytucji dozwolonego użytku, bez konieczności uciekania się do przygotowywania długich i skomplikowanych kontraktów.

Prawdziwym powodem, dla którego to licencjonowanie, a nie prosta umowa sprzedaży jest powszechną formą stosowaną w obrocie oprogramowaniem jest dążenie producentów do obrony przed wyżej wspomnianym "gapowiczostwem". Występuje ono co najmniej w dwóch rodzajach. Po pierwsze, łatwość kopiowania oprogramowania powoduje zjawisko piractwa i zmniejszenie bazy konsumentów danego producenta, czyli bezpośredni spadek zysków. Po drugie, konkurenci producenta wykorzystują słabość ochrony prawa autorskiego i jego dozwolenie na dekompilację aby poznać działanie i idee leżące u podstaw programu i tym samym "przejechać się na gapę" za jego wydatki poniesione na badania i

[25] Oczywiście nie może być mowy o pozbawianiu uprawnień wynikających z *iuris cogentis*.

rozwój. Pierwszym utrudnieniem dla "gapowiczów" są zabezpieczenia techniczne, które czynią niemożliwym skopiowanie oprogramowania bez użycia specjalistycznych narzędzi oraz praktyka rozprowadzania samego tylko kodu wynikowego, który zmusza "podglądaczy" do mozolnej dekompilacji i tym samym znacząco utrudnia dostęp do prawnie niechronionych elementów programu. Licencje stanowią drugi poziom zabezpieczeń; zawierają one wyraźne postanowienia zakazujące dekompilacji, kopiowania czy choćby nawet korzystania z oprogramowania na więcej niż jednym stanowisku.

W celu zapewnienia sobie jeszcze większej ochrony, producenci i dystrybutorzy oprogramowania często dodają do umów licencyjnych dodatkowe postanowienia, nie wynikające już z uwarunkowań prawnoautorskich, lecz także często spotykane w umowach stosowanych w masowym obrocie. Najważniejsze z tych klauzul to ograniczenia odpowiedzialności licencjodawcy oraz wyłączenie gwarancji i rękojmi w maksymalnym stopniu dopuszczonym przez prawo. Dążenie do zapewnienia sobie tak daleko idącej ochrony jest wynikiem wskazanej już właściwości oprogramowania do zawierania niespodziewanych błędów. Z uwagi na praktyczną niemożliwość przygotowania oprogramowania bez jakichkolwiek błędów, gdyby producenci mieli odpowiadać choćby za ułamek szkód powodowanych przez ich produkty woleliby raczej w ogóle nie podejmować się tej działalności. Ewentualnie, dążąc do ekonomicznego ubezpieczenia się od odpowiedzialności za szkody podnosiliby ceny i przenosiliby w ten sposób ryzyko na użytkowników. Zobowiązanie licencjodawców do rezygnacji z dochodzenia odpowiedzialności i odszkodowania pozwala im na utrzymywanie cen na akceptowalnym przez dużą część użytkowników poziomie i jest kolejnym przykładem w jaki sposób umowy licencyjne wykorzystywane są do ochrony interesów producentów i dystrybutorów.

Można jeszcze wspomnieć, że adhezyjny charakter licencji pozwala licencjodawcom maksymalnie wykorzystać omówione powyżej "efekty sieciowe". Dzięki przygotowaniu jednego wzorca umownego przedstawianego dziesiątkom tysięcy klientów bez potrzeby indywidualnych negocjacji przedsiębiorcy są w stanie zbudować sporą bazę użytkowników i zdobyć pozycję rynkową w krótkim czasie.

Jak widać z powyższego opisu, aktualny system prawnej ochrony oprogramowania jest wykorzystywany głównie dla ochrony prywatnych interesów przedsiębiorców -

producentów i dystrybutorów. Pomimo podstawowego celu prawa własności intelektualnej, jakim jest wyważenie prywatnej kontroli nad wiedzą z publicznym dostępem do niej, praktycznym efektem jest prawie całkowity brak takiego dostępu, zwłaszcza biorąc pod uwagę współistnienie prawnych i technicznych mechanizmów ochrony. Użytkownicy przyjmują pasywną rolę konsumentów, a rynek oprogramowania kontrolowany jest przez stronę podaży. Od uznania deweloperów zależy dostarczenie poprawek błędów oprogramowania, czy wyposażenie go w dodatkową funkcjonalność. Podobna rola konsumentów na rynkach tradycyjnych materialnych dóbr jest być może zrozumiała, lecz takie towary nie mogłyby zostać łatwo i szybko dopasowane do potrzeb użytkowników. W przypadku oprogramowania byłoby to możliwe, gdyby tylko nie były wykorzystywane prawne i techniczne mechanizmy, które to uniemożliwiają.

Free Software a Open Source

Większość powyższych uwag dotyczących wykorzystania prawa w praktyce obrotu oprogramowaniem jest prawdą w stosunku do tak zwanego proprietary software, a niniejszy artykuł ma na celu przedstawienie systemu prawnej ochrony jego przeciwieństwa, czyli Open Source Software. Jednak ruch Open Source działa w ramach tego samego systemu prawa własności intelektualnej, chociaż jego cele w ich czystej postaci są dokładnym zaprzeczeniem proprietary software. Zatem, podczas rozważań ideologii i racji ruchu Open Source, które nastąpią poniżej, należy mieć na uwadze wszystkie dotychczasowe uwagi.

Dla zrozumienia ruchu Open Source konieczne jest wyjaśnienie różnicy pomiędzy “Open Source” i “Free Software”. Ogólnie rzecz biorąc, oba te pojęcia odnoszą się do programów komputerowych, nad którymi użytkownicy mają całkowitą kontrolę. Mogą oni korzystać z programu w jakimkolwiek celu, modyfikować go oraz rozpowszechniać program i jego modyfikacje. Użytkownicy mają dostęp do kodów źródłowych programu, tylko wtedy bowiem mają oni faktyczną możliwość obserwacji jego działania, znalezienia przyczyn błędów, poprawienia ich oraz dalszego rozwijania programu poprzez dodawanie dodatkowych funkcji. Wobec tego zarówno “Open Source” jak i “Free Software” odróżniają się od proprietary software, które rozpowszechniane są jedynie w kodzie wynikowym opatrzonym licencją zakazującą wszelkiego kopiowania i modyfikacji. “Open Source” i “Free Software” różnią się także od public domain software, czyli programów, w stosunku do których autorzy

zrzekają się praw autorskich lub które po prostu ochronie prawnoautorskiej nie podlegają.

Chociaż "Open Source" i "Free Software" mogą w zasadzie być używane zamiennie na oznaczenie tego samego zjawiska, istnieją pomiędzy nimi różnice formalne i historyczne. Określenie "Free Software" pojawiło się jako pierwsze i zostało zaproponowane przez Richarda M. Stallmana.[26] Uważa on, że korzystanie z oprogramowania, dzielenie się nim i wspólna praca nad jego rozwojem to naturalne prawa użytkowników. Publikacje tego autora zawierają dużo odniesień do etyki i ideałów takich jak wolność jednostki, swoboda wypowiedzi, przyjacielska współpraca. Stallman przedstawia czytelnikom utopijne wizje idealnej wspólnoty użytkowników komputerów, mających dostęp do dobrego oprogramowania. Porównuje on aktualny system ochrony oprogramowania do totalitarnego systemu Związku Radzieckiego. Nie wydaje się on być przeciwnikiem prawa własności intelektualnej w ogóle, a przynajmniej zgadza się z jego podstawowym celem, jakim jest dostarczanie społeczeństwu wiedzy i innowacji. Na pewno jednak, Stallman nade wszystko przedkłada ideały wolnej i otwartej współpracy, przyjacielskiego dzielenia się wiedzą i pomagania bliźnim.

"Open Source" jest określeniem, które zostało zaproponowane w momencie ogłoszenia przez firmę Netscape zamiaru udostępnienia kodu źródłowego ich przeglądarki internetowej. Za Open Source Initiative stoją osoby takie jak Eric S. Raymond czy Bruce Perens.[27] Nowe określenie na istniejące od dawna zjawisko ma według nich ujawnić praktyczne znaczenie udostępniania kodów źródłowych i doprowadzić do zwiększenia popularności tej idei wśród przedsiębiorców. Chociaż Stallman nigdy nie sprzeciwiał się czerpaniu zysku z oprogramowania, to jednak z lektury jego "GNU Manifesto" i innych publikacji jasno wynika, że stworzenie prosperującego przedsiębiorstwa produkującego Free Software nie jest jego największym marzeniem. A przecież w języku biznesu, to przyjacielskie dzielenie się, do którego nawołuje Stallman oznacza mniejsze koszty wykrywania błędów w oprogramowaniu oraz zbierania informacji o oczekiwaniach użytkowników. Wobec tego, Open Source Initiative odżegnuje się od prowadzenia moralnej

[26] Przedstawione poniżej poglądy Stallmana zaczerpnięto z: RICHARD M. STALLMAN, FREE SOFTWARE, FREE SOCIETY: SELECTED ESSAYS OF RICHARD M. STALLMAN (GNU Press, Boston, 2002); CHRIS DIBONA, SAM OCKMAN I MARK STONE (ED.), OPEN SOURCES: VOICES FROM THE OPEN SOURCE REVOLUTION (O'Reilly, 1999); LAWRENCE LESSIG, THE FUTURE OF IDEAS: THE FATE OF THE COMMONS IN A CONNECTED WORLD (Random House, New York, 2001).

[27] Przedstawione poniżej poglądy działaczy Open Source Initiative zaczerpnięto z http://www.opensource.org oraz publikacji Erica S. Raymond dostępnych tamże oraz pod http://www.catb.org/~esr/writings/.

krucjaty, a przedstawia się jedynie jako program marketingowy dla wolnego oprogramowania oparty na solidnych pragmatycznych podstawach. W swych publikacjach zwracają uwagę na takie korzyści wynikające z modelu Open Source, jak niezawodność, wysoka jakość czy niezależność od jednego dostawcy. Ich zdaniem, "Free Software" jest zbyt mylącym pojęciem, aby mogło przyciągnąć firmy działające dla zysku, głównie dlatego, że sugeruje brak tego zysku. Zdaniem Raymonda, jeżeli już nawet uniknie się dwuznaczności angielskiego słowa "free", to i tak "Free Software" za sprawą publikacji Stallmana jest zbyt często kojarzone z komunizmem, czy innymi pomysłami raczej nie cieszącymi się popularnością wśród biznesmenów. Nawet jeżeli nie jest to skojarzenie poprawne to "Open Source" ma przynajmniej na celu uniknięcie gorącej debaty ideologicznej. Open Source Initiative wyraźnie się od niej dystansuje deklarując, że nie ma stanowiska w sprawie tego, czy idee mogą być zawłaszczane, czy patenty na oprogramowanie są dobre i w innych podobnych kontrowersyjnych sprawach.

Free Software i Open Source z punktu widzenia prawnika

Różnice pomiędzy ruchem Free Software i Open Source wynikają przede wszystkim z definicji tych terminów, jakie każda z obu organizacji proponuje.[28] Możliwe jest porównanie samych definicji lecz nie są one dokumentami prawnymi, choć używają języka normatywnego. Dla prawnika istotne znaczenie mają dopiero postanowienia konkretnych licencji, gdyż tylko one w wiążący sposób określają prawa i obowiązki użytkowników i tylko na ich podstawie można określić dany program jako proprietary lub nie. Ponieważ obie organizacje dokonują oceny różnych licencji pod kątem zgodności ze swoimi definicjami wydaje się, że porównanie ich wyników pozwoli stwierdzić czy z prawnego punktu widzenia "Open Source" rzeczywiście różni się od "Free Software". Takie porównanie ujawnia pewne różnice, ale są one nieznaczne. Co ważniejsze, wszystkie najbardziej popularne licencje, takie jak GNU GPL, GNU LGPL, BSD, MIT, Artistic License, Apache License czy Mozilla Public License, są uznawane za zgodne z obiema definicjami.

Stallman prawdopodobnie nie zgodziłby się na przejście do porządku nad tymi różnicami, gdyż postrzega on "Free Software" i "Open Source" jako dwa obozy polityczne w jednej społeczności wolnego oprogramowania. Pomimo, że jako głównego wroga wskazuje

[28] Patrz odpowiednio: http://www.fsf.org, http://www.opensource.org.

on oprogramowanie proprietary, to nie chce pójść za przykładem Open Source Movement i przestać podnosić argumenty etyczne lub choć trochę zrezygnować z "praw i wolności użytkownika". Zysk nie jest zbyt wysoko w hierarchii wartości Stallmana, a już na pewno nie przed wolnością i przyjaźnią.

Trzeba zatem przyznać, pomimo braku doniosłych różnic prawnych dwie organizacje różnią się wyraźnie pod względem ideologicznym. Obie jednak, tak jak i proprietary software, korzystają z tego samego systemu prawnej ochrony własności intelektualnej. Dokładnie rzecz ujmując, korzystają z tego systemu dla realizacji swoich różnych celów. Na przykład licencja GNU GPL opracowana jako umowa modelowa przez Free Software Foundation zawiera bardzo rygorystyczną klauzulę "copyleft", która ogólnie rzecz biorąc ma na celu zobowiązanie informatyków do rozpowszechniania ich programów na licencji GNU GPL, jeżeli są one w określonym związku z programem już objętym tą licencją. Ten "wirus copyleft", jak bywa on niekiedy określany, ma doprowadzić do jak najszerszego rozpowszechnienia idei Free Software i uczynienie pisania proprietary software trudniejszym. Dla porównania, wielu producentów decydujących się na wykorzystanie modelu Open Source nie korzysta z klauzul "copyleft" lub proponuje je w znacznie złagodzonej formie. Model "Open Source" nie ma zatem na celu nakłanianie nikogo do jakiejkolwiek szczególnej ideologii. Ponadto, Free Software Movement dąży do opracowania całego systemu wolnego oprogramowania i z tego powodu dokładnie bada licencje, czy zezwalają one na łączenie objętych nimi programów. Free Software Foundation informuje na przykład, że niektóre licencje, chociaż zgodne z definicją Free Software są "niekompatybilne z GNU GPL", co oznacza że programy nimi objęte nie mogą być zgodnie z prawem połączone z programami rozpowszechnianymi na warunkach GNU GPL. Open Source Initiative ogranicza się do badania, czy licencje są zgodne z ich definicją i nie różnicuje pomiędzy nimi, prawdopodobnie dlatego, że przedsiębiorcy nie są zainteresowani tak zaawansowaną współpracą, jaką Stallman chciałby widzieć pomiędzy wszystkimi użytkownikami komputerów.

Złożoność relacji uczestników ruchu Open Source

Ruch Open Source w szerokim tego słowa znaczeniu, obejmującym oba powyżej opisane obozy, jest bardzo różnorodny. Pomysł leżący u jego podstaw, czyli upublicznienie

swojej pracy i zaproszenie innych do dzielenia się uwagami oraz do wnoszenia swojego wkładu pozwala na współistnienie podmiotom bardzo różniącym się między sobą. Zalążkiem ruchu było środowisko akademickie, takie jak to, którego częścią był Stallman przed swoim odejściem z MIT, oparte z definicji na otwartej współpracy studentów i profesorów. Społeczność naukowa dała początek hakerom, którzy swój dalszy rozwój zawdzięczają przede wszystkim Internetowi i obecnie zaliczają do swych szeregów wielu zdolnych informatyków lubiących po prostu wspólnie rozwiązywać trudne zadania. Dopiero później, zwabieni korzyściami, na które zwraca uwagę Open Source Initiative, do ruchu dołączyli przedsiębiorcy. Zatem, ogólnie rzecz biorąc, wszyscy uczestnicy Open Source mogą zostać podzieleni według kryterium ich motywacji na graczy non-profit i tych, którzy wykorzystują ten model produkcji dla zysku.

Upraszczając, grupę non-profit stanowią właśnie hakerzy i to oni właśnie cechują się najbardziej bogatym dorobkiem kulturowym. Interesującym jego przykładem jest dystrybucja systemu GNU/Linux, Debian.[29] Dystrybucja ta jest tworzona pod rządami "Debian Social Contract" (duża część tego dokumentu, "Debian Free Software Guidelines" posłużyła za wzór Open Source Definition promowanej przez Open Source Initiative). "Umowa społeczna Debiana" zawiera reguły pracy nad systemem oraz jego rozpowszechniania, lecz trudno jednoznacznie uznać ją za dokument prawny. Jest raczej proklamacją misji organizacji czy też jej statutem, a sami uczestnicy projektu określają ją jako swoją "Konstytucję". Z drugiej jednak strony, niektóre jej części adresowane są do użytkowników i zawierają pewne informacje na temat jakości oprogramowania.

Prawo czy normy moralne?

Być może postanowienia "Umowy społecznej Debiana" mogłyby być podstawą roszczeń prawnych, nie jest to jednak najważniejsze. Jest o wiele bardziej ciekawe, że stanowi ona przykład specyfiki reguł rządzących społecznością hakerską. Czasami zasady te wyrażane są na piśmie, Raymond pracuje nawet nad "Kodeksem hakerskim", jednak tak czy inaczej mają one zwykle charakter norm moralnych, a hakerzy są im posłuszni niezależnie od ich prawnej egzekwowalności. Dla porównania, inna niekomercyjna dystrybucja Linuxa,

[29] http://www.debian.org.

Slackware[30] nie ma żadnej specjalnej konstytucji, misji, czy filozofii a ich strona internetowa jest bardziej skoncentrowana na aspektach technicznych. Niemniej jednak, istnienie niezwykle rozbudowanych instytucji społecznych w ramach ruchu Open Source jest faktem i musi być brane pod uwagę w rozważaniach prawniczych. Amerykański prawnik, Robert W. Gomulkiewicz, podnosi, że nawet licencja GNU GPL jest tylko po części kontraktem, gdyż oprócz tego nosi charakter programu politycznego, traktatu filozoficznego, czy też poradnika dla nie-prawników.[31]

Samo prawo jest traktowane przez hakerów w specyficzny sposób, który może wydać się nieco dziwny dla prawników. Oczywiście istnieją podobieństwa pomiędzy językiem prawa a językiem programowania, jednak informatycy niejednokrotnie traktują dokumenty prawne wprost jak programy komputerowe. Powszechną praktyką jest "upgrade" licencji Open Source przez wydawanie ich kolejnych wersji numerowanych tak jak kolejne wersje oprogramowania. Ponadto, hakerzy żywo dyskutują kompatybilność swoich licencji, możliwość ich łączenia itp.

Oprócz wyżej przedstawionych, istnieje w kulturze hakerskiej wiele interesujących zwyczajów społecznych, które zostały dobrze opisane przez Raymonda w jego "Homesteading the Noosphere". Najważniejszy wniosek, jaki można wyciągnąć z tej lektury jest taki, że wspólnota hakerska i ich współpraca nie jest wcale taka swobodna i niezorganizowana, jakby to mogła sugerować jej przyjacielska i nie przesycona chęcią zysku atmosfera. W drodze społecznej ewolucji doszło do powstania organizacyjnej hierarchii oraz sformalizowanych procedur. Na przykład, postanowienia licencji Open Source zwykle nie zawierają zakazów "rozwidlania" (forking) projektów i mogłoby się wydawać, że jest to zjawisko częste wobec przyznania każdemu prawa do modyfikacji i rozpowszechniania programu. Jednak w całej historii Open Source nie doszło do wielu "rozwidleń" (z ważniejszych projektów należy wymienić rodzinę systemów BSD). Zdaniem Raymonda dzieje się tak, gdyż kultura hakerska wykształciła specyficzne pojęcie własności. Zwykle tylko jedna osoba lub mała grupa uznawana jest przez społeczność za "właścicieli" projektu, jego "koordynatorów". Oni decydują o jego dalszym rozwoju i oni są "uprawnieni" do

[30] http://www.slackware.org.

[31] Robert W. Gomulkiewicz, *De-bugging Open Source Software Licensing*, 64 UNIVERSITY OF PITTSBURG LAW REVIEW 75, 92 (2002).

rozpowszechniania kolejnych wersji programu. Wobec tego, duża część uprawnień z licencji Open Source pozostaje "w rezerwie" na wypadek, na przykład porzucenia projektu przez jego koordynatorów. W przypadku proprietary software, rozwiązanie firmy zajmującej się konkretnym projektem oznacza zwykle jego koniec. W świecie Open Source, możliwe jest "odziedziczenie" projektu przez innych bez konieczności powtarzania całej dotychczasowej pracy twórczej. Jednym z dobrodziejstw tego systemu, a prawo pełni w nim znaczącą rolę, jest więc zachowywanie dorobku jednostek dla całej społeczności.

Prawno-ekonomiczna praktyka ruchu Open Source

Głównym celem tego systemu jest jednak obrona przed "wrogiem nr 1", czyli proprietary software. Największym zagrożeniem dla Open Source jest mianowicie zawłaszczenie wolnego oprogramowania i zaprzestanie udostępniania kodów źródłowych modyfikacji. Ponieważ dostępność kodów źródłowych decyduje o wszystkich korzyściach płynących z Open Source Software, ich zamknięcie przed publicznością oznaczałoby po prostu koniec danego projektu, a przynajmniej koniec dobrowolnej pracy hakerów nad nim. Wobec tego zagrożenia, normy moralne zakazujące zatajania kodu źródłowego są szczególnie silne i tu też istnieje duże prawdopodobieństwo wytoczenia przez społeczność Open Source znacznie skuteczniejszej broni prawniczej. Niedawno doszło w Niemczech do postępowania sądowego w takiej właśnie sprawie, gdy jedna z firm złamała GNU GPL nie publikując kodów źródłowych modyfikacji, których dokonała w oryginalnym programie Open Source.[32] W obronie projektu wystąpił jego koordynator a efektem jego akcji było udostępnienie tychże kodów. Prawo nie zostało jednak wykorzystane do obrony jedynie prywatnych interesów danego dewelopera, jak to się dzieje w przypadku proprietary software, lecz całej społeczności, przed "obrabowaniem" jej z prawa dostępu do kodów źródłowych.

Postępowanie sądowe w przypadku Open Source Software jest jednak rzadkością i najciekawsze jest to, że system ten działa praktycznie bez takich tarć. W skali światowej było do tej pory wiele spraw sądowych dotyczących naruszenia postanowień licencji proprietary, podczas gdy autorowi tego artykułu nieznane jest żadne orzeczenie dotyczące bezpośrednio licencji Open Source, oprócz przedstawionego powyżej orzeczenia niemieckiego.[33] Zdaniem

[32] Landgericht München, 19.5.2004, 21 O 6123/04, nieoficjalne tłumaczenie na język angielski jest dostępne pod: http://www.groklaw.net/article.php?story=20040725150736471.

[33] Co prawda, sądy amerykańskie zetknęły się z zagadnieniem przynajmniej trzy razy: *Berkeley Software*

Ebena Moglena, który jest jak najbardziej właściwą osobą jeżeli chodzi o kwestie stosowania i egzekwowania licencji GNU GPL, Free Software Foundation nie była nigdy zmuszona do wnoszenia pozwów i jakiekolwiek naruszenia licencji zawsze były naprawiane po prostym wezwaniu lub krótkich negocjacjach.[34] Moglen twierdzi, że naruszyciele mają po prostu więcej korzyści z trzymania się postanowień licencji i wynikającej z tego możliwości pełnej kontroli nad oprogramowaniem, niż z prób jego zawłaszczenia i ryzykowania odpowiedzialności wobec uprawnionych z praw autorskich.

Hakerzy uczestniczą w projektach Open Source głównie dlatego, że po prostu potrzebują narzędzi, które sami tworzą (szczególnym tego przykładem jest Apache Server), chcą pomagać innym w swoim wolnym czasie lub też poszukują uznania i zależy im na zdobyciu autorytetu wśród innych informatyków. Oczywiste jest, że powód dla którego w ruch Open Source włączają się przedsiębiorcy, to zysk. Okazuje się zatem, że jest możliwe współistnienie tak odmiennych interesów, ale w efekcie nie dochodzi do rozdzielenia wyraźnych ról pomiędzy przedsiębiorców - producentów i użytkowników - konsumentów, tak jak w przypadku rynku oprogramowania proprietary. Wszyscy uczestniczą w rozwoju Open Source Software na równych zasadach, a odpowiednie zastosowanie systemu prawnego czyni to uczestnictwo atrakcyjnym dla każdego.

Prowadzenie prosperującej działalności gospodarczej wymaga dość abstrakcyjnego i twórczego myślenia. Przede wszystkim, jest raczej trudno uczynić produkt z wolnego programu jako takiego i przedsiębiorcy zwykle uciekają się do jednego z "pośrednich" modeli opisanych przez Raymonda w "The Magic Cauldron". Jednak nawet teraz, w rejonach świata gdzie dostęp do Internetu nie jest za dobry, można znaleźć spory popyt na wolne oprogramowanie utrwalone na materialnych nośnikach i użytkownicy gotowi są zapłacić za nie, nawet jeżeli jest ono jednocześnie dostępne za darmo na serwerze FTP. Mogą oni zapłacić nawet więcej, jeżeli oprogramowanie zostało wcześniej odpowiednio skonfigurowane, przetestowane lub opatrzone gwarancją. Przedsiębiorcy Open Source

Design, Inc. v. Unix System Laboratories, 1993 Copr.L.Dec. P 27,075, (27 U.S.P.Q.2d 1721); 27 U.S.P.Q.2d 1721; 1993 Copr.L.Dec. P 27,166, (86 Ed. Law Rep. 738, 29 U.S.P.Q.2d 1561); *Progress Software Corp. v. MySQL AB*, 195 F. Supp. 2d 328 (D. Mass. 2002) (preliminary injunction); *Planetary Motion, Inc. v. Techsplosion, Inc.*, 261 F.3d 1188 (11th Circ. (Fla.), 2001), jednak dwie pierwsze sprawy zakończyły się ugodami a w sprawie ostatniej sąd odniósł się do GNU GPL jedynie w *obiter dictae*, gdyż istota sporu dotyczyła naruszenia praw ze znaku towarowego.

[34] Eben Moglen, *Free Software Matters: Enforcing the GPL*, I, II, http://moglen.law.columbia.edu/publications/.

zajmują się zatem dodawaniem wartości do wolnego oprogramowania na wszelkie możliwe sposoby. Alternatywą jest wykorzystywanie Open Source Software do wspierania sprzedaży innych produktów, zaprojektowanych aby działać wraz z nim. Taką strategię realizują na przykład producenci serwerów, kiedy wspierają rozwój projektu Apache.

Te cele komercyjne realizowane są z wykorzystaniem ogólnie rzecz biorąc tego samego systemu prawnej ochrony, co system zaprojektowany przez hakerów. Oczywiście, istnieją różnice nie do pogodzenia, stanowiące główny powód tworzenia przez firmy własnych wzorców umownych zamiast zastosowania jednej z popularnych umów modelowych, jak GNU GPL czy BSD. Firmy nie prowadzą bowiem światowej krucjaty w imię idei wolnego oprogramowania, lecz skromnie mają na celu jedynie dobro swoich akcjonariuszy. Open Source Software jest dla nich jedynie środkiem, a nie celem samym w sobie i zwykle są zainteresowani jakiegoś rodzaju równowagą pomiędzy tym modelem produkcji a proprietary software.

Jednym z bardzo popularnych pomysłów jest dążenie do opanowania obu tych rynków i rozpowszechnianie oprogramowania na zasadach wielu licencji jednocześnie. Przykładu dostarczają projekty takie jak Mozilla czy OpenOffice.org, pozwalające użytkownikom wybrać, jakich zasad chcą oni przestrzegać. Jedną z licencji do wyboru jest zwykle GNU GPL, która jak najbardziej odpowiada deweloperom Open Source. Ci, którzy obawiają się, na przykład, “wirusa copyleft” mogą otrzymać to samo oprogramowanie na zasadach innej licencji i dzięki temu połączyć je ze swoją własnością intelektualną bez ryzykowania konieczności udostępnienia jej publicznie. Sam koordynator projektu może zresztą rozpowszechniać swój program także jako proprietary, być może z dodatkową funkcjonalnością. Nie jest to całkowicie zgodne z ideałami tego ruchu, ale cel ekonomiczny jest oczywisty: doprowadzenie do korzystania z programu przez jak największą ilość użytkowników i wykorzystanie “efektów sieciowych”.

Open Source jest poważnym atutem w walce przedsiębiorców z konkurencją, uniemożliwiając jej osiągnięcie dominującej pozycji na danym rynku. Przedsiębiorca, który oprócz udostępniania oprogramowania za darmo, dodatkowo jeszcze upublicznia jego kod źródłowy, zyskuje zwykle dużą popularność. Niektórzy posuwają się jeszcze dalej, gdyż widzą w Open Source drogę do zdobycia pozycji rynkowej dla samych siebie. Rzeczywiście,

jednym z podstawowych efektów klauzul “copyleft” jest to, że wszystkie modyfikacje i ulepszenia programu dodane przez innych muszą i tak wrócić do autora oryginału. To pozwala na utrzymanie konkurencji pomiędzy graczami i walkę przez proponowanie lepszych usług dodatkowych związanych z oprogramowaniem. Podobny cel przyświeca firmie Sun, autorowi licencji Sun Industry Standards Source License. Pozwala ona na zawłaszczenie oprogramowania, jeżeli tylko jest ono zgodne ze standardami publikowanymi przez autora oryginału. Jednak wszelkie modyfikacje niezgodne ze standardami muszą być opublikowane, przez co Sun zachowuje pewien rodzaj kontroli nad rynkiem a zysk czerpie głównie ze sprzedaży oprogramowania bazującego na standardzie.

Zamiast zakończenia

Jak wynika z powyższych rozważań, istnieją poważne różnice pomiędzy celami, dla których z prawnej ochrony oprogramowania korzystają producenci pracujący w modelu proprietary, a tymi, które realizują uczestnicy ruchu Open Source. Podstawowym celem tych pierwszych jest zapewnienie samym sobie pozycji rynkowej. System stworzony przez tych drugich jest wyraźnie zorientowany na ochronę interesów całej społeczności, choć daje również duże szanse sprytnym jednostkom kierującym się chęcią zysku, zdających sobie sprawę, jak dużą przewagę daje otwarcie kodów źródłowych. Jednak przede wszystkim, system ten chroni zwykłych użytkowników, którzy nie są już na łasce producentów i nie muszą pozostawać biernymi konsumentami.