Uniwersytet Ekonomiczny w Krakowie

Wydział Zarządzania

Przemysław Kulczycki

Analiza porównawcza modeli licencjonowania oprogramowania

Praca magisterska

Promotor:
dr Barbara Nowarska

Kraków 2008

Spis treści

Wstęp

Większość użytkowników oprogramowania nie zastanawia się nad jego licencją. Intuicyjnie traktują je jak inne dobra (przede wszystkim dobra materialne), z których korzystają na co dzień bez żadnych ograniczeń. Jednakże, oprogramowanie różni się od dóbr materialnych. Jest dobrem niematerialnym, nie można go zużyć, można je eksploatować w nieskończoność. A dzięki technologii cyfrowej można je szybko kopiować przy praktycznie zerowym koszcie. Sam zakup oprogramowania też różni się od zakupu dóbr materialnych. Klient nie dostaje oprogramowania na własność, otrzymuje jedynie licencję – pozwolenie na jego używanie. Typowi klienci myślą, że mogą używać oprogramowania bez ograniczeń, tak jak kupowane dobra materialne. Tak jednak nie jest. Korzystanie z oprogramowania jest objęte pewnymi ograniczeniami. Ich nieznajomość nie zwalnia użytkowników od obowiązku ich przestrzegania i ponoszenia konsekwencji prawnych.

Każdy twórca oprogramowania udostępniając je innym osobom może dowolnie określić warunki korzystania z niego. Warunki te powinny być umieszczone w licencji oprogramowania, która określa kto i jak może z niego korzystać.

Tematyka licencjonowania oprogramowania zainteresowała mnie z dwóch powodów. Pierwszy, techniczno–praktyczny, to uzyskanie odpowiedzi na pytanie „co mogę dzięki oprogramowaniu osiągnąć i jakim kosztem?”. Drugi, to powód etyczno–filozoficzny: „dlaczego oprogramowanie jest objęte różnymi ograniczeniami?”. W literaturze naukowej tematyka ta jest częściej poruszana przez prawników, niż informatyków. Niektórzy prawnicy, nie będąc jednak informatykami, nie zawsze w pełni rozumieją na czym polegają problemy związane z użytkowaniem oprogramowania lub nieprecyzyjnie posługują się terminologią informatyczną.

Celem niniejszej pracy było rozpoznanie skomplikowanego procesu ewolucji zasad dystrybucji oprogramowania i porównanie dzisiejszych – powstałych w wyniku tego procesu – modeli licencjonowania oprogramowania. Na użytek badawczy postawiłem więc następujące pytanie: **jakie są zalety i wady istniejących dziś modeli oprogramowania?** Odpowiedź na to pytanie powinna być ciekawa i przydatna w praktyce.

Analizę tak postawionego problemu postanowiłem zacząć od poznania historycznych uwarunkowań powstania dzisiejszych modeli licencjonowania oprogramowania (rozdział 1). Prześledziłem tu historię powstawania modeli licencjonowania oraz przyczyny i skutki pojawienia się czterech najważniejszych nurtów w/w modeli. Opisałem je z punktu widzenia ich głównych przedstawicieli: Projektu GNU, Microsoftu, producentów oprogramowania shareware oraz twórców systemu BSD. Tą prezentację poprzedziłem wyjaśnieniami terminologicznymi w świetle zróżnicowanych na świecie systemów prawnych i handlowych.

W drugim rozdziale opisałem szczegółowo wszystkie cechy danych modeli licencjonowania oprogramowania, ich zalety i wady oraz problemy związane z danymi licencjami.

Trzeci rozdział zawiera zestawienie porównawcze wymienionych modeli ze względu na cztery wybrane przeze mnie kryteria: wolność i ograniczenia użytkownika, jakość oprogramowania, koszty jego zakupu i utrzymania oraz organizację produkcji.

1. Właściwości oprogramowania

1.1. Dylematy terminologiczne

Program komputerowy

Definicja programu komputerowego nie jest prosta. Zwłaszcza dla prawników stanowi nie lada wyzwanie. Zarówno polska ustawa o prawie autorskim i prawach pokrewnych[1] jak i dyrektywa Rady EWG w sprawie ochrony prawnej programów komputerowych[2] nie zawierają definicji programu komputerowego. Wg polskiego prawa program komputerowy zalicza się do:

> *„utworów wyrażonych słowem, symbolami matematycznymi, znakami graficznymi”[3].*

Traktowany jest zatem tak samo jak utwory literackie, publicystyczne, naukowe, czy kartograficzne. Dyrektywa rady EWG określa przedmiot ochrony w następującej definicji:

> *Zgodnie z przepisami niniejszej dyrektywy Państwa Członkowskie chronią prawem autorskim programy komputerowe w taki sposób, jak dzieła literackie w rozumieniu Konwencji berneńskiej o ochronie dzieł literackich i artystycznych. Do celów niniejszej dyrektywy pojęcie "programy komputerowe" obejmuje ich przygotowawczy materiał projektowy.*

W amerykańskim systemie prawnym program komputerowy posiada ścisłe odniesienie do dziedziny informatycznej. Tutaj występuje jako:

> *„zestaw instrukcji (rozkazów) przeznaczonych do użycia bezpośrednio, lub pośrednio w komputerze w celu osiągnięcia określonego rezultatu”[4].*

---

1 Ustawa z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych, Dz. U. 1994 nr 24 poz. 83.

2 Dyrektywa Rady EWG z dnia 14 maja 1991 r. w sprawie ochrony prawnej programów komputerowych (91/250/EWG). http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31991L0250:PL:HTML

3 Zob. ustawa o pr. aut.

4 „*Prawo Internetu*”, red. P. Podrecki, Wydawnictwo Prawnicze LexisNexis, Warszawa 2007, s. 433–434.

Rosyjski zaś system prawny w swojej definicji programu, oprócz w/w instrukcji, obejmuje także:

> *„dane przeznaczone dla komputera oraz przedstawienia audiowizualne generowane przez program”*[5].

Oprogramowanie

Encyklopedia PWN[6] definiuje oprogramowanie jako:

> *„ogół programów, w które wyposażony jest system komputerowy”.*

Encyklopedia portalu Interia.pl[7] nieco dokładniej je określa. To:

> *„zbiór programów komputerowych umożliwiających lub ułatwiających pracę z komputerem”.*

Najbardziej precyzyjną definicję oprogramowania podaje Wikipedia[8]:

> *„całość informacji w postaci zestawu instrukcji, zaimplementowanych interfejsów i zintegrowanych danych przeznaczonych dla komputera do realizacji wyznaczonych celów. Celem oprogramowania jest przetwarzanie danych w określonym przez twórcę zakresie. Oprogramowanie jest synonimem terminów program komputerowy oraz aplikacja, przy czym stosuje się go zazwyczaj do określania większych programów oraz ich zbiorów”.*

Ogólnie rzecz biorąc, oprogramowanie składa się z programów (najczęściej skompilowanych, rzadziej w postaci interpretowanych skryptów), dokumentacji, grafiki, dźwięków i wszelkich innych danych potrzebnych programom do działania (np. pliki konfiguracyjne, klucze szyfrujące).

5 Ibidem.
6 http://encyklopedia.pwn.pl/haslo.php?id=3951439
7 http://encyklopedia.interia.pl/haslo?hid=92188
8 http://pl.wikipedia.org/wiki/Oprogramowanie

1.2. Oprogramowanie jako przedmiot handlu

Sprzedaż oprogramowania różni się od handlu typowymi produktami. Oprogramowanie samo w sobie jest dobrem niematerialnym, a koszt wytworzenia kolejnej jego kopii jest bliski zeru. Możliwość łatwego skopiowania zakupionego oprogramowania wymusiła na producentach oprogramowania znalezienie nowych metod dystrybucji swojego produktu, gdyż tradycyjne zasady sprzedaży produktów materialnych nie pasowały do oprogramowania.

Licencjonowanie oprogramowania

Obecnie najpopularniejszym sposobem dystrybucji oprogramowania jest jego licencjonowanie. Klient kupując program nie staje się jego właścicielem. Twórca lub dystrybutor programu udziela mu jedynie licencji na jego używanie (podobnie jak w przypadku np. utworów muzycznych). W owej licencji bardzo dokładnie określone są sposoby i cele w jakich dany program może zostać użyty. Najczęściej licencja jest tekstem jednostronnie narzuconym przez producenta, który kupujący musi zaakceptować w całości, aby móc korzystać z oprogramowania. Większość takich licencji ogranicza liczbę komputerów, na których można zainstalować oprogramowanie, liczbę użytkowników którzy mogą go używać, a czasem także liczbę procesorów jaką może posiadać komputer, na którym owo oprogramowanie ma działać.

Standardowym elementem niemal każdej licencji oprogramowania jest klauzula wyłączająca odpowiedzialność producenta z tytułu używania oprogramowania przez licencjobiorcę, w której producent nie gwarantuje bezbłędnego działania oprogramowania, nie gwarantuje jego przydatności do jakichkolwiek celów, ani nie odpowiada za ewentualne straty spowodowane jego używaniem. Takie klauzule stały się konieczne, gdyż złożoność programów komputerowych powoduje, iż niemożliwe jest wyeliminowanie wszystkich błędów w dużych, nietrywialnych programach.

Licencje grupowe

Na rynku, wraz ze wzrostem sprzedaży oprogramowania, pojawiło się zapotrzebowanie na licencje wielostanowiskowe. Wszelkie organizacje chcące zakupić oprogramowanie dla wielu komputerów nie musiały już wielokrotnie kupować pojedynczych licencji. Dzięki temu sama procedura zakupu stała się prostsza, a jednostkowa cena niższa. Licencja wielostanowiskowa różni się od zwykłej licencji jednostanowiskowej zmniejszeniem, lub całkowitym zniesieniem ograniczenia dopuszczalnej liczby stanowisk, na których zakupione oprogramowanie może zostać zainstalowane.

Licencje OEM

Specyficzną odmianą licencjonowania oprogramowania są licencje OEM[9]. Są to licencje przeznaczone dla producentów sprzętu komputerowego, bądź gotowych zestawów komputerowych (komputerów biurkowych, a także laptopów), którzy sami instalują oprogramowanie na sprzedawanych przez siebie komputerach, lub dołączają je do sprzedawanego sprzętu (np. nagrywarki CD/DVD, karty graficzne), z zastrzeżeniem, że owo oprogramowanie może być tylko z nim używane.

Klient nie może oddzielić oprogramowania OEM od sprzętu, z którym je zakupił, nie może go oddzielnie używać, nie może go oddzielnie sprzedać, ani nie może odsprzedać sprzętu bez dołączonego oprogramowania OEM. W przypadku oprogramowania dołączonego do komputerów biurkowych i laptopów, wymiana jednego lub kilku komponentów komputera (np. płyty głównej) może spowodować utratę licencji. Zaś przy samym zakupie komputerów klient najczęściej nie ma do wyboru modelu bez oprogramowania (np. bez systemu operacyjnego). Przez to klienci mają ograniczony wybór i są zmuszani do zakupu oprogramowania, którego nie chcą lub nie potrzebują. Microsoft co prawda umożliwia zwrot pieniędzy za nieużywany, zakupiony wraz z komputerem system Windows, ale procedura

9 OEM (ang. *Original Equipment Manufacturer*) – organizacja sprzedająca pod własną marką produkty wytworzone przez inne firmy. Zob. http://pl.wikipedia.org/wiki/OEM

uzyskania tego zwrotu jest na tyle długa, że odstrasza większość osób chcących go uzyskać.

Licencje OEM z reguły są tańsze od wymienionych wcześniej typowych licencji, zwanych także oprogramowaniem pudełkowym (zakupionym w pudełku) lub wersjami BOX (od ang. *box* – pudełko).

Sprzedaż praw majątkowych

Znacznie rzadziej spotykaną formą sprzedaży oprogramowania jest sprzedaż praw majątkowych do programu. Taką formę sprzedaży spotyka się przy tworzeniu oprogramowania na zlecenie i każdy taki przypadek jest uregulowany osobną umową. W takich umowach twórca oprogramowania sprzedaje klientowi wszystkie prawa majątkowe do programu. Ta forma sprzedaży jest poza spektrum zainteresowania niniejszej pracy.

Oprogramowanie jako usługa

Kolejną unikalną formą sprzedaży oprogramowania jest traktowanie go jak usługę (ang. *SaaS – Software as a Service*) i udostępnianie jej przez Internet. Ów model staje się ostatnio coraz popularniejszy, gdyż pozwala obejść ograniczenia wielu licencji oprogramowania. Ta forma sprzedaży również jest poza spektrum zainteresowania niniejszej pracy.

Patenty

Niezależnie od w/w form handlu, ze sprzedażą oprogramowania mogą być także związane patenty. Właściciel patentu udziela klientowi licencji na używanie technologii objętej owym patentem. Amerykańskie prawo zezwala na patentowanie oprogramowania, chociaż nie istnieją w nim żadne specyficzne uregulowania tej dziedziny[10]. Obecnie w Polsce obowiązuje Konwencja o patencie europejskim, której art. 52 wyłącza programy komputerowe z możliwości udzielenia patentu[11]. Z tego względu ta forma ochrony prawnej znalazła się poza spektrum zainteresowania niniejszej pracy.

10 http://en.wikipedia.org/wiki/Software_patents_under_United_States_patent_law
11 Konwencja o udzielaniu patentów europejskich, Dz. U. 2004 nr 79 poz. 737. Zob. http://www.epo.org/patents/law/legal–texts/html/epc/2000/e/ar52.html

1.3. Historia rozpowszechniania oprogramowania

Chcąc w pełni zrozumieć dzisiejsze modele licencjonowania oprogramowania należy cofnąć się do lat 60–tych[12], kiedy pojawiły się pierwsze komputery dla rozwiązań biznesowych. Wtedy oprogramowanie stanowiło nieodłączną część produktu jakim był komputer i było dostarczane za darmo, razem z kodem źródłowym. Ówcześnie dostarczane programy zawierały dużo błędów, więc dostarczanie kodu źródłowego było konieczne, aby jego użytkownicy – wtedy głównie programiści – mogli je poprawiać.

Oprogramowanie było rozwijane przez pracowników firm oraz przez uniwersytety. Firmy bardzo chętnie udostępniały swoje oprogramowanie uczelniom, gdyż były one dla nich darmową siłą roboczą. Jednymi z takich akademickich programistów byli hakerzy[13] (jak sami siebie nazywali) z Laboratorium Sztucznej Inteligencji w Massachusetts Institute of Technology (w skrócie: [ang.] AI Lab). Hakerzy udostępniali swoje programy wszystkim, za darmo. Dzielenie się kodem źródłowym pisanych przez siebie programów było wtedy na porządku dziennym. Jeden z nich wspomina tamte czasy w następujący sposób: *„Nie nazywaliśmy naszego oprogramowania «wolnym oprogramowaniem», bo taki termin wówczas nie istniał. Ale właśnie takie ono było. Kiedy pracownicy innego uniwersytetu albo firmy chcieli przenieść jakiś program na swój system i korzystać z niego, to z chęcią się zgadzaliśmy. Kiedy widziało się kogoś korzystającego z nieznanego, interesującego programu, to zawsze można było poprosić o jego kod źródłowy – żeby go przeczytać, zmienić lub użyć jego fragmentów do stworzenia nowego programu[14]”.*

12 http://www.softwarehistory.org/history/d_60s.html
13 Słowo *haker* wtedy oznaczało osobę o dużych umiejętnościach informatycznych. Szczegółową definicję, wraz z opisami późniejszych przeinaczeń tego terminu podaje Wikipedia: http://pl.wikipedia.org/wiki/Haker_(slang_komputerowy)
14 R. M. Stallman, „*Projekt GNU*”, tłum. Grupa tłumaczy witryny Projektu GNU, http://www.gnu.org/gnu/thegnuproject.pl.html

W 1969 roku IBM, głównie z powodu obaw o proces antymonopolowy, zdecydował się oddzielić oprogramowanie od sprzętu komputerowego, który sprzedawał[15]. Wtedy też coraz więcej firm zajmujących się tworzeniem oprogramowania zaczęło ograniczać do niego dostęp – za pomocą kontraktów, prawa autorskiego, i umów NDA (ang. *non–disclosure agreement* – umowa o zachowaniu poufności). Z powodu podpisywania NDA i chęci szybkiego zysku programiści przestali udostępniać za darmo swoje programy i zaczęli w nich umieszczać różne zabezpieczenia mające ograniczać swobodne ich kopiowanie (np. „bomby czasowe" blokujące program po określonym upływie czasu).

Jedną z pierwszych oznak tego trendu można było zauważyć w *„Otwartym liście do hobbystów"* autorstwa Billa Gatesa, opublikowanym w 1976 roku w biuletynie Klubu Hobbystów Komputerowych (*Homebrew Computer Club)*: *„Kogo stać na to, żeby wykonywać profesjonalną pracę za darmo? Jaki hobbysta potrafi przez 3 lata programować, wyszukiwać błędy, dokumentować swój produkt i rozpowszechniać go za darmo?[16]"*

W tym momencie, w celu łatwiejszego zrozumienia wydarzeń, należy rozdzielić historię na 4 równolegle biegnące części.

1.3.1. Projekt GNU i ruch wolnego oprogramowania

W 1980 roku Richard Matthew Stallman – haker ówcześnie pracujący w AI Lab – próbował zdobyć kod źródłowy do sterownika drukarki laserowej (którą Xerox podarował MIT), aby dodać do niej funkcję informującą użytkowników o zacięciu papieru. Poprzednie drukarki były dostarczane z kodem źródłowym sterowników, więc możliwość ich poprawienia była dla Stallmana czymś naturalnym. Tym razem uległ on rozczarowaniu, gdyż odmówiono mu dostępu do owego kodu źródłowego. Chcąc uzyskać dostęp do tego kodu musiałby podpisać umowę NDA, co było niezgodne z jego własnym etosem hakera. To anegdotyczne wydarzenie było zwiastunem nadchodzących zmian *„Oprogramowanie stało się dla firm rzeczą tak ważną,*

15 http://www–03.ibm.com/ibm/history/history/decade_1960.html
16 Zob. http://www.blinkenlights.com/classiccmp/gateswhine.html; tłumaczenie własne.

że [firmy] uznały publikowanie kodów źródłowych za zbędne i niebezpieczne, gdyż mogło stać się dla konkurencji tanim źródłem pożądanych informacji[17]*."*

W 1981 roku firma Symbolics zatrudniła niemal wszystkich pracowników AI Lab. Z hakerskiej społeczności zostali tylko nieliczni, w tym Richard Stallman. Stanął on wtedy przed trudnym wyborem. Mógł podpisać NDA i dołączyć do świata zamkniętego oprogramowania. Mógł też wycofać się z branży komputerowej, żeby nie wykorzystywać swoich umiejętności do ograniczania wolności użytkowników. Żadne z tych rozwiązań nie było dla niego akceptowalne, więc postanowił poszukać innego wyjścia, takiego, które by pozwoliło odtworzyć społeczność programistów.

We wrześniu 1983 roku Stallman ogłosił rozpoczęcie prac[18] nad stworzeniem kompletnego, kompatybilnego z Uniksem[19] systemu operacyjnego będącego **wolnym oprogramowaniem** (więcej na ten temat w rozdziale 2.2), zwanego **GNU**[20] (ang. *Gnu's Not Unix* – Gnu to nie Unix). Na początku 1984 roku Stallman opuścił MIT, aby w pełni skupić się na pracy nad projektem GNU oraz żeby MIT – jako pracodawca – nie mogło sobie rościć żadnych praw do jego właśnie tworzonego dzieła. W 1985 roku założył Free Software Foundation[21] (fundację wolnego oprogramowania) – organizację non–profit skupiającą programistów tworzących wolne oprogramowanie i tworzącą prawne podstawy ruchu wolnego oprogramowania. Najważniejszym dziełem FSF jest GNU General Public License[22] (Powszechna Licencja Publiczna GNU, w skrócie **GPL**) – ogólna licencja, którą każdy programista może zastosować do swojego dzieła, w celu nadania mu statusu wolnego oprogramowania i ochrony wolności jego

17 S. Williams, „*W obronie wolności. Krucjata hakera na rzecz wolnego oprogramowania*" (w oryg. „*Free as in Freedom: Richard Stallman's Crusade for Free Software*"), tłum. K. Masłowski, wyd. Helion, 2003. Książka dostępna w Internecie: http://stallman.helion.pl

18 Wiadomość wysłana do grupy dyskusyjnej net.unix–wizards, obecnie dostępna w archiwum: http://groups.google.pl/group/net.unix–wizards/msg/4dadd63a976019d7

19 Unix był uniwersalnym i bardzo popularnym systemem operacyjnym stworzonym przez AT&T (American Telephone & Telegraph). Szczegółowa historia jego powstania została opisana w rozdz. 1.3.4. Zob. też http://www.bell–labs.com/history/unix/

20 The GNU Project – http://www.gnu.org/

21 http://www.fsf.org/

22 Więcej na jej temat w rozdz. 2.2.1.

użytkowników. Pierwsza jej wersja została wydana w 1989 roku. Wersja druga (GPLv2), wydana w 1991 roku, obecnie pozostaje najpopularniejszą licencją wolnego oprogramowania[23]. 29 czerwca 2007 roku została wydana trzecia wersja licencji GPL (GPLv3), ale postęp w jej przyjmowaniu jest powolny[24].

Przez lata programiści GNU tworzyli kolejne elementy swojego systemu operacyjnego (kompilator GCC, debugger GDB, biblioteka języka C) lub wykorzystywali istniejące (X Window System, system składu druku TeX), ale wciąż brakowało im jednego najważniejszego elementu – jądra systemu[25]. Stworzenie jądra, nazwanego później Hurd, okazało się trudniejsze niż początkowo przypuszczano. Projekt ciągle był w fazie głębokiego rozwoju, a nawet do dziś nie został jeszcze ukończony. Tą patową sytuację przerwało dopiero pojawienie się w 1991 roku nowego, niezależnie stworzonego jądra **Linux**[26], autorstwa fińskiego studenta Linusa Torvaldsa. Warto zauważyć, że początkowo Linux wydany był na własnej licencji zakazującej wykorzystywania komercyjnego, a dopiero rok później (w 1992 roku), został wydany na licencji GPLv2.

Po skompletowaniu systemu rozwój Linuksa i narzędzi GNU uległ przyspieszeniu. Powstały liczne dystrybucje systemu GNU z jądrem Linux. Jedne były rozwijane niekomercyjnie przez społeczności i fundacje (Debian, Slackware), a drugie były rozwijane komercyjnie przez firmy (Red Hat, Suse, Mandriva). Dzięki GPL i innym licencjom wolnego oprogramowania każdy mógł stworzyć swoją dystrybucję systemu GNU/Linux samodzielnie zbierając poszczególne komponenty systemu lub opierając się na jednej

23 Źródło: D. A. Wheeler, „*Make Your Open Source Software GPL–Compatible. Or Else*", p. 2, http://www.dwheeler.com/essays/gpl–compatible.html oraz statystyki serwisu Freshmeat: http://freshmeat.net/stats/#license

24 Dane z serwisu http://gpl3.palamida.com

25 W literaturze informatycznej spotyka się 2 pojęcia systemu operacyjnego: pierwsze odnosi się do głównego programu zarządzającego sprzętem i aplikacjami uruchamianymi na komputerze, drugie zaś rozciąga w/w pojęcie także na podstawowe programy wraz z nim dostarczane a sam program zarządzający nazywa jądrem (ang. *kernel*) systemu operacyjnego. W niniejszej pracy użyto drugiego pojęcia.

26 Z nazwą *Linux* też wiąże się kolizja nazewnictw: system stworzony z jądra Linux i komponentów systemu GNU zyskał popularność właśnie jako „*Linux*", a nie „*GNU*". Aby uniknąć kolizji nazw, taki system nazywa się „*dystrybucją Linuksa*", lub (wedle zaleceń FSF) *„GNU/Linux"*. Zob. http://www.gnu.org/gnu/linux–and–gnu.html

z istniejących dystrybucji[27]. Obecnie istnieje ok. 550 dystrybucji Linuksa i ciągle powstają nowe[28]. Dzięki otwartemu kodowi źródłowemu Linux został przeportowany (przystosowany do pracy) na większość architektur sprzętowych, dzięki czemu znalazł zastosowanie na serwerach, komputerach domowych, klastrach, superkomputerach, telefonach komórkowych, palmtopach, i innych urządzeniach[29].

Nie wszyscy zgadzali się z filozoficzną argumentacją Richarda Stallmana dla istnienia wolnego oprogramowania (patrz rozdział 2.2). Wielu twórców wolnych programów miało do niego znacznie bardziej pragmatyczne podejście. Dodatkowym problemem była dwuznaczność angielskiego terminu *free software,* który w powszechnym rozumieniu wielu osób oznacza tylko darmowe oprogramowanie. W języku polskim nie ma takiego problemu. Stallman definiując w języku angielskim wolne oprogramowanie musiał objaśniać, że słowo „*free"* oznacza „wolny", jak w słowie „wolność" (*free as in freedom*), a nie tylko „darmowy", jak darmowe piwo (*free as in free beer*).

W 1997 roku Eric Steven Raymond, znany w środowisku wolnego oprogramowania programista i autor eseju „*The Cathedral and the Bazaar*" (Katedra i Bazar[30]), był zaniepokojony tym, że bezkompromisowa postawa Stallmana odstrasza konserwatywnych biznesmenów. Chcąc zainteresować świat biznesu wolnym oprogramowaniem Raymond postanowił wymyślić dla niego nową, mniej ideologicznie nacechowaną nazwę. W porozumieniu z innymi osobami utworzyli termin o*pen source*[31] (dosłownie: „otwarte źródło", w języku polskim jest tłumaczony jako „otwarte oprogramowanie"[32]) i założyli organizację zajmującą się promocją otwartego oprogramowania –

27 Graficzne zobrazowanie tworzenia kolejnych dystrybucji w czasie można znaleźć na stronie http://futurist.se/gldt/
28 Dane z serwisu http://distrowatch.com
29 http://kernel.org/#whatislinux
30 Raymond napisał ten esej w 1997 roku. W 1999 roku wydawnictwo O'Reilly opublikowało go razem z innymi jego esejami w formie książki: E. S. Raymond, „*The Cathedral & The Bazaar. Musings on Linux and Open Source by an Accidental Revolutionary*", wyd. O'Reilly Media, Sebastopol, California 1999. Eseje z tej książki są także dostępne w Internecie: http://catb.org/~esr/writings/cathedral–bazaar/
31 Szczegółowa historia jest opisana na stronie http://opensource.org/history
32 Więcej na ten temat w rozdz. 2.2.

Open Source Initiative (OSI)[33]. Organizacja ta przyznaje znak handlowy „Open Source Initiative Certified"[34] programom, które są wydane na jednej z zaakceptowanych przez nią licencji. Działania Raymonda przyniosły pierwszy znaczący efekt w 1998 roku, kiedy firma Netscape Communications Corporation zdecydowała się udostępnić kod źródłowy swojej przeglądarki www Netscape Communicator[35].

Będąc bardziej jednoznaczny niż free software, termin open source szybko osiągnął dużą popularność przyćmiewając osiągnięcia Stallmana i założonej przez niego Free Software Foundation. Stworzenie OSI zostało przez niego uznane za rozłam i próbę wymazania go z historii. Zwolennicy terminu open source kładą nacisk na techniczną doskonałość kodu źródłowego oprogramowania, będącą efektem większej liczby programistów mogących go poprawić. Eric Raymond sformułował to w swoim Prawie Linusa (na cześć Linusa Torvaldsa): „*przy wystarczającej liczbie oczu wszystkie błędy są proste*"[36]. Poza tym, Raymond w swoim eseju zwrócił uwagę także na otwartość procesu tworzenia oprogramowania. Raymond porównał duże, scentralizowane projekty (niezależnie od otwartości ich kodu źródłowego) do budowy katedry, przy której wszystko jest robione pod dyktando głównego architekta. Zaś na przykładzie rozwoju Linuksa Raymond opisał nową metodologię tworzenia oprogramowania, którą nazwał bazarem. Metodologia ta opiera się na publicznym i zdecentralizowanym rozwoju oprogramowania[37]. Stallman zaś odrzuca takie pragmatyczne myślenie i stale przypomina, że podstawą dla stworzonej przez niego idei wolnego oprogramowania jest wolność. Choć oba ruchy mają odmienne filozoficzne podstawy swojego działania, to osoby do nich należące na co dzień potrafią ze sobą współpracować, a w krytycznych sytuacjach, takich jak pozwy

33 http://opensource.org/
34 W USA nie można zarejestrować terminu o*pen source* jako znaku handlowego, dlatego zdecydowano się użyć *OSI Certified.*
Zob. http://opensource.org/docs/certification_mark.html
35 http://wp.netscape.com/newsref/pr/newsrelease591.html
36 Źródło: E. S. Raymond, op. cit., s. 19 (tłumaczenie własne).
37 Więcej na ten temat w rozdz. 3.4.

sądowe SCO przeciwko firmom rozwijającym i używającym Linuksa[38], potrafią stanąć razem po jednej stronie.

1.3.2. Microsoft i zamknięte oprogramowanie

Firma Microsoft została założona w 1975 roku przez Billa Gatesa oraz Paula Allena i obecnie jest jedną z największych międzynarodowych korporacji zajmujących się produkcją i dystrybucją oprogramowania. W swojej historii Microsoft wywarł olbrzymi wpływ na praktyki dystrybucji i licencjonowania oprogramowania.

Pierwszym systemem operacyjnym Microsoftu był wydany w 1980 roku Xenix[39], odmiana Uniksa. Wartym odnotowania jest fakt, że był to system, na którym działała pierwsza wersja Microsoft Word. Xenix nie był sprzedawany bezpośrednio konsumentom, Microsoft sprzedawał go wyłącznie producentom OEM[40]. Pod koniec 1980 roku był najpopularniejszą odmianą Uniksa. W 1983 roku Microsoft udzielił firmie Santa Cruz Operations licencji na sprzedawanie Xeniksa. W połowie lat 80–tych Microsoft wycofał się z systemów uniksowych i sprzedał SCO wszystkie prawa do Xeniksa, w zamian za 25% udziałów w firmie.

Systemem, który przyniósł Microsoftowi sukces był DOS (ang. *Disk Operating System* – dyskowy system operacyjny). W 1981 roku Microsoft podpisał z IBM–em (najstarszym na świecie producentem sprzętu komputerowego) kontrakt na dostarczenie własnej wersji ówcześnie popularnego systemu operacyjnego CP/M do przyszłych komputerów IBM PC[41]. Do wypełnienia tego kontraktu Microsoft kupił system 86–DOS[42] od firmy Seattle Computer Products. Microsoft rozwijał ten system jako MS–DOS, a IBM sprzedawał go (razem ze swoimi komputerami) jako PC–DOS.

38 Zob. rozdz. 1.3.2.
39 Źródło: *The History of Computing Project*, http://www.thocp.net/companies/microsoft/microsoft_company.htm
40 Zob. rozdz. 1.2: Licencje OEM.
41 Ang. *Personal Computer* – komputer osobisty.
42 86–DOS był początkowo znany jako QDOS (ang. *Quick and Dirty Operating System* – szybki i niechlujny system operacyjny) – nazwa ta wzięła się od pośpiechu podczas pisania tego systemu.

Olbrzymi sukces jaki odniósł komputer IBM PC utorował Microsoftowi drogę do rynkowej dominacji. PC–DOS, będąc najtańszym z trzech oferowanych razem z IBM PC systemów operacyjnych[43] (pozostałe 2 to CP/M–86 oraz UCSD p–System), zdobył największą popularność. Otwarta, modularna architektura komputera[44] oraz użycie powszechnie dostępnych części (zamiast produkowania wszystkich części samodzielnie) spowodowały pojawienie się dużej liczby tańszych klonów komputerów PC produkowanych przez konkurencję, które były sprzedawane z systemem MS–DOS.

W 1983 roku Microsoft nawiązał współpracę z firmą Apple Computer i w rok później wydał 2 programy na ich komputery Macintosh: Word i MultiPlan (prekursor dzisiejszego Excela). Macintoshe wyróżniały się od PC–tów m.in. posiadaniem graficznego interfejsu użytkownika (GUI – ang. *Graphical User Interface*). Rok później Microsoft wydał swoją wersję GUI dla systemu DOS – Windows 1.0. W rok po wydaniu Windows 2.0, w 1988 roku, Apple wytoczyło Microsoftowi proces o skopiowanie interfejsu ich systemu MacOS. Po 4 latach procesu sąd odrzucił wszystkie zarzuty firmy Apple, stwierdzając m.in., że same pomysły na interfejs nie mogą być przedmiotem ochrony prawa autorskiego[45].

W 1985 roku IBM i Microsoft po raz kolejny podjęli współpracę, tym razem w celu utworzenia nowego systemu operacyjnego – OS/2[46]. IBM reklamował OS/2 w połączeniu ze swoim nowym (wypuszczonym w 1987 roku) komputerem PS/2, a później sprzedawał go także osobno. Chociaż OS/2 był kompatybilny z programami pisanymi pod DOSa i Windows (ale nie ze sterownikami sprzętu), to zamknięta architektura komputerów PS/2 oraz późniejszy sukces Microsoft Windows (będącego początkowo jedynie graficzną nakładką na DOSa) przyczyniły się do jego porażki. Dodatkową przyczyną porażki OS/2 było to, że IBM pobierał opłaty za zestaw narzędzi

43 http://wiki.oldos.org/Dos/OriginOfDos
44 Wiele elementów tej architektury przetrwało do dziś. Obecnie dominująca architektura ATX jest potomkiem IBM PC.
45 J. Myers, „*Apple v. Microsoft: Virtual Identity in the GUI Wars*", Richmond Journal of Law & Technology, 1999, http://law.richmond.edu/jolt/v1i1/myers.html
46 http://www.os2bbs.com/os2news/OS2History.html

do tworzenia oprogramowania (tzw. *Software Development Kit*, w skrócie SDK) pod OS/2, podczas gdy Microsoft rozdawał Windows SDK za darmo, co przełożyło się na ilość oprogramowania (a zwłaszcza sterowników) dostępnego pod oba systemy.

W 1990 roku wydanie Windows 3.0 przyniosło już Microsoftowi większe zyski niż OS/2, przez co firma zdecydowała się przeznaczyć więcej pracowników na jego dalszy rozwój. Spowodowało to konflikt z IBM–em, który chciał wciąż rozwijać OS/2. Firmy doszły do zgody i ustaliły, że IBM zajmie się rozwijaniem OS/2 2.0, mającego zastąpić OS/2 1.3 i Windows 3.0, a Microsoft zajmie się rozwijaniem jego następcy – OS/2 3.0. Wkrótce jednak (w 1991 roku) Bill Gates ogłosił pracownikom Microsoftu, że partnerstwo z IBM–em zostało zakończone. IBM kontynuował rozwój OS/2, a Microsoft przekształcił niedokończony projekt systemu OS/2 3.0 w Windows NT.

W latach 1992–1995 Microsoft umocnił swoją pozycję na rynku, m.in. dzięki instalowaniu przez producentów sprzętu DOSa i Windows na sprzedawanych przez nich nowych komputerach. Obok Windows, kolejnym produktem Microsoftu, który odniósł sukces na rynku był pakiet biurowy Microsoft Office. W 1993 roku Microsoft Windows stał się najczęściej używanym graficznym systemem operacyjnym na świecie[47]. W tym samym roku Microsoft wydał także pierwszą wersję Windows NT (wersja 3.1)[48] – systemu dla serwerów i stacji roboczych.

W 1995 roku Microsoft wydał kolejną wersję swojego systemu operacyjnego, Windows 95. Z technicznego punktu widzenia nowa wersja wciąż była tylko nakładką graficzną, ale nie wymagała wcześniejszego zakupu DOSa (tak jak wcześniejsze wersje Windows), bo sama go zawierała. Wsparty olbrzymią kampanią reklamową, Windows 95 również odniósł duży sukces.

Windows 95 nie posiadał żadnej przeglądarki www, bo Microsoft jeszcze żadnej nie napisał. Sukces sieci Internet był dla niego dużą

47 Źródło: *The History of Computing Project*, http://www.thocp.net/companies/microsoft/microsoft_company_part2.htm

48 Wersja systemu odpowiadała numerowi aktualnej wersji Windows.

niespodzianką i dopiero wtedy Microsoft kupił od firmy Spyglass licencję na ich przeglądarkę Mosaic, którą później rozwijał pod nazwą Internet Explorer. Przeglądarka ta została później zintegrowana z systemem Windows 95 (od wersji OSR1) i stała się standardowym komponentem kolejnych wersji systemu. Ruch ten wywołał w 1998 roku proces antymonopolowy z amerykańskim Departamentem Sprawiedliwości. Microsoft przegrał ten proces w 2000 roku, ale później złożył apelację (którą częściowo wygrał) i zawarł ugodę[49].

Rok 1998 przyniósł kolejne ważne dla Microsoftu wydarzenia. Firma wydała kolejną wersję swojego flagowego produktu – Windows 98. System zawierał wbudowaną przeglądarkę Internet Explorer 4.0, która oprócz swojej standardowej funkcji (przeglądanie stron www) odpowiadała także za wyświetlanie części interfejsu użytkownika. Przeglądarki nie można było normalnie z systemu odinstalować, co było jednym z argumentów w w/w procesie antymonopolowym. Drugim ważnym wydarzeniem tego roku był wyciek tzw. Dokumentów Halloween[50] – poufnych memorandów dotyczących potencjalnej strategii Microsoftu w odniesieniu do ruchu open source, a w szczególności Linuksa. Dokumenty te określają Linuksa i open source jako główne zagrożenia dla dominacji Microsoftu na rynku oprogramowania, a także sugerują sposoby ich zwalczania. Owe dokumenty były sprzeczne z wcześniejszymi publicznymi wypowiedziami Microsoftu, który odżegnywał się od takich zamiarów. Microsoft potwierdził ich autentyczność, ale zaprzeczył jakoby stosował się do strategii w nich opisanych. Kolejne dokumenty zaliczone do Dokumentów Halloween wyciekły w 2002 i 2004 roku. Dokument z 2004 roku to email jednego z pracowników SCO Group[51]

49 Szczegóły procesu są dostępne na stronie http://www.usdoj.gov/atr/cases/ms_index.htm

50 Dokumenty te dostępne są na stronie Erica Raymonda: http://www.catb.org/~esr/halloween/ Ich nazwa pochodzi od daty publikacji pierwszych z nich – okolic święta Halloween.

51 Firma Santa Cruz Operations, która w latach 80–tych kupiła od Microsoftu prawa do Xeniksa, w 2000 roku sprzedała prawa do wszystkich swoich systemów uniksowych firmie Caldera Systems, znanej w latach 90–tych z oferowania własnej dystrybucji Linuksa. SCO po tej sprzedaży zmieniła nazwę na Tarantella Inc., a Caldera później zmieniła nazwę na SCO Group.

ujawniający, że Microsoft finansował SCO, które wytoczyło IBM–owi proces[52], m.in. o to, że włączył do Linuksa kod źródłowy należący do SCO[53]. SCO groziło (i wciąż grozi) pozwami sądowymi także firmom używającym Linuksa.

Na początku roku 2000 Microsoft wydał Windows 2000, kolejny produkt z linii NT, a pod koniec tego samego roku wydał Windows Me, następcę Windows 98. Oba systemy zawierały odtwarzacz filmów Windows Media Player. Podczas gdy Windows 2000 był znacznym ulepszeniem, Windows Me okazał się porażką. Poważne problemy z jego stabilnością przyniosły mu miano najgorszego systemu Microsoftu.

W 2001 roku Microsoft wydał system Windows XP, będący połączeniem funkcjonalności dotychczasowych 2 linii produktów – biznesowej NT i domowej 9x. XP był pierwszym systemem, w którym pojawił się system aktywacji systemu przez Internet lub telefon – zabezpieczenie antypirackie[54]. Windows XP był głównym systemem Microsoftu przez 6 lat (najdłużej w historii firmy) i do tej pory jest najpopularniejszą wersją Windows.

W 1998 roku firma Sun Microsystems złożyła do Komisji Europejskiej skargę na Microsoft, związaną z nieudostępnianiem przez nich interfejsów systemu Windows NT. Z czasem sprawa[55] została poszerzona o inne technologie zintegrowane z systemem Windows, a do grona skarżących dołączyły inne firmy i organizacje – m.in. Time Warner (właściciel portalu America OnLine), Novell (właściciel Suse Linux), RealNetworks (producent

52 SCO jednocześnie wytoczyło procesy innym firmom (m.in. Novell), a Red Hat wytoczył proces SCO. Procesy te trwają do dziś, część zarzutów została odrzucona, a w sprawie innych zapadły wyroki. Najbardziej znaczącym jest wyrok z 2007 roku orzekający, że to Novell jest właścicielem praw autorskich do kodu Uniksa i UnixWare, a nie SCO. Treść wyroku: http://www.groklaw.net/staticpages/index.php?page=20070810205256644

53 Kwestia praw autorskich Uniksa i jego pochodnych jest bardzo zawiła, szczegółowe informacje na ten temat oferuje serwis Grokline: http://www.grokline.net/

54 Słowo *piractwo* tutaj funkcjonuje jako określenie na nielicencjonowane kopiowanie i używanie oprogramowania. Richard Stallman nie zgadza się z tym pojęciem i odradza jego używania, gdyż uważa porównywanie nieautoryzowanego kopiowania do mordowania ludzi na statkach wodnych za niesłuszne. Zob. R. M. Stallman, L. Lessig, J. Gay, „*Free Software, Free Society: Selected Essays of Richard M. Stallman*", wyd. GNU Press, Boston, Massachusetts 2002, s. 190–191. Dostępne w Internecie: http://www.gnu.org/philosophy/fsfs/rms-essays.pdf

55 Szczegóły sprawy są dostępne na stronach Komisji Europejskiej: http://ec.europa.eu/comm/competition/antitrust/cases/microsoft/

odtwarzacza Real Player) i Free Software Foundation Europe (europejski oddział FSF). Proces trwał wiele lat, i dopiero w 2003 roku zapadła pierwsza decyzja. Komisja Europejska nakazała Microsoftowi udostępnić szczegóły działania swoich protokołów sieciowych oraz zaoferować klientom wersję systemu Windows bez odtwarzacza Windows Media Player. Dodatkowo w marcu 2004 roku KE nałożyła na Microsoft grzywnę w wysokości 497 mln €. Microsoft złożył apelację, którą później przegrał, i później zapłacił grzywnę. W 2005 roku firma wydała Windows XP N (bez Windows Media Playera). Pod koniec tego samego roku KE, z pomocą niezależnego eksperta, stwierdziła, że Microsoft nie spełnił wszystkich wymagań wyroku i zagroziła kolejnymi grzywnami. Kilka tygodni później Microsoft udostępnił jako referencję (tylko do wglądu, bez prawa kopiowania) źródła dodatku Service Pack 1 do systemu Windows Server 2003. Owe źródła były udostępniane tylko członkom swojej grupy roboczej WSPP (Work Group Server Protocol Program), którzy za swoje członkostwo musieli zapłacić. W czerwcu 2006 roku KE nałożyła kolejną grzywnę 280,5 mln € za przeciwstawianie się wyrokowi i nieudostępnianie wymaganych informacji na sprawiedliwych warunkach. We wrześniu 2007 roku Microsoft przegrał ostatnią apelację, a w miesiąc później ogłosił, że nie będzie już więcej apelować i zastosuje się do wyroku. W efekcie Microsoft obniżył opłaty patentowe za dostęp do informacji o swoich protokołach, a dla projektów open source ustalił stałą jednorazową opłatę w wysokości 10000€.

W listopadzie 2006 roku Microsoft podpisał kontrowersyjną umowę patentową z firmą Novell[56], która produkowała dominujący w latach 80–tych serwerowy system operacyjny Netware[57], a teraz jest dystrybutorem Linuksa (Suse Linux). Umowa ta sprytnie ominęła sekcję 7 licencji GPLv2, która

56 http://www.microsoft.com/presspass/press/2006/nov06/11-02MSNovellPR.mspx

57 Netware od późnych lat 80–tych do połowy lat 90–tych był dominującym sieciowym systemem operacyjnym. Oparty na protokole IPX/SPX, oferował duża stabilność i wydajność. Netware miał tylko konsolowy interfejs i był trudny w konfiguracji, co było jednym z czynników późniejszego porzucania go na rzecz Windows NT. Drugim czynnikiem była popularyzacja opartego na TCP/IP Internetu. Novell zbyt późno zaimplementował protokół TCP/IP, przez co stracił większość rynku na rzecz innych systemów (Windows NT, Linux, BSD). Dziś Netware pozostał w użyciu tylko w dużych korporacjach (np. banki), które nie mogą sobie pozwolić na częste i ryzykowne migracje.

zabrania udzielania pojedynczych licencji patentowych dystrybutorom oprogramowania objętego tą licencją. Richard Stallman nazwał tę sekcję „wolność albo śmierć", bo właściciel patentu musi udzielić wolnej od tantiem jednakowej licencji wszystkim użytkownikom programu objętego licencją GPLv2, albo żadnemu. Microsoft i Novell, zamiast udzielić sobie licencji na swoje patenty, podpisali umowę o niepozywaniu klientów drugiej strony o naruszenie swoich patentów. Umowa objęła także współpracę obu firm w zakresie interoperacyjności ich produktów, oraz marketingu. Microsoft zapłacił z góry Novellowi 348 mln $ za prawa do ich patentów oraz kupony na subskrypcję (wsparcie techniczne) Suse Linux Enterprise Server, a Novell w ciągu 5 lat ma zapłacić ok. 40 mln $[58] (zależnie od wielkości zysków). Umowa odbiła się szerokim echem w środowisku wolnego oprogramowania, część użytkowników domowych zbojkotowała produkty Novella, a niektórzy jego pracownicy zrezygnowali z pracy w geście protestu. Zaś Free Software Foundation zapowiedziała zmiany w nadchodzącej trzeciej wersji licencji GPL, które miałyby zapobiec przyszłemu zawieraniu podobnych umów. Wydana w czerwcu 2007 roku GPLv3 co prawda nie blokuje wcześniej zawartej umowy Microsoft–Novell, ale może ją wykorzystać przeciw Microsoftowi, jeśli Novell do swojej dystrybucji dołączy oprogramowanie objęte GPLv3. Wtedy, po zrealizowaniu kuponu na SLES zakupionego od Microsoftu, zostałby on dystrybutorem oprogramowania objętego GPLv3 i musiałby rozszerzyć swoją ochronę patentową na wszystkich użytkowników danego programu objętego licencją GPLv3[59].

W maju 2007 roku przedstawiciele Microsoftu ogłosili, że Linux, OpenOffice, i inne programy należące do ruchu wolnego oprogramowania, łącznie naruszają 235 patentów należących do Microsoftu[60]. Microsoft nie wskazał o jakie patenty chodzi, nie wskazał o jaką funkcjonalność chodzi, ani nie wskazał konkretnych linii kodu (Linuksa i innych programów), które by naruszały jego patenty. Eben Moglen, prawnik przez długie lata

58 http://www.linux-watch.com/news/NS7235986827.html
59 http://news.zdnet.co.uk/software/0,1000000121,39287338,00.htm
60 http://money.cnn.com/magazines/fortune/fortune_archive/2007/05/28/100033867/

doradzający FSF, uważa, że nie liczą się liczby, tylko dokładna analiza każdego przypadku rzekomego naruszenia patentu[61]. Linus Torvalds (autor Linuksa) w wywiadzie dla tygodnika InformationWeek stwierdził[62], że istnieje znacznie większa szansa, iż to Microsoft narusza czyjeś patenty[63]. Torvalds, wraz z innymi członkami ruchu wolnego oprogramowania, namawia Microsoft do wskazania tych rzekomo naruszonych patentów, żeby mogły zostać przetestowane w sądzie (sąd orzekłby czy dane patenty są ważne), lub żeby programiści mogli usunąć naruszający je kod źródłowy. Na końcu wywiadu Torvalds wyraził opinię, że Microsoft woli nikogo nie pozywać, zwłaszcza użytkowników Linuksa, którzy są jednocześnie jego klientami, a nie ujawniając o które patenty chodzi, Microsoft może kontynuować swoją taktykę siania strachu, niepewności, i zwątpienia (tzw. FUD – od ang. *fear, uncertainty, doubt*).

1.3.3. Freeware, shareware i adware – darmowe oprogramowanie[64]

W 1982 roku amerykański programista Andrew Fluegleman, autor programu telekomunikacyjnego PC–TALK, wymyślił nietypowy (jak na owe czasy) sposób dystrybucji: rozdawał swój program użytkownikom i prosił, żeby za niego zapłacili, jeśli się im on spodobał. Fluegleman nazwał ten sposób dystrybucji *freeware* (od ang. *free* – darmowy) i zarejestrował go jako znak handlowy (ale z czasem z niego zrezygnował).

W tym samym roku kolejny amerykański programista Jim Knopf (pseud. Jim Button) wymyślił coś podobnego: rozdawał swój program EASY–FILE z prośbą o podzielenie się nim z innymi użytkownikami oraz

61 http://legalpad.blogs.fortune.cnn.com/2007/05/13/msft–linux–free–software–infringe–235–of–our–patents/

62 http://www.informationweek.com/news/showArticle.jhtml?articleID=199600443

63 Warto tutaj odnotować, że największe na świecie portfolio patentów dotyczących komputerów i oprogramowania posiada IBM. Gdyby Microsoft zdecydował się na otwartą wojnę patentową, to mógłby paść ofiarą własnej broni. Sam IBM mógłby narazić Microsoft na kolosalne odszkodowania za naruszanie patentów, a jeśli do potencjalnych przeciwników doda się inne firmy wspierające rozwój Linuksa i również posiadające pokaźne liczby patentów (Red Hat, Novell, Sun, Sony, HP, Oracle), to rachunek jest prosty: Microsoftowi nie opłaca się wywoływać wojny patentowej. Niektórzy komentatorzy porównują tą sytuację do zimnej wojny.

64 Na podstawie *The History of Shareware*, http://www.asp–shareware.org/users/history–of–shareware.asp

prosił o dotację, jeśli program się spodobał. Jeden z użytkowników jego programu korzystał także z PC–TALK i zauważywszy podobieństwo między ich metodami dystrybucji wspomniał o tym Buttonowi. Zaciekawiony tą informacją, Button postanowił skontaktować się z Flueglemanem, po czym – aby sobie nawzajem pomóc – uzgodnili wspólne warunki dystrybucji swoich programów. Obaj polecali program drugiego w dokumentacji swojego programu, ustalili wspólną cenę, a Button dodatkowo zmienił nazwę swojego programu na PC–FILE, aby bardziej pasowała do nazwy programu Flueglemana. Jedyną różnicą było to, że Button na określenie tego sposobu dystrybucji używał zwrotu *user supported software* (oprogramowanie wspierane przez użytkowników).

W 1983 roku jeszcze jeden amerykański programista dołączył do tej historii. Bob Wallace, jeden z pierwszych pracowników Microsoftu, postanowił go opuścić i założyć swoją firmę QuickSoft. Wallace napisał edytor tekstu PC–WRITE i rozprowadzał go w podobny sposób jak Button i Fluegleman, ale z drobną różnicą. Wallace przy sprzedaży swojego programu oferował kupującym prowizję od tych użytkowników, którym dany kupujący przekazał ten program, i którzy później również go kupią. Polegało to na tym, że kupujący dostawał od Wallace'a numer seryjny, który można było umieścić na ekranie pokazywanym przy uruchomieniu programu. Rozdając program, dostarczało się go swoim znajomym razem z tym numerem seryjnym, a gdy oni decydowali się kupić PC–WRITE, to byli pytani o ów numer seryjny osoby polecającej. Wallace nazwał ten typ sprzedaży *shareware* (od ang. s*hare* – dzielić się).

W 1984 roku kolejny programista dołączył do układanki. Był to Nelson Ford, który w magazynie Softalk–PC ogłosił konkurs na nową nazwę dla tego modelu, ponieważ nazwa *freeware* była zarejestrowana. Zwyciężyła nazwa *shareware*. Nelson spytał Buttona o pozwolenie na używanie tej nazwy przez innych, a ten wyraził zgodę. Wraz ze wzrostem popularności oprogramowania shareware Button i Wallace zostali milionerami. Shareware stało się źródłem sukcesu wielu małych firm, zarówno produkujących

programy użytkowe, jak i gry komputerowe. Firmy takie jak Id Software czy Epic Megagames są dziś znaczącymi graczami na rynku.

W latach 1982–1983 dystrybucja programów shareware i freeware była darmowa. Ludzie wymieniali się dyskietkami w lokalnych grupach użytkowników lub ściągali programy z BBS–ów[65]. **Wraz ze wzrostem ilości udostępnianego oprogramowania rosły koszty utrzymania serwisu, więc administratorzy zaczęli wprowadzać opłaty za dostęp.** Pojawili się też ochotnicy oferujący wysyłanie programów na dyskietkach po koszcie nośnika i wysyłki. Z czasem do dystrybucji shareware dołączyli wydawcy magazynów komputerowych, którzy do swoich tytułów zaczęli dodawać dyskietki (a później płyty CD), oraz firmy, które – podobnie jak w/w ochotnicy – zajmowały się sprzedażą wysyłkową.

W 1987 roku zostało założone Association of Shareware Professionals[66], stowarzyszenie promujące model shareware. ASP obecnie ma ponad 1000 członków, wśród których znajdują się najwięksi producenci i dystrybutorzy oprogramowania shareware, znani z programów takich jak WinZip, czy Paint Shop Pro.

W latach 90–tych Internet i strony WWW wyparły serwisy BBS, a wraz z rozpowszechnieniem się szerokopasmowych łącz ograniczyły także rolę płyt CD. Powstały setki serwisów oferujących łatwiejsze wyszukiwanie i pobieranie oprogramowania shareware. **Popularność i łatwość użycia Internetu spowodowały także wprowadzenie przez część producentów ograniczeń redystrybucji swoich programów.** Producenci ograniczając możliwość pobierania programów tylko do swojej witryny i ewentualnych autoryzowanych dystrybutorów chcieli otrzymać dokładniejsze statystyki liczby użytkowników, a także zabezpieczyć ich przed nieautoryzowanymi kopiami, które mogły zawierać wirusy komputerowe.

65 Bulletin Board System – popularny w latach 80–tych system komputerowy działający na liniach telefonicznych, oferujący użytkownikom wymianę wiadomości i danych. Więcej informacji: http://pl.wikipedia.org/wiki/Bulletin_Board_System

66 http://www.asp–shareware.org

Popularność internetu umożliwiła także wprowadzenie nowej metody dystrybucji: *adware*[67] (ang. a*d,* skrót od *advertisment* – reklama). Programy tego typu podczas swojego działania wyświetlają reklamy pobierane z internetu, dzięki czemu użytkownik nie musi za program płacić. Z czasem jednak część producentów do programów adware zaczęło dołączać działające w ukryciu oprogramowanie zwane *spyware* (od ang. *spy* – szpieg), które zbierało różne informacje o użytkowniku i wysyłało je do producenta. Spowodowało to dużą niechęć zaawansowanych technicznie użytkowników oraz dało impuls do powstania programów usuwających takie niechciane, ukryte programy (np. Ad–Aware).

Obecnie oprogramowanie shareware cieszy się znacznie mniejszą popularnością niż w latach 90–tych. Dostępność w Internecie tzw. cracków[68] skłoniła część producentów do porzucenia tego modelu dystrybucji. **Z biegiem czasu niektóre programy shareware stały się typowymi programami komercyjnymi lub przeszły na model adware. Inne zostały udostępnione jako freeware lub porzucone. Zaś programy w wersjach testowych, na które zostały nałożone ograniczenia redystrybucji, już nie mogły być uznane za shareware.** Za spadek popularności programów shareware odpowiada także wzrost popularności wolnego oprogramowania, które nie nakłada na użytkownika tylu restrykcji (patrz rozdział 2.1.2 i 2.2).

1.3.4. Unix i BSD – akademickie podejście[69]

Unix jest jednym z pierwszych systemów operacyjnych. Napisany został przez Kena Thompsona i Dennisa Ritchie'go, pracowników należącego do AT&T centrum badawczego Bell Labs. Pierwsze prace nad Uniksem rozpoczęły się we wrześniu 1969 roku, a pierwsza edycja systemu została

67 Na podstawie: J. Stern, „*Spyware, Malware, Adware, Thief*", 2004, http://www.filetiger.com/articles/spyware.html

68 Crack – program łamiący zabezpieczenia przed kopiowaniem lub ograniczenia funkcjonalności, zob. rozdz. 2.1.2.

69 Na podstawie: M. K. McKusick, „*Twenty Years of Berkeley Unix: From AT&T–Owned to Freely Redistributable*", w: „*Open Sources: Voices from the Open Source Revolution*", wyd. O'Reilly Media, Sebastopol, California 1999, rozdz. 3, s. 31–46, dostępne w Internecie: http://www.oreilly.com/catalog/opensources/book/kirkmck.html

ukończona w 1971 roku[70]. System po raz pierwszy został zaprezentowany szerszej publiczności na sympozjum dotyczącym zasad funkcjonowania systemów operacyjnych, które odbyło się w listopadzie 1973 roku na Uniwersytecie Purdue. Na tym sympozjum obecny był także profesor Bob Fabry z Uniwersytetu Kalifornijskiego w Berkeley, który po prezentacji Thompsona i Ritchie'go zainteresował się ich systemem, i postanowił poeksperymentować z nim na swojej uczelni. W tym celu do Berkeley zakupiony został nowy komputer typu mainframe, PDP–11/45. W styczniu 1974 roku został zainstalowany na nim Unix, wersja 4.

Akademicka licencja Uniksa pozwalała na jego używanie i rozszerzanie przez studentów i pracowników akademickich, więc coraz więcej wykładowców prowadziło na nim zajęcia. System szybko zyskał popularność, więc uniwersytet zakupił kolejne komputery w celu uruchomienia na nich Uniksa. W 1975 roku Ken Thompson wziął roczny urlop naukowy, żeby poświęcić się pracy naukowej na uniwersytecie w Berkeley i uczyć studentów jak rozwijać Uniksa. W tym samym roku dwóch studentów Berkeley (Bill Joy i Chuck Haley) postanowiło napisać dla Uniksa kompilator języka Pascal. W międzyczasie, w związku z ograniczeniami edytora *ed*, Haley i Joy napisali swój własny edytor, *ex*. W 1976 roku Joy i Haley zaczęli także poprawiać różne części systemu, i wysyłać swoje poprawki do Bell Labs. Z czasem także inne uniwersytety zainteresowały się ich poprawkami i kompilatorem Pascala, więc Joy na początku 1977 roku postanowił skompletować pierwszą Dystrybucję Oprogramowania (z) Berkeley (ang. *Berkeley Software Distribution*), później zwaną 1BSD.

W 1978 roku kompilator Pascala dostał wiele poprawek od swoich użytkowników, więc postanowiono wydać drugą dystrybucję oprogramowania, której nazwę skrócono do 2BSD. W 1978 roku uniwersytet zakupił nowy komputer VAX, ale Unix wtedy nie posiadał obsługi pamięci wirtualnej więc pracownicy naukowi uczelni postanowili ją dla niego napisać. Nowe jądro systemu, wraz z narzędziami z Uniksa i uaktualnionymi

70 Zob. http://www.unix.org/what_is_unix/history_timeline.html

programami z 2BSD zostało w 1979 roku wydane jako 3BSD, pierwszy kompletny system operacyjny z Berkeley.

W międzyczasie Uniksem zainteresowała się DARPA – Agencja Zaawansowanych Projektów Badawczych Departamentu Obrony Stanów Zjednoczonych. Agencja potrzebowała uniwersalnego systemu operacyjnego dla sieci swoich centrów badawczych rozproszonych po terenie całego kraju. DARPA zdecydowała się ujednolicić system operacyjny w celu zlikwidowania wysokich kosztów związanych z utrzymywaniem różnych systemów we wszystkich swoich ośrodkach. Agencja podpisała z uniwersytetem w Berkeley kontrakt na rozszerzenie funkcjonalności Uniksa i sfinansowała prace powołanej w tym celu Computer Systems Research Group (grupy badawczej zajmującej się systemami komputerowymi). Efektem tych prac był m.in. protokół TCP/IP, będący podstawą sieci Internet. W 1980 roku CSRG wydała 4BSD, kolejną wersję systemu oferującą liczne ulepszenia (m.in. kontrolę procesów, oraz protokół pocztowy Delivermail, będący protoplastą późniejszego Sendmaila). Kolejna, wydana w 1981 roku wersja, nosiła nazwę 4.1BSD w celu uniknięcia kolizji nazw z systemem UNIX System V, sprzedawanym przez AT&T.

W 1982 roku Bill Joy opuścił CSRG, by wraz z 3 absolwentami Uniwersytetu Stanforda założyć firmę Sun Microsystems. Sun dostarczał ze swoimi stacjami roboczymi system SunOS, oparty na 4.1BSD. W tym samym roku AT&T wydało pierwszą komercyjną wersję swojego systemu pod nazwą UNIX System III. Wraz z wydaniem tej wersji zostały podwyższone opłaty licencyjne dla instytucji akademickich. Ponieważ systemy BSD zawierały kod Uniksa, to do ich używania wymagane było posiadanie licencji na Uniksa od AT&T. Z biegiem czasu do Berkeley napływało coraz więcej próśb o wydanie wersji systemu wolnej od kodu AT&T. W efekcie tych starań w 1989 roku wydany został pakiet Networking Release 1. Pakiet ten był niekompletnym systemem operacyjnym, ponieważ spora część kodu należała do AT&T i nie zdołano go jeszcze zamienić na kod bez ograniczeń licencyjnych. Networking Release 1 był pierwszą dystrybucją wydaną w całości na

liberalnej licencji BSD[71]. Co prawda za taśmę z dystrybucją trzeba było zapłacić 1000$, ale po jej kupieniu można było ją kopiować bez ograniczeń.

W międzyczasie AT&T udzielało licencji różnym firmom, które tworzyły swoje, również komercyjne wersje Uniksa (m.in. IBM AIX, HP–UX, SGI IRIX, Microsoft Xenix). Z czasem liczba będących w obiegu różnych, nie do końca kompatybilnych ze sobą, wersji Uniksa stała się tak duża, że w 1987 roku AT&T razem z Sunem postanowiło zunifikować kod kilku najpopularniejszych systemów uniksowych: UNIX System V Release 3, 4.3BSD, SunOS, i Xenix. Prace zostały rozpoczęte w 1988 roku, i ich efektem był wydany w 1990 roku UNIX System V Release 4.0 (w skrócie: SVR4). Ta wersja odniosła największy sukces, i na niej Sun oparł swój system Solaris.

W 1991 roku CSRG wydało pakiet Networking Release 2, który był prawie kompletnym systemem operacyjnym. Niedługo po nim zostały wydane dwa kompletne, oparte na nim systemy: darmowy 386BSD i komercyjny BSD/OS, rozwijany przez firmę Berkeley Software Design Incorporated (BSDi) założoną przez byłych pracowników CSRG. Niedługo po wydaniu 386BSD grupa użytkowników niezadowolonych z zastoju jaki zapanował przy rozwijaniu systemu postanowiła rozwijać system samodzielnie pod nazwą NetBSD[72]. Nieco później kolejna grupa użytkowników i programistów mająca własny pomysł na rozwijanie systemu utworzyła projekt FreeBSD[73].

W 1992 roku AT&T wytoczyło BSDi i Uniwersytetowi Kalifornijskiemu w Berkeley proces o naruszenie praw autorskich i znaku towarowego Unix. **Proces spowolnił rozwój systemów BSD na prawie dwa lata, dzięki czemu wielu użytkowników zainteresowało się Linuksem, który był wolny od zawirowań prawnych.** W 1993 roku Novell kupił od AT&T wszelkie prawa do Uniksa, a na początku 1994 zawarł z pozwanymi ugodę. Zgodnie z jej postanowieniami z systemu usunięto 3 pliki, a do 70 dołączono informację

71 Więcej na jej temat w rozdz. 2.2.2.
72 http://netbsd.org
73 http://freebsd.org

o prawach autorskich AT&T. **W czerwcu tego samego roku wydana została wolna od roszczeń dystrybucja 4.4BSD–Lite.** Projekty NetBSD i FreeBSD w celu uniknięcia podobnych pozwów sądowych oparły swoje systemy na tej wersji. Ostatnią wersją systemu wydaną przez Computer Systems Research Group była wydana w 1995 roku 4.4BSD–Lite Release 2. Po jej wydaniu CSRG uległa rozwiązaniu.

W grudniu 1994 roku Theo de Raadt, jeden z założycieli projektu NetBSD, pokłócił się z pozostałymi członkami projektu[74] i w kilka miesięcy później stworzył nowe odgałęzienie dystrybucji – OpenBSD[75]. Projekt stał się znany ze swojego olbrzymiego przywiązania do jakości kodu źródłowego oraz bezkompromisowej postawy w stosunku do licencjonowania oprogramowania. Twórcy OpenBSD odrzucają licencję GPL jako zbyt restrykcyjną, a także kategorycznie sprzeciwiają się umieszczaniu w systemie jakichkolwiek elementów binarnych (bez kodu źródłowego). Licencja GPL w OpenBSD jest akceptowana tylko dla komponentów, których napisanie od nowa trwało by zbyt długo (np. kompilator GCC). Twórcy OpenBSD chwalą się tym, że przez okres 10 lat w domyślnej instalacji systemu znaleziono tylko 2 luki, które można było zdalnie (na odległość) wykorzystać. OpenBSD obecnie cieszy się dużą popularnością wśród administratorów serwerów stawiających na pierwszym miejscu bezpieczeństwo.

Projekty NetBSD i FreeBSD również są aktywne do dzisiejszego dnia. NetBSD wspiera ponad 54 platformy sprzętowe i utrzymuje opinię najbardziej uniwersalnego systemu uniksowego. Chociaż jądro Linux obsługuje więcej architektur procesorów, to żadna dystrybucja Linuksa nie obsługuje tylu platform jednocześnie co NetBSD. FreeBSD zaś pozostaje najpopularniejszym systemem z rodziny BSD. FreeBSD ma najbardziej liberalne podejście do licencji kodu oraz zamkniętych komponentów, dzięki czemu cieszy się większą, niż pozostałe odmiany BSD, obsługą sprzętu.

74 http://www.forbes.com/forbes/2005/0704/071.html
75 http://openbsd.org

Chociaż wszystkie systemy BSD są mniej popularne od Linuksa, to wiele programów będących ich częścią jest dziś składnikami większości dystrybucji Linuksa (np. OpenSSH, Bind). Warto także odnotować, że spora część komponentów systemu FreeBSD stała się podstawą systemu Mac OS X firmy Apple[76]. Microsoft również wykorzystał trochę kodu BSD w swoich systemach[77] – narzędzia ftp, nslookup, finger, rsh i rcp dołączone do systemów Windows 2000, XP i Vista pochodzą z systemów BSD.

Zaś jeśli chodzi o dalsze losy Uniksa, to wartym uwagi jest Solaris, którego kod źródłowy został otwarty w ramach projektu OpenSolaris[78]. OpenSolaris jest dostępny na licencji CDDL[79].

76 http://developer.apple.com/opensource/
77 http://www.everything2.com/index.pl?node=BSD%20Code%20in%20Windows
78 http://www.sun.com/smi/Press/sunflash/2005-01/sunflash.20050125.1.xml
79 Zob. rozdz. 2.2.1: Inne licencje z klauzulą copyleft.

2. Rodzaje licencji oprogramowania

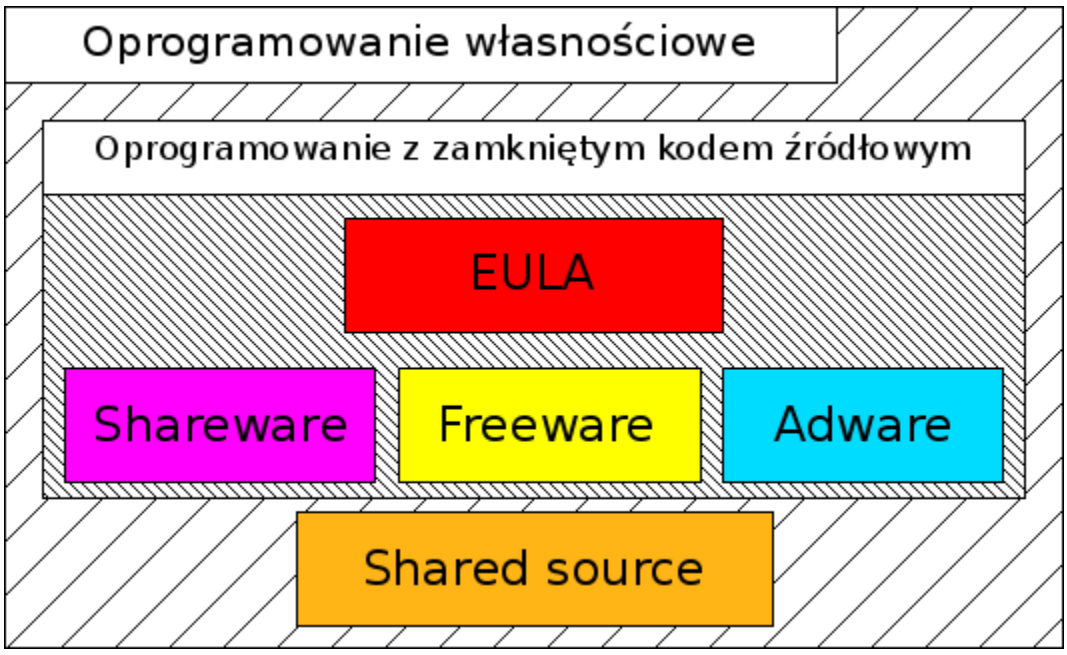
Podstawowym podziałem w kwestii licencjonowania oprogramowania jest podział na model zamknięty (oprogramowanie własnościowe, prawnie zastrzeżone, zamknięte) i model otwarty (wolne oprogramowanie, oprogramowanie o otwartym kodzie źródłowym). Istotą tego podziału jest podejście do praw dawanych użytkownikom oprogramowania. W przypadku oprogramowania własnościowego główną jego cechą jest zastrzeżenie wszelkich praw przez autora oprogramowania i – w konsekwencji – odebranie ich użytkownikowi. Zaś w przypadku wolnego oprogramowania podstawową jego cechą jest udzielenie i zagwarantowanie zachowania wolności użytkownikowi, co w konsekwencji oznacza dobrowolną rezygnację z większości praw przez autora. Oprócz modelu otwartego i zamkniętego istnieją także modele mieszane, które łączą ich cechy.

2.1. Oprogramowanie własnościowe

Nie ma żadnej definicji oprogramowania własnościowego. Pojęcie to jest po prostu tradycyjnym podejściem do produkcji i dystrybucji oprogramowania, w którym producent generuje swoje zyski ze sprzedaży licencji na używanie pojedynczych lub wielu kopii oprogramowania. Oprogramowanie własnościowe można podzielić na dwie grupy: typowe oprogramowanie z licencją EULA oraz oprogramowanie shareware/freeware/adware. Shareware, freeware i adware są ze sobą powiązane, dlatego można je traktować jak jedną grupę. Specyficznym przypadkiem jest Shared Source, które zostanie omówione w osobnym rozdziale[80]. Poszczególne kategorie oprogramowania własnościowego przedstawia Rys. 1.

80 Zob. rozdz. 2.4.

Rys. 1: Kategorie oprogramowania własnościowego.



Źródło: opracowanie własne.

2.1.1. EULA

Pojęcie „*EULA*" (ang. *End-User License Agreement* – licencja użytkownika końcowego) oznacza standardowy typ licencji, na której jest dystrybuowane oprogramowanie własnościowe. Termin ten dotyczy zarówno licencji dla użytkowników indywidualnych, jak i korporacyjnych. Licencje oprogramowania komercyjnego różnią się między sobą, ale można między nimi zauważyć pewne cechy wspólne. Każda licencja typu EULA zastrzega wszelkie prawa dla producenta oprogramowania. Użytkownik otrzymuje tylko ściśle określony, wąski zakres uprawnień[81].

Przykładowo licencja systemu Microsoft Windows XP Professional[82] zezwala użytkownikowi na:

– instalowanie i używanie tylko 1 kopii systemu,

81 A. M. St. Laurent, „*Understanding Open Source and Free Software Licensing*", wyd. O'Reilly Media, Sebastopol, California 2004, s. 114–120. Książka dostępna w Internecie: http://oreilly.com/catalog/osfreesoft/book/

82 http://download.microsoft.com/documents/useterms/Windows%20XP_Professional_Polish_a675687b-f2e8-4c76-b108-9657210d2135.pdf

- używanie systemu na komputerze posiadającym maksymalnie 2 procesory,
- udostępnianie usług obsługi plików, drukowania, połączenia z Internetem, oraz dostępu zdalnego maksymalnie 10 komputerom lub innym urządzeniom,
- wykonanie tylko 1 kopii zapasowej nośnika z oprogramowaniem,
- jednorazowe przekazanie (sprzedaż lub podarunek) produktu osobie trzeciej.

Licencja zabrania odtwarzania, dekompilacji i dezasemblacji produktu, oraz zastrzega wszystkie prawa nie udzielone licencjobiorcy. Oznacza to, że licencja zabrania m.in.:

- kopiowania oprogramowania (z wyjątkiem wykonania 1 kopii zapasowej),
- instalowania oprogramowania na większej niż 1 liczby stanowisk,
- modyfikowania oprogramowania,
- dystrybucji zmodyfikowanych kopii.

Licencja udziela użytkownikowi ograniczonej gwarancji na okres 90 dni, ale wyłącza wszelką odpowiedzialność producenta za ewentualne szkody powstałe w wyniku używania systemu.

Licencje typu EULA stały się przedmiotem krytyki ze względu na zbyt szerokie, nieraz niezgodne z prawem ograniczenia w nich zawarte. Electronic Frontier Foundation[83] opublikowała dokument opisujący takie kontrowersyjne ograniczenia zawarte w licencjach[84]. Dokument ten wymienia następujące klauzule zawarte w licencjach:

- zakaz publicznego krytykowania produktu (McAfee VirusScan, Microsoft .NET),

83 Amerykańska organizacja pozarządowa mająca na celu walkę o wolności obywatelskie (takie jak prawo do anonimowości, prywatności i wolności słowa) w dobie powszechnego użycia urządzeń elektronicznych.
84 http://www.eff.org/wp/dangerous-terms-users-guide-eulas

- zgoda na monitorowanie użytkownika przez producenta oprogramowania (Windows XP, Windows Vista),
- zakaz tzw. inżynierii wstecznej[85] (Windows XP, Windows Vista),
- zakaz używania produktu razem z produktami innych producentów (GAIN),
- automatyczna zgoda na wszystkie przyszłe wersje licencji (Apple iTunes),
- całkowite ograniczenie odpowiedzialności producenta (Windows XP, Windows Vista).

Lokalne ustawodawstwo (w zależności od miejsca zamieszkania użytkownika oraz miejsca zakupu produktu) może modyfikować lub całkowicie negować niektóre terminy zawarte w licencjach EULA. Prawo lokalne może np. określać minimalny zakres gwarancji i odpowiedzialności producenta oraz dopuszczać inżynierię wsteczną w celu wyprodukowania kompatybilnego oprogramowania. Zarówno prawo Stanów Zjednoczonych jak i Unii Europejskiej zezwala na w/w użycie inżynierii wstecznej.

Nie istnieje żadna uniwersalna wersja licencji EULA – każdy producent oprogramowania ma swoją, specyficzną dla każdego produktu. Użytkownik przed użyciem programu powinien dokładnie zapoznać się z warunkami jego użycia. W przypadku tzw. oprogramowania sprzedawanego w pudełku konsument nie ma możliwości zapoznania się z licencją przed zakupem produktu, gdyż jest ona zamieszczona w środku opakowania (najczęściej zafoliowanego) w formie papierowej lub na nośniku danych. Na początku wielu producentów stosowało tzw. s*hrink–wrap license* (licencja w folii) – licencje, które zaczynają obowiązywać w momencie otwarcia opakowania. Istnieje sporo kontrowersji co do ich legalności, dlatego producenci zamiast nich obecnie stosują tzw. c*lick–wrap license* – licencje zatwierdzane poprzez kliknięcie w uruchomionym programie. Część producentów umożliwia zwrot oprogramowania za cenę zakupu, jeśli użytkownik nie zgadza się

85 Inżynieria wsteczna (ang. *reverse engineering*) – proces badania urządzenia lub programu komputerowego w celu ustalenia jak on dokładnie działa. Proces ten jest zazwyczaj prowadzony w celu wyprodukowania odpowiednika badanego produktu.

z warunkami licencji (np. w/w system Windows XP). Inni zaś nic nie wspominają w licencji o możliwości ewentualnego zwrotu. W takim wypadku konsumentowi pozostaje jedynie dochodzenie swoich praw w sądzie. Licencje typu EULA są zazwyczaj długie i skomplikowane, dlatego większość klientów ich nie czyta. Ale nie zwalnia to ich od obowiązku przestrzegania licencji.

2.1.2. Shareware, freeware, adware

Pojęcia shareware, freeware i adware są blisko ze sobą związane, często się przenikają i mają wspólną historię[86].

Shareware

Shareware nie jest licencją. Jest to metoda marketingowa powstała w latach 80–tych, polegająca na udostępnianiu programów użytkownikom, aby mogli je wypróbować przed zakupem. Zwyczajowo jednak określenie shareware używane jest jako jedna z kategorii oprogramowania własnościowego. Nie ma żadnych sformułowanych definicji określających co jest, a co nie jest oprogramowaniem shareware.

Podstawową cechą oprogramowania shareware jest zezwolenie użytkownikom na dalsze rozpowszechnianie (redystrybucję) programu. Programy ograniczające prawo do redystrybucji nie mogą być określane mianem shareware. Programy shareware są dostarczane tylko w wersji binarnej, bez ich kodu źródłowego. Ich użytkownicy są zachęcani do kopiowania ich, i rozdawania swoim znajomym, co pozwala na osiągnięcie dużej popularności w krótkim czasie. Dzięki temu producenci mogą obniżyć koszty dystrybucji oraz reklamy oprogramowania, co przekłada się na jego cenę.

Drugą podstawową cechą shareware jest umieszczanie w programach zachęt do zakupu oprogramowania. Autor programu oczekuje zapłaty od klienta, jeśli po wypróbowaniu zamierza on go dalej używać. Jedni autorzy

86 Zob. rozdz. 1.3.3.

umieszczają takie informacje o możliwości zapłaty w formie sugestii, inni w formie żądania.

Każdy program shareware ma swoją osobną licencję, nie ma żadnego ogólnego typu licencji shareware. Nie ma też żadnej oficjalnej szczegółowej definicji (jak w przypadku wolnego oprogramowania). Początkowo programy shareware były w pełni funkcjonalne. Z czasem producenci zaczęli modyfikować funkcjonalność i warunki licencyjne używania swoich programów, aby osiągnąć większy zysk.

Pierwszą odmianą był podział na darmową wersję podstawową (ang. *basic*) lub lekką (*light*) i płatną wersję pełną (*full, retail*), zwaną także zarejestrowaną (*registered*) albo profesjonalną (*pro*). W wersji podstawowej umieszczano tylko podstawową funkcjonalność, a w wersji pełnej dodatkowe lub zaawansowane funkcje. W przypadku gier pełne wersje oferowały kolejne rozdziały, dodatkowe postacie, przedmioty, itd. Pełne wersje programów są już zwykłymi programami komercyjnymi i ich udostępnianie jest zabronione. Po zakupie pełnej wersji producent dawał kupującemu kod odblokowujący pełną funkcjonalność programu, zwany także numerem seryjnym (*serial number*), kluczem produktu (*product key*), bądź kodem aktywacyjnym (*activation code*).

Drugą odmianą były wersje testowe (ang. *trial*), rzadziej nazywane wersjami demonstracyjnymi (*demo*[87]). Programy tego typu miały ograniczenia działania, najczęściej czasowe (np. 30 dni) lub określające maksymalną liczbę uruchomień. Po upłynięciu okresu testowego lub po wykorzystaniu liczby uruchomień, program – w zależności od pomysłów jego autora – mógł już więcej się nie uruchamiać, przechodzić w tryb ograniczonej funkcjonalności (np. brak możliwości drukowania lub zapisu plików) lub regularnie wyświetlać informację o zakończeniu okresu próbnego, która jednocześnie zachęca do zakupu pełnej wersji. Programy, które wyświetlają taką informacje są nazywane *nagware* (od ang. *nag* – zrzędzić, męczyć,

87 Określenia *demo* rzadko używa się w odniesieniu do programów shareware, gdyż jest ono popularne wśród producentów gier, którzy często publikują wersje demonstracyjne swoich produktów nie będące oprogramowaniem shareware.

poganiać), zaś programy o celowo ograniczonej podstawowej funkcjonalności są nazywane *crippleware* (od ang. *crippled* – uszkodzony, kaleki). W tym przykładzie wyraźnie widać nieprecyzyjność tych terminów, gdyż mogą się one łączyć i przenikać.

Programy w wersjach testowych często umieszczają w systemie operacyjnym użytkownika ukryte pliki lub klucze rejestru, mające zapobiec dalszemu używaniu programu po jego odinstalowaniu i ponownym zainstalowaniu. Z czasem do programów shareware i trial zaczęły się pojawiać tzw. *cracki* (czyt. *kraki,* od ang. *crack* – łamać) – programy tworzone przez anonimowych programistów, usuwające zabezpieczenia przed dalszym używaniem lub odblokowujące pełną funkcjonalność. Oprócz nich, w crackerskich serwisach publikowano także generatory numerów seryjnych (*keygen*) oraz same numery seryjne. Chcąc walczyć z tym zjawiskiem, część producentów przestała umieszczać w testowych i niepełnych wersjach pełną funkcjonalność programu. W związku z tym, po rejestracji wymagane było ściągnięcie pełnej wersji i jej instalacja. Takie wersje testowe z technicznego i logicznego punktu widzenia nie różniły się od gier w wersji demo.

Freeware

Kolejną odmianą shareware jest *freeware* (od ang. f*ree software* – darmowe oprogramowanie). Określenie to początkowo było używane zamiennie z shareware, z czasem jego użycie ograniczono do darmowych programów nieograniczających czasowo używania. Freeware obejmuje także darmowe programy, które zabraniają redystrybucji lub ją ograniczają. Chociaż do pojęcia freeware pasuje także wolne oprogramowanie (*free software*), to należy unikać włączania go do kategorii freeware. Typowe programy rozprowadzane jako freeware, podobnie jak shareware, są udostępniane bez kodu źródłowego, a ich modyfikowanie jest zabronione. Dodatkowo część programów freeware ogranicza prawa użytkowników w zakresie konkretnych zastosowań – najczęściej chodzi o zezwalanie tylko na użytek niekomercyjny, a do użytku komercyjnego trzeba kupić pełną wersję. W celu uniknięcia kolizji nazw, zwolennicy wolnego oprogramowania

zamiast *freeware* proponują określenia *gratis software* oraz *free as in beer* (darmowe, jak darmowe piwo). W polskim języku nie ma tego problemu.

Chociaż oprogramowanie freeware jest darmowe, to część jego twórców prosi o dobrowolne wpłaty pieniędzy. Taka odmiana jest określana jako *donationware* (ang. *donation* – datek, dotacja). Twórca może prosić o wpłaty na swoje konto, lub na konto jakiejś organizacji charytatywnej. Inni twórcy zamiast o pieniądze proszą o wysłanie do nich kartki pocztowej (*postcardware*, *cardware*). Jeszcze inni wymyślają zabawne sposoby zapłaty za swój program, np.:

- *beerware* – twórca prosi o to, żeby przesłać mu piwo lub wypić je za jego zdrowie,
- *catware* – twórca prosi użytkownika, żeby przez godzinę pobawił się z przynajmniej 1 kotem,
- *sisterware* – twórca prosi użytkownika, żeby przedstawił mu swoją siostrę.

Inne przykłady takich nietypowych licencji można znaleźć w Wikipedii pod hasłem *Otherware*[88].

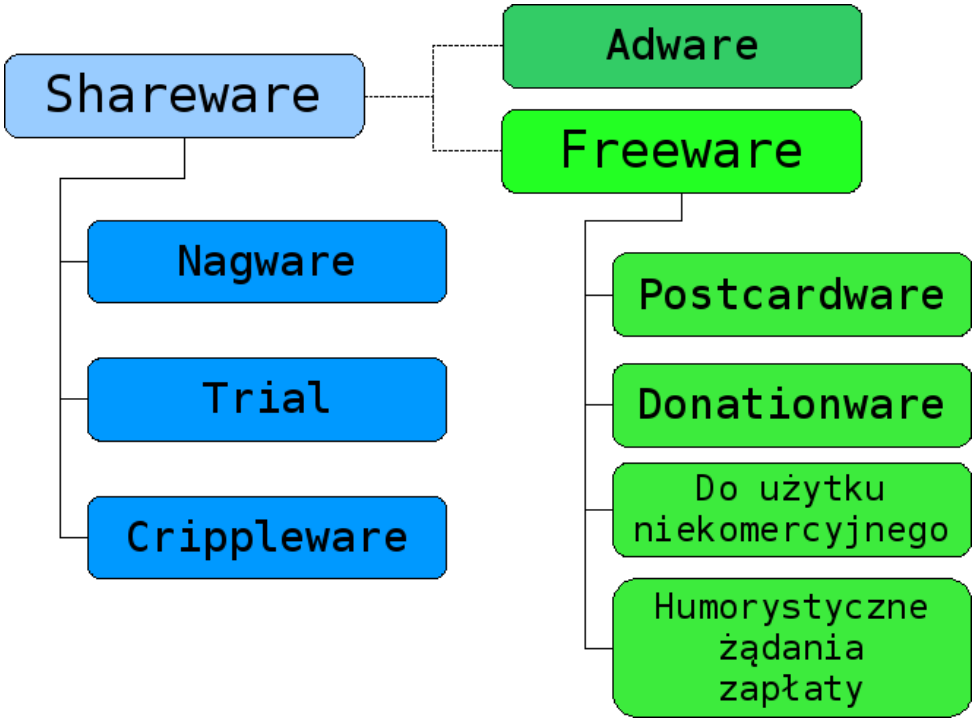
Adware

Ostatnią kategorią jest *adware* – darmowe oprogramowanie wyświetlające użytkownikowi reklamy. Reklamy wyświetlane w takim programie mogą być dla twórcy głównym źródłem dochodu. Niektórzy twórcy łączą adware z shareware, i za jednorazową opłatą oferują wersję bez reklam. Można też łączyć wiele w/w kategorii – np. program pocztowy Eudora przez długi czas był oferowany w 3 wersjach: płatnej, bez reklam; darmowej, z reklamami; oraz darmowej, bez reklam, ale z ograniczoną funkcjonalnością.

Jak już wcześniej było wspomniane, wszystkie te terminy mogą się łączyć w przeróżnych kombinacjach (zob. Rys. 2), dlatego warto przeczytać licencję każdego programu i sprawdzić na co ona pozwala. Tym bardziej, że niektórzy producenci mylnie te pojęcia interpretują i np. określają swój program mianem shareware, mimo iż jego redystrybucja jest zabroniona.

88 http://en.wikipedia.org/wiki/Otherware

Rys. 2: Kategorie oprogramowania shareware i freeware.



Źródło: opracowanie własne.

2.2. Wolne oprogramowanie

W przeciwieństwie do zamkniętego oprogramowania, wolne oprogramowanie (ang. *free software*) posiada jasną definicję czym jest. Określenie to po raz pierwszy zostało oficjalnie użyte najprawdopodobniej w wiadomości, którą Richard Stallman wysłał do grupy dyskusyjnej net.unix-wizards 28 września 1983 roku, ogłaszając w niej rozpoczęcie prac nad projektem GNU. Pierwszą próbą zdefiniowania wolnego oprogramowania był *Manifest GNU*[89] opublikowany przez Stallmana w miesięczniku *Dr. Dobb's Journal of Software Tools* w marcu 1985 roku[90]. Napisany w celu pozyskania współpracowników i poparcia dla Projektu GNU, przez wiele lat był podstawowym filozoficznym źródłem dla ruchu wolnego oprogramowania. Obecnie najbardziej precyzyjną definicję wolnego oprogramowania można

89 R. M. Stallman, L. Lessig, J. Gay, op. cit., s. 31.
90 http://www.math.utah.edu/ftp/pub/tex/bib/toc/dr-dobbs-1980.html#10(3):March:1985

przeczytać na stronach Projektu GNU[91]. Jest to oficjalna **definicja wg Free Software Foundation (FSF)**. Czytamy w niej:

> *„«Wolne oprogramowanie» to kwestia wolności, nie ceny. By zrozumieć tę koncepcję, powinniśmy myśleć o «wolności słowa», a nie «darmowym piwie» [ang. «free» znaczy najczęściej «wolny», «swobodny», ale może też oznaczać «darmowy» – przyp. tłum.].*
>
> *«Wolne oprogramowanie» odnosi się do prawa użytkowników do swobodnego uruchamiania, kopiowania, rozpowszechniania, analizowania, zmian i ulepszania programów. Dokładniej, mówimy o czterech rodzajach wolności użytkowników programu[92]:*
>
> *– wolność uruchamiania programu, w dowolnym celu (wolność 0),*
>
> *– wolność analizowania, jak program działa, i dostosowywania go do swoich potrzeb (wolność 1). Warunkiem koniecznym jest tu dostęp do kodu źródłowego.*
>
> *– wolność rozpowszechniania kopii, byście mogli pomóc sąsiadom (wolność 2)*
>
> *– wolność udoskonalania programu i publicznego rozpowszechniania własnych ulepszeń, dzięki czemu może z nich skorzystać cała społeczność (wolność 3). Warunkiem koniecznym jest tu dostęp do kodu źródłowego.*
>
> *Oprogramowanie nazywamy wolnym, jeśli wszyscy użytkownicy posiadają w pełni wszystkie te prawa. Zatem, powinniście mieć swobodę rozpowszechniania kopii programu, zmodyfikowanych bądź oryginalnych, za darmo bądź pobierając opłatę za dystrybucję, wszędzie i każdemu. Wolność robienia tego wszystkiego oznacza (między innymi), że nie musicie prosić o pozwolenie ani płacić za nie."*

Wolne oprogramowanie należy wyraźnie odróżnić od freeware[93]. Samo udostępnianie za darmo oraz zezwolenie na redystrybucję programu nie wystarczają do nazwania danego programu wolnym.

---

91 R. M. Stallman, „*Czym jest Wolne Oprogramowanie?*", tłum. Grupa tłumaczy witryny Projektu GNU, http://www.gnu.org/philosophy/free-sw.pl.html

92 Stallman w swojej definicji numeruje wolności od zera, tak jak programiści numerują np. zmienne przy pisaniu swoich programów.

93 Zob. rozdz. 2.1.2.

Otwarte oprogramowanie

Pokrewnym pojęciem do wolnego oprogramowania jest otwarte oprogramowanie (*open source*, oprogramowanie o otwartym kodzie źródłowym). Określenie to zostało sformułowane 3 lutego 1998 roku, podczas burzy mózgów w Palo Alto w Kalifornii. W owej sesji brali udział Todd Anderson, Christine Peterson (z Foresight Institute), John "maddog" Hall, Larry Augustin (obaj z Linux International), Sam Ockman (z Silicon Valley Linux User's Group), Michael Tiemann i Eric Raymond. Uczestnicy poszukiwali nowej nazwy dla wolnego oprogramowania: takiej, która by nie powodowała niejednoznaczności i nie kojarzyła się tylko z darmowym oprogramowaniem, i która by była bardziej przyjazna dla ludzi biznesu. Christine Peterson wymyśliła nazwę *open source*, która przez pozostałych uczestników spotkania została uznana za najlepszą.

Podobnie jak wolne oprogramowanie, także otwarte oprogramowanie ma swoją oficjalną definicję, dostępną na stronie Open Source Initiative[94]. **Definicja OSI** została oparta na Wytycznych Debiana dotyczących Wolnego Oprogramowania[95]. Pełna treść definicji brzmi[96]:

> *„Open source nie oznacza tylko dostępu do kodu źródłowego. Warunki dystrybucji oprogramowania open source muszą być zgodne z następującymi kryteriami:*
>
> ***1. Swoboda redystrybucji***
>
> *Licencja nie może ograniczać swobody którejkolwiek ze stron do sprzedawania lub rozdawania oprogramowania jako elementu szerszej dystrybucji zawierającej programy z różnych źródeł. Licencja nie może wymagać pobierania honorariów lub innych opłat od takiej sprzedaży.*
>
> ***2. Kod źródłowy***
>
> *Do programu musi być dołączony kod źródłowy, a licencja musi zezwalać na dystrybucję zarówno w postaci kodu*

94 Oficjalna definicja jest dostępna na stronie OSI: http://opensource.org/docs/osd
95 http://www.debian.org/social_contract#guidelines
96 Polskie (nieoficjalne) tłumaczenie pochodzi z archiwalnej wersji strony OSI: http://www.free-soft.org/mirrors/www.opensource.org/docs/osd-polish.php Punkt 10 definicji: tłumaczenie własne.

źródłowego, jak i skompilowanej. Jeśli któryś produkt nie jest rozprowadzany wraz z kodem źródłowym, musi istnieć dobrze udokumentowany sposób uzyskania tego kodu źródłowego za cenę nie przekraczającą rozsądnych kosztów wykonania kopii – najlepiej poprzez darmowe pobranie z Internetu. Kod źródłowy musi być dostępny w zalecanej postaci, pozwalającej na prostą modyfikację. Nie jest dozwolone celowe gmatwanie kodu źródłowego. Formaty pośrednie, takie jak wynik działania preprocesora lub translatora, nie są dozwolone.

3. Dzieła pochodne

Licencja musi zezwalać na dokonywanie zmian oraz tworzenie dzieł pochodnych. Musi również umożliwiać dystrybucję takich dzieł na tych samych warunkach, jakie opisuje licencja oryginalnego oprogramowania.

4. Spójność kodu źródłowego autora

Licencja może ograniczać dystrybucję kodu źródłowego w zmodyfikowanej postaci tylko wtedy, jeśli dozwolona jest przy tym dystrybucja «poprawek» (ang. patch) wraz z kodem źródłowym, za pomocą których program jest potem modyfikowany w trakcie kompilacji. Licencja musi jawnie zezwalać na dystrybucję oprogramowania skompilowanego ze zmodyfikowanego kodu źródłowego. Licencja może wymagać, aby dzieła pochodne nosiły inną nazwę lub numer wersji niż oprogramowanie oryginalne.

5. Niedozwolona dyskryminacja osób i grup

Licencja nie może dyskryminować jakichkolwiek osób czy grup.

6. Niedozwolona dyskryminacja obszarów zastosowań

Licencja nie może zabraniać wykorzystywania programu w jakimś konkretnym obszarze zastosowań. Na przykład, nie może zabraniać wykorzystania programu w sposób komercyjny lub używania go do badań genetycznych.

7. Dystrybucja licencji

Określenie praw dołączone do programu musi obowiązywać wszystkich, którzy otrzymują oprogramowanie bez konieczności przestrzegania przez te osoby dodatkowych licencji.

8. Licencja nie może obejmować konkretnego produktu

Określenie praw dołączone do programu nie może zależeć od tego, że dany program stanowi część określonej dystrybucji oprogramowania. Jeśli program został pobrany z takiej dystrybucji i wykorzystywany lub rozprowadzany zgodnie z warunkami licencji, wszystkie osoby do których program trafia powinny posiadać te same prawa, które określone są dla oryginalnej dystrybucji oprogramowania.

9. Licencja nie może ograniczać stosowania innego oprogramowania

Licencja nie może nakładać ograniczeń na inne oprogramowanie rozprowadzane wraz z oprogramowaniem objętym licencją. Na przykład, nie może wymagać aby wszystkie inne programy rozprowadzane na tym samym nośniku były programami open source.

10. Licencja musi być neutralna technologicznie

Żaden punkt licencji nie może narzucać konkretnej technologii lub stylu interfejsu."

Wolne czy Otwarte?

Obie definicje w dużej mierze się pokrywają, choć definicja OSI jest znacznie bardziej sformalizowana. Zakres pokrywania się definicji FSF i OSI ilustruje Tab. 1.

Tab. 1: Porównanie definicji wg FSF i OSI.

FSF \ OSI	P. 1	P. 2	P. 3	P. 4	P. 5	P. 6	P. 7	P. 8	P. 9	P. 10
Wolność 0					X	X	X			X
Wolność 1		X								X
Wolność 2	X						X	X	X	
Wolność 3	X	X	X							X

Źródło: opracowanie własne.

Definicja FSF nic nie wspomina o możliwości dystrybucji modyfikacji tylko w postaci poprawek (p. 4 definicji OSI). Choć definicja OSI wyraźnie zakazuje nakładania wielu typów ograniczeń, to ani razu nie pada w niej słowo

„wolność". Jest to główny punkt sporny pomiędzy zwolennikami FSF i OSI. Obie definicje wyraźnie zezwalają na pobieranie opłat za dystrybucję oprogramowania i jednocześnie wymagają aby w ten sposób pozyskany program można było bezpłatnie redystrybuować. Obie definicje żądają dostarczenia kodu źródłowego programu oraz zezwolenia na jego modyfikację i dystrybucję zmodyfikowanych wersji. W praktyce niemal każdy program program open source jest jednocześnie wolnym oprogramowaniem i vice versa. FSF i OSI publikują na swoich stronach listy licencji zgodnych z ich definicjami[97]. OSI dodatkowo przyznaje znak „Open Source Initiative certified" („certyfikowane przez OSI", zob. Rys. 3) wszystkim programom, których licencja jest zatwierdzona przez OSI jako zgodna z definicją oprogramowania open source[98].

Rys. 3: Znak "OSI certified".

Źródło: http://opensource.org/trademarks/osi-certified/

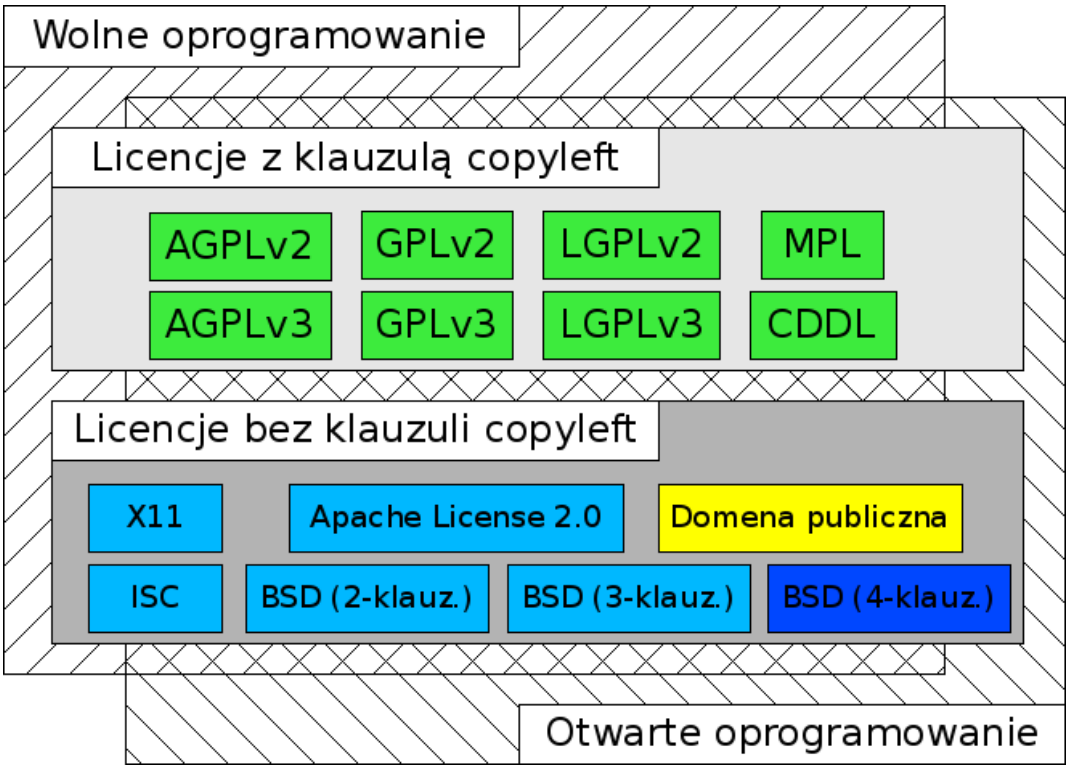
Wg Stallmana oba terminy opisują niemal tę samą kategorię oprogramowania, ale oznaczają poglądy oparte na fundamentalnie odmiennych wartościach[99]. Dla niego open source to metodologia tworzenia oprogramowania, a wolne oprogramowanie to ruch społeczny. Poza tym zwraca on uwagę, że chociaż twórcom terminu *open source* udało się uniknąć dwuznaczności terminu *free software* (wolny vs darmowy), to wpadli w inną,

97 http://www.fsf.org/licensing/licenses/
http://www.opensource.org/licenses/category
98 http://www.opensource.org/docs/certification_mark.html
99 http://www.gnu.org/philosophy/open-source-misses-the-point.html

gorszą pułapkę, bo w powszechnym mniemaniu wielu ludzi (przy zastosowaniu zwykłych reguł języka angielskiego) *open source* oznacza tylko możliwość obejrzenia kodu źródłowego programu, zaś ewentualna możliwość jego modyfikacji nie jest z tym pojęciem powiązana. Stallman nie uważa open source za wroga, tylko za jedno ze skrzydeł tego samego ruchu. Wrogiem dla niego jest zamknięte oprogramowanie. Zaś wg przedstawicieli ruchu open source, taka konfrontacyjna postawa Stallmana jest szkodliwa[100]. Otwarty kod źródłowy jest dla nich tylko sposobem na tworzenie lepszego oprogramowania. Rishab Aiyer Ghosh (członek zarządu OSI) zaproponował Free/Libre Open-Source Software (w skrócie FLOSS, ew. FOSS) jako neutralne pojęcie, bez przechylania się na którąkolwiek ze stron, ale popularność tego terminu jest niska.

Licencje wolnego oprogramowania można podzielić ze względu na obecność klauzuli *copyleft*. Ilustruje to Rys. 4.

Rys. 4: Podział licencji wolnego oprogramowania.



Źródło: opracowanie własne.

100 http://opensource.org/history

2.2.1. Licencje z klauzulą copyleft

Pojęcie *copyleft* jest ściśle związane z licencją GNU General Public License, choć samo było już wcześniej używane poza jej kontekstem. Angielskie słowo *copyleft* powstało przez sparodiowanie słowa *copyright* (prawo autorskie). Można je przetłumaczyć na język polski jako „*lewo autorskie*". Richard Stallman po raz pierwszy zetknął się z tym pojęciem w 1984 lub 1985 roku[101]. Pojęcie to tak bardzo mu się spodobało, że postanowił go użyć jako nieformalną nazwę swojej licencji. Wbrew pozorom, copyleft w ujęciu Stallmana nie oznacza negacji lub umyślnego łamania praw autorskich. Copyleft działa w ramach obowiązującego prawa, tylko wykorzystuje je odwrotnie od zamysłu ustawodawcy. Zamiast służyć ograniczaniu praw odbiorców dzieł i zatrzymywaniu ich przez twórców, copyleft oddaje prawie wszystkie prawa autorskie odbiorcom (prawo do używania, dystrybucji, tworzenia dzieł zależnych, czerpania korzyści majątkowych z dzieła) i zostawia twórcom tylko prawo do uznania autorstwa i ewentualnej zmiany licencji. Zasady działania pojęcia copyleft są objaśnione na stronie Projektu GNU[102]:

> *„By poddać program działaniu copyleft, najpierw zastrzegamy copyright, prawa autorskie do niego. Następnie dokładamy warunki rozpowszechniania, będące prawnym środkiem, dzięki któremu dajemy każdemu prawo do używania, modyfikowania i rozpowszechniania kodu naszego programu lub dowolnego programu pochodnego, ale tylko wtedy gdy warunki rozpowszechniania pozostaną niezmienione. W ten sposób, kod i przekazane wolności stają się prawnie nierozdzielne.*
>
> *Konstruktorzy oprogramowania prawnie zastrzeżonego wykorzystują system praw autorskich do odebrania wolności użytkownikom, my używamy praw autorskich, bym im tę wolność zagwarantować."*

Copyleft w jednakowym stopniu chroni prawa wszystkich użytkowników poprzez wymaganie przekazania tych samych praw podczas redystrybucji.

101 Stallman nie pamięta dokładnie który to był rok. Źródło: S. Williams, op. cit., s. 167–192 oraz R. M. Stallman, L. Lessig, J. Gay, op. cit., s. 21.

102 R. M. Stallman, „*Czym jest copyleft?",* tłum. Grupa tłumaczy witryny Projektu GNU, http://www.gnu.org/copyleft/copyleft.pl.html

Symbolem copyleft jest odwrócony znak copyright („©”), przedstawiony na Rys. 5.

Rys. 5: Symbol copyleft.

Źródło: http://en.wikipedia.org/wiki/Image:Copyleft.svg

GNU General Public License (GPL)

Copyleft samo w sobie nie jest licencją. Jest tylko modelem opisującym pewien typ licencji[103]. Podstawowym przykładem zastosowania idei copyleft jest licencja GNU General Public License (GPL), stworzona przez Richarda Stallmana. Stallman po raz pierwszy zastosował copyleft w 1985 roku, w licencji swojego edytora GNU Emacs. Kolejnymi licencjami, w których użył copyleft były licencje GNU Debuggera oraz GCC (GNU Compiler Collection – kolekcja kompilatorów GNU). Szybko jednak zauważył że ich licencje, specyficzne dla swoich programów, są ze sobą niekompatybilne, mimo zawierania niemal identycznej treści. Widząc to, Stallman postanowił stworzyć jedną, uniwersalną licencję, którą mógłby użyć dla każdego programu, umożliwiając w ten sposób łączenie kodu z różnych programów. Po wielu konsultacjach z prawnikami i członkami FSF, Stallman opublikował w 1989 roku pierwszą wersję licencji GPL[104].

Licencja GPLv1 zezwala użytkownikowi na:

– kopiowanie i dystrybucję programu, pod warunkiem zachowania informacji o licencji programu,

103 Licencje typu copyleft w angielskiej terminologii są też określane słowem *reciprocal*, co oznacza *obustronne, obopólne, odwzajemnione, zwrotne*. Żadne z tłumaczeń tego pojęcia nie przyjęło się w Polsce.

104 http://www.gnu.org/licenses/old-licenses/gpl-1.0.txt

- modyfikowanie programu w całości, lub w części, pod warunkiem umieszczenia informacji o dokonaniu modyfikacji, oraz licencjonowaniu ich na tej samej licencji (w przypadku umieszczenia części lub całości kodu programu objętego licencją GPL w innym programie licencja obejmuje całość jego kodu; samo umieszczenie programu objętego licencją GPL na tym samym nośniku co inne programy, nie powoduje objęcia ich tą licencją),
- pobieranie opłat za dystrybucję programu lub zapewnienie gwarancji jego działania,
- dystrybucję całości programu, jego części, lub programu na nim opartego w formie skompilowanej, pod warunkiem dołączenia do niego pełnego kodu źródłowego, lub ważnej przez 3 lata pisemnej oferty dostarczenia kodu źródłowego dowolnej osobie trzeciej, lub informacji o tym, gdzie ów kod można zdobyć,
- objęcie programu nowszą wersją licencji GPL, jeśli takowa zostanie opublikowana przez Free Software Foundation.

Licencja GPLv1 zabrania użytkownikowi:

- kopiowania, modyfikowania, sublicencjonowania lub dystrybuowania programu na warunkach innych niż ta licencja (lub jej nowsze wersje),
- nakładania na użytkownika dodatkowych ograniczeń.

Chociaż licencja zezwala na pobieranie opłat za dystrybucję programu, to osoby, które w ten sposób otrzymają program, mogą go dystrybuować innym osobom za darmo. GPL zawiera także klauzulę informującą o braku jakichkolwiek gwarancji co do jakości i przydatności programu oraz klauzulę wyłączającą (w zakresie dopuszczonym przez prawo) odpowiedzialność autorów i dystrybutorów programu od wszelkich szkód bądź skutków ubocznych wynikłych z użytkowania programu poddanego tej licencji. Autor programu może taką gwarancję osobno zapewnić i pobrać za to odpowiednią opłatę. Jeśli program posiada interaktywny interfejs (tekstowy lub graficzny), to GPL wymaga, aby pojawiła się w nim informacja o licencji programu.

GPLv2, LGPL i linkowanie

W 1991 roku Stallman opublikował drugą wersję licencji GPL[105], która, oprócz nieco zmienionego słownictwa, zawiera nową klauzulę 7[106]:

„Jeśli na skutek wyroku sądowego lub zarzutu naruszenia patentu, jak też z każdej innej przyczyny (nie ograniczonej do kwestii patentowych) zostaną narzucone na ciebie (niezależnie czy to mocą wyroku sądowego, umowy, czy w inny sposób) warunki sprzeczne z warunkami niniejszej Licencji, to nie zwalniają one ciebie z warunków Licencji. Jeśli nie możesz prowadzić dystrybucji tak, aby wypełniać jednocześnie swoje obowiązki z tytułu niniejszej Licencji i inne odnośne obowiązki, to w rezultacie nie możesz wcale rozprowadzać Programu. Na przykład, gdyby licencja patentowa nie zezwalała na wolną od opłat licencyjnych redystrybucję Programu przez wszystkie osoby, które otrzymały kopie bezpośrednio lub pośrednio od ciebie, to jedynym sposobem pozwalającym ci na przestrzeganie i licencji patentowej, i Licencji niniejszej, byłoby całkowite powstrzymanie się od jakiejkolwiek dystrybucji Programu."

Klauzula ta ma na celu uniemożliwienie ograniczania dystrybucji programów na licencji GPL za pomocą patentów i innych praw.

W GPLv2 została także zmodyfikowana klauzula 2, dotycząca modyfikacji programów i łączenia ich kodu z innymi. Wiąże się z nią problem linkowania (łączenia) programów ze sobą. Fragment tej klauzuli brzmi[107]:

„Musisz doprowadzić do tego, aby każda rozpowszechniana lub publikowana przez ciebie praca, która w całości lub części zawiera Program, albo pochodzi od niego lub jego części, była w całości i bezpłatnie licencjonowana dla wszelkich stron trzecich na warunkach niniejszej Licencji. (...)

105 http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
106 Cytowany tekst pochodzi z nieoficjalnego tłumaczenia licencji GPLv2: http://gnu.org.pl/text/licencja-gnu.html
Tłumaczenia licencji GPL nieopublikowane przez FSF w świetle prawa nie stanowią warunków rozpowszechniania oprogramowania i służą tylko lepszemu zrozumieniu treści licencji.
107 Ibidem (nieoficjalne tłumaczenie).

Niniejsze wymogi odnoszą się do zmodyfikowanej pracy jako całości. Jeśli dające się ustalić sekcje danej pracy nie pochodzą od Programu i mogą być racjonalnie uważane za samodzielne i odrębne same w sobie, to niniejsza Licencja i jej warunki nie mają zastosowania do takich sekcji przy rozprowadzaniu ich przez ciebie jako odrębne prace. Jeśli jednak rozprowadzasz je jako część całości, będącej pracą opartą na Programie, rozpowszechnianie tej całości musi być dokonywane na warunkach niniejszej Licencji, której zezwolenia dla innych licencjobiorców rozciągają się w całej szerokości na tę całość, a tym samym i na każdą indywidualną jej część, niezależnie od jej autorstwa.

Dlatego też intencją tego fragmentu nie jest roszczenie sobie praw albo podważanie twych praw do pracy napisanej w całości przez ciebie. Chodzi nam raczej o korzystanie z prawa kontrolowania dystrybucji pochodnych i zbiorowych prac opartych na Programie.

I jeszcze jedno: samo tylko połączenie z Programem (lub z pracą opartą na Programie) innej pracy – nie opartej na Programie, w ramach wolumenu nośnika przechowywania lub dystrybucji, nie powoduje objęcia takiej pracy zakresem niniejszej Licencji."

FSF na swojej stronie podaje[108], że każde linkowanie (zarówno statyczne jak i dynamiczne[109]) programu objętego licencją GPL z innym programem nią nie objętym powoduje, że ów program także musi zostać objęty licencją GPL. Wiele osób nie zgadza się z tą interpretacją. Lawrence Rosen[110] i Rod Dixon[111]

108 http://www.gnu.org/licenses/gpl–faq.html#IfLibraryIsGPL
http://www.gnu.org/licenses/gpl–faq.html#LinkingWithGPL

109 *Statyczne linkowanie* polega na połączeniu dwóch programów w 1 plik binarny, i dystrybuowaniu tego właśnie pliku. Linkowany program zazwyczaj zawiera często używane, podstawowe funkcje, z których korzystają także inne programy. Taki zbiór wielokrotnie wykorzystywanych funkcji nazywany jest *biblioteką* (ang. *library*). *Dynamiczne linkowanie* polega na łączeniu programu i biblioteki podczas jego uruchamiania. Program i biblioteka pozostają połączone tylko na czas działania programu. Program i biblioteka są dystrybuowane osobno, użytkownik może być zmuszony do samodzielnego zdobycia wymaganej do uruchomienia programu biblioteki.

110 L. Rosen, „*Open Source Licensing. Software Freedom and Intellectual Property Law*", wyd. Prentice Hall PTR, Upper Saddle River, New Jersey 2004, s. 103–140, książka dostępna w Internecie: http://rosenlaw.com/oslbook.htm

111 R. Dixon, „*Open Source Software Law*", wyd. Artech House, Norwood, Massachusetts 2004, s. 32–24.

twierdzą, że tylko statyczne linkowanie wymaga objęcia linkującego programu licencją GPL, zaś przy dynamicznym linkowaniu nie można uznać linkującego programu za dzieło pochodne od linkowanej biblioteki. Owa niejasność związana z linkowaniem stała się impulsem do wydania w 1991 roku specjalnej wersji licencji GPL przeznaczonej dla bibliotek. GNU Lesser General Public License[112] (ang. *lesser* – mniejsza), początkowo zwana GNU Library General Public License, adresuje ten problem pozwalając na linkowanie z programami własnościowymi (zamkniętymi). LGPL pozwala na relicencjonowanie kodu nią objętego na licencji GPL. FSF zaleca używanie LGPL tylko wtedy, kiedy jest to strategicznie opłacalne[113] – np. kiedy istnieją inne, zamknięte biblioteki implementujące podobną funkcjonalność. Zamiast licencji LGPL można także użyć licencji GPL ze specjalnym wyjątkiem dotyczącym linkowania (ang. *linking exception*). Taka licencja pozwala na linkowanie tylko z wybranymi programami (wymienionymi w informacjach o prawach autorskich do danego programu), których licencja jest niezgodna z GPL. Szczegółowe informacje na ten temat są podane na stronie Projektu GNU[114]. Warto także odnotować, że GPL domyślnie zawiera wyjątek dotyczący linkowania ze standardowymi komponentami systemu operacyjnego. Twórca programu korzystającego ze standardowych komponentów systemu (jądro, biblioteki, itp.) nie musi dostarczać ich źródeł przy dystrybucji swojego programu. Licencja GPL w wersji 2 obecnie pozostaje najpopularniejszą na świecie licencją stosowaną do wolnego oprogramowania[115].

GPLv3 i AGPLv3

W 2007 roku została wydana 3 wersja licencji GPL[116] oraz 3 wersja LGPL[117]. GPLv3 adresuje problemy i luki powstałe po opublikowaniu GPLv2.

112 http://www.gnu.org/licenses/old-licenses/lgpl-2.1.html
113 http://www.gnu.org/licenses/why-not-lgpl.html
114 http://www.gnu.org/licenses/gpl-faq.html#GPLIncompatibleLibs
115 Dane z serwisu Freshmeat: http://freshmeat.net/stats/#license
116 http://www.gnu.org/licenses/gpl.html
117 http://www.gnu.org/licenses/lgpl.html

Pierwszym z takich problemów jest tzw. tivoizacja (ang. *tivoization*). Termin ten został wymyślony przez Richarda Stallmana na określenie wszelkich sprzętowych prób ograniczania uruchamiania zmodyfikowanego oprogramowania objętego licencją GPL[118]. Nazwa pochodzi od TiVo, urządzenia do nagrywania programów telewizyjnych na dysk twardy w formacie cyfrowym. TiVo zawiera oprogramowanie na licencji GPLv2 (Linux, i narzędzia GNU), a jego producent publikuje na swojej stronie www kod źródłowy używanych w TiVo programów. Mimo to, użytkownik nie może uruchomić na TiVo zmodyfikowanych wersji programów, bo sprzęt blokuje nieautoryzowane wersje oprogramowania. Sekcja 6 licencji GPLv3 przy dystrybucji programów nią objętych w produktach konsumenckich wymaga podania wszelkich informacji niezbędnych do uruchamiania zmodyfikowanych programów (metody, procedury, klucze autoryzujące, itp.).

Drugim problemem obecnym w licencji GPLv2 jest stosunek do patentów na oprogramowanie. Choć GPLv2 blokuje możliwość udzielania licencji patentowych pojedynczym użytkowników w celu ograniczania dystrybucji oprogramowania nią objętego, to można to ominąć, tak jak to zrobił Microsoft z Novellem – podpisując umowę o niepozywaniu swoich klientów o naruszanie patentów[119]. GPLv3 zawiera szczegółową klauzulę o patentach (sekcja 11). Sekcja ta określa, że każdy autor i współautor udziela licencjobiorcy licencji na swoje patenty mające zastosowanie w danym programie. Klauzula ta zabrania licencjobiorcy dalszej dystrybucji oprogramowania, jeśli użyje on patentów do ograniczania praw, które są gwarantowane innym użytkownikom tego oprogramowania. GPLv3 obejmuje zarówno licencje na używanie patentów, jak i umowy o niepozywaniu o naruszenie patentów. W przypadku podpisania takiej umowy z 1 klientem, ochrona z niej wynikająca zostaje rozciągnięta na wszystkich użytkowników danego oprogramowania objętego licencją GPLv3.

Trzecim problemem z licencją GPLv2 jest niekompatybilność z niektórymi popularnymi licencjami. Problem ten został częściowo

118 http://www.gnu.org/licenses/gpl-faq.html#Tivoization
119 Zob. rozdz. 1.3.2.

zaadresowany poprzez sekcję 7, pozwalającą na nałożenie pewnych szczegółowo określonych dodatkowych warunków licencyjnych – np. ograniczenia użycia znaków handlowych, ograniczenia użycia nazwisk autorów i licencjodawców w celach reklamowych, wymagania wyraźnego odróżnienia wersji pochodnych od oryginału. Dzięki tej klauzuli osiągnięto kompatybilność m.in. z licencją Apache License 2.0 (tzn., że kod na tej licencji można włączyć do kodu na GPLv3). Szczegółową listę zmian pomiędzy licencjami GPLv2 i GPLv3 publikuje FSF[120].

Ostatnim dużym problemem z GPLv2 jest tzw. luka usług internetowych (ang. a*pplication service provider loophole*, w skrócie *ASP loophole*). Świadczenie usług przez sieć komputerową przy użyciu oprogramowania na licencji GPLv2 nie wymaga dystrybucji kodu źródłowego używanego programu, gdyż nie dochodzi wtedy do dystrybucji programu. W celu załatania tej luki, 19 listopada 2007 roku została wydana osobna licencja, GNU Affero General Public License[121], wersja 3 (AGPLv3). Licencja ta jest rozszerzeniem GPLv3 o warunki udostępniania oprogramowania jako usługi sieciowej. AGPLv3 wymaga udostępnienia kodu źródłowego programów, które komunikują się z użytkownikiem poprzez sieć komputerową. GPLv3 jest kompatybilna z AGPLv3 – tj. kod na GPLv3 można włączyć do kodu na AGPLv3, i wtedy całość będzie objęta tą drugą licencją. GNU AGPLv3 jest uaktualnieniem licencji Affero General Public License[122] (wersja 1), wydanej w 2002 roku przez firmę Affero Inc. AGPLv1 jest oparta na GPLv2, ale nie jest z nią kompatybilna. Kod na AGPLv1 można relicencjonować na AGPLv3.

Licencje GPL (w każdej wersji) są określane mianem *strong copyleft* (silne copyleft), a LGPL – *weak copyleft* (słabe copyleft). AGPL również jest zaliczana do *strong copyleft*, ale jej klauzula *copyleft* jest silniejsza niż w GPL.

120 http://www.fsf.org/licensing/licenses/quick-guide-gplv3.html
121 http://www.gnu.org/licenses/agpl.html
122 http://www.affero.org/oagpl.html

Mity związane z GPL

Powszechnie funkcjonuje wiele mitów związanych z używaniem licencji GPL. Najczęściej wynikają one z niewiedzy, ale część z nich jest efektem wrogiej propagandy sianej przez Microsoft i SCO[123]. Bill Gates i Steve Ballmer wielokrotnie porównywali licencję GPL i Linuksa[124] do wirusa, raka, komunizmu, a nawet Pac–Mana[125]. SCO zaś kwestionowało legalność licencji GPL[126].

Pierwszym, i chyba najpopularniejszym z tych mitów jest twierdzenie, że oprogramowania na GPL nie można sprzedawać i trzeba je rozdawać za darmo. Stwierdzenie to jest nieprawdziwe, gdyż każda wersja licencji GPL zezwala na pobieranie opłat za dystrybucję oprogramowania. Poza tym, firmy sprzedające produkty oparte na oprogramowaniu objętym licencją GPL (np. Red Hat, Novell, Mandriva) opierają swój biznes na wartości dodanej – dokumentacji, szkoleniach, wsparciu technicznym, lub aktualizacjach oprogramowania. Mit ten wynika z nadinterpretacji faktu, że każdy kupujący może zakupiony program redystrybuować za darmo.

Drugim popularnym mitem jest twierdzenie, że każdy program stworzony przy użyciu oprogramowania na GPL również podlega tej licencji. Stwierdzenie to również jest nieprawdziwe, gdyż GPL nie nakłada żadnych ograniczeń na zastosowania uruchamianego programu. Wynik działania programu może być objęty licencją GPL tylko wtedy, gdy zawiera kod danego programu.

Trzecim mitem jest stwierdzenie, że GPL, tak jak wirusy, zaraża każde oprogramowanie, z którym się zetknie. Również jest to nieprawdą, gdyż w licencji GPL wyraźnie jest zapisane, że dystrybucja oprogramowania GPL

123 Zob. rozdz. 1.3.2.
124 http://www.theregister.co.uk/2000/07/31/ms_ballmer_linux_is_communism/
http://www.theregister.co.uk/2001/06/02/ballmer_linux_is_a_cancer/
http://www.theregister.co.uk/2001/06/20/gpl_pacman_will_eat_your/
http://www.theregister.co.uk/2002/04/22/gates_gpl_will_eat_your/
125 Pac–Man – tytułowy bohater popularnej gry komputerowej, który w owej grze ma za zadanie zjeść wszystkie kulki na planszy.
126 http://www.fsf.org/licensing/sco/sco–preemption.html

na tym samym nośniku, co inne oprogramowanie, nie powoduje objęcia tego oprogramowania licencją GPL.

Ostatnim popularnym mitem jest twierdzenie, że każdy musi ujawnić wszystkie swoje modyfikacje programu objętego GPL. Jest to częściowo prawdą, bo GPL wymaga ujawnienia swoich modyfikacji tylko przy dystrybucji zmodyfikowanego programu. W przypadku używania zmodyfikowanego programu tylko we własnym zakresie, lub uruchamiania go na serwerze, GPL nie wymaga ujawnienia źródeł programu. Wyjątkiem jest AGPL, która wymaga ujawnienia źródeł programu uruchamianego na serwerze, ale mimo to AGPL nie wymaga ujawnienia źródeł programu uruchamianego lokalnie. Pozostałe wątpliwości co do licencji GPL są wyjaśnione na stronie Projektu GNU[127].

Inne licencje z klauzulą copyleft

Oprócz GPL istnieje także wiele innych licencji typu copyleft które nie zostały wydane przez FSF. Najbardziej znanymi przykładami są Mozilla Public License[128] (MPL) oraz, oparta na niej, Common Development and Distribution License[129] (CDDL). Obie licencje zostały uznane za licencje wolnego oprogramowania (przez FSF) oraz open source (przez OSI). Obie są określane jako słabe copyleft, bo pozwalają na statyczne linkowanie z programami na innych licencjach (przez co są niekompatybilne z GPL), oraz umieszczanie w kodzie programu plików na innych licencjach. Najbardziej znanym programem na licencji MPL jest przeglądarka Mozilla, zaś na CDDL – system operacyjny OpenSolaris. FSF stanowczo odradza tworzenie swoich licencji, gdyż z dużym prawdopodobieństwem mogą one być niekompatybilne z GPL.

127 http://www.gnu.org/licenses/gpl-faq.html
128 http://www.mozilla.org/MPL/MPL-1.1.html
129 http://www.sun.com/cddl/cddl.html

2.2.2. Licencje bez klauzuli copyleft

Licencje bez klauzuli copyleft nazywane są zezwalającymi (ang. *permissive*) lub licencjami akademickimi. Ich treść jest znacznie krótsza od licencji copyleft, dzięki czemu są prostsze do zrozumienia.

Licencja BSD

Pierwszymi programami udostępnionymi na licencji non–copyleft były elementy wydanej w latach 70–tych dystrybucji 1BSD, ale pierwszym oprogramowaniem w całości rozpowszechnianym na warunkach tej licencji była wydana w czerwcu 1989 roku dystrybucja Networking Release 1[130]. Jej licencja, to **4–klauzulowa licencja BSD**[131], zwana także starą licencją BSD. Licencja ta zezwala na użytkowanie i redystrybucję oprogramowania, z modyfikacjami lub bez, w formie kodu źródłowego lub binarnego, pod warunkiem zachowania informacji o prawach autorskich w kodzie źródłowym i dokumentacji programu, oraz pod warunkiem nie używania nazwy Uniwersytetu Kalifornijskiego w Berkeley i nazwisk jego współpracowników do reklamowania produktów pochodnych od danego programu, bez uzyskania pozwolenia. Konsekwencją tych liberalnych warunków licencji jest zezwolenie na włączanie oprogramowania nią objętego do produktów na innych licencjach, zarówno otwartych, jak i zamkniętych. Licencja ta, tak jak większość licencji wolnego oprogramowania, zawiera także informację o braku jakichkolwiek gwarancji oraz nie ponoszeniu przez autorów i dystrybutorów programu odpowiedzialności za szkody wynikłe z użytkowania programu.

Klauzula ogłoszeniowa, 3–klauzulowa licencja BSD

Specyficzną cechą 4–klauzulowej licencji BSD jest tzw. klauzula ogłoszeniowa (lub reklamowa), która wymaga, aby wszystkie materiały

130 Chociaż już wcześniejsze wersje BSD były licencjonowane na takich warunkach, to wymagały licencji od AT&T, gdyż zawierały także kod Uniksa. Zob. rozdz. 1.3.4.
131 Treść licencji jest dostępna na stronie projektu FreeBSD: http://www.freebsd.org/copyright/license.html

reklamujące produkty korzystające z kodu na tej licencji zawierały informację:

„*Ten produkt zawiera oprogramowanie stworzone przez Uniwersytet Kalifornijski w Berkeley oraz jego współpracowników*[132]”.

Klauzula ta powoduje niekompatybilność z wieloma licencjami (zwłaszcza GPL[133]), oraz sprawia problem praktyczny, kiedy kolejni autorzy wpisują w niej swoje nazwisko lub nazwę firmy w miejsce Uniwersytetu Kalifornijskiego. W dużych projektach, przez używanie takich klauzul, liczba informacji o autorach wymaganych w materiałach reklamowych wzrasta. Przykładem może być system NetBSD, w którego dokumentacji (jak podaje Richard Stallman[134]) w 1997 roku było 75 takich informacji reklamowych o poszczególnych autorach. W 1999 roku Uniwersytet Kalifornijski zrezygnował z używania tej klauzuli i zezwolił na usunięcie jej z wszelkiego kodu do którego ma on prawa autorskie[135]. Powstała w ten sposób licencja jest nazywana **3–klauzulową licencją BSD**[136], lub zmodyfikowaną (albo nową) licencją BSD. Licencja ta jest kompatybilna z GPL i innymi popularnymi licencjami. Od czasu tej decyzji wszystkie systemy BSD przeszły na 3–klauzulową licencję, z wyjątkiem NetBSD, które nadal używa 4–klauzulowej wersji[137]. W konsekwencji, w dokumentacji bieżącej wersji NetBSD można znaleźć 239 informacji reklamowych o poszczególnych autorach[138].

2–klauzulowa BSD, MIT (X11) oraz ISC

Część programistów używa także 2–klauzulowej licencji BSD[139], powstałej przez usunięcie klauzuli zakazującej używania nazwisk autorów do reklamowania produktów pochodnych. Licencja ta jest funkcjonalnie

132 Tłumaczenie własne.
133 http://www.gnu.org/philosophy/bsd.pl.html
134 Ibidem.
135 ftp://ftp.cs.berkeley.edu/pub/4bsd/README.Impt.License.Change
136 http://www.opensource.org/licenses/bsd–license.php
137 http://www.netbsd.org/about/redistribution.html#default
138 http://cvsweb.netbsd.org/bsdweb.cgi/~checkout~/src/distrib/notes/common/legal.common?rev=1.47
139 http://www.gnu.org/licenses/info/BSD_2Clause.html

identyczna z licencją MIT[140] (zwaną także licencją X11), oraz licencją ISC[141] (używaną m.in. przez OpenBSD[142]). Mnogość licencji z „BSD" w nazwie powoduje niejednoznaczność terminu „licencja BSD", dlatego Stallman proponuje zamiast którejkolwiek z licencji BSD używać licencji X11[143] (nazwa „licencja MIT" też może być niejednoznaczna, ponieważ MIT używało wielu licencji[144]). Z tego samego powodu można preferować licencję ISC, która dodatkowo jest napisana prostszym językiem, dzięki czemu sprawia mniej problemów w interpretacji prawnej. Licencja X11 zawiera nieprecyzyjne sformułowania „*zajmować się oprogramowaniem*[145]" (w oryg. „*to deal in the Software"*) oraz „*oprogramowanie i związane [dołączone] pliki dokumentacji*[146]" (w oryg. „*this software and associated documentation files"*), które (wg Rosena[147]) mogą sprawiać problemy w interpretacji prawnej.

Apache License

Osobnym ciekawym przypadkiem jest licencja Apache[148]. Jest to licencja stworzona przez Apache Software Foundation, początkowo tylko dla ich serwera protokołu HTTP – Apache. Pierwsze wersje tej licencji były podobne do 4–klauzulowej licencji BSD, z dodaniem klauzuli zabraniającej produktom pochodnym używania nazwy Apache. Wersja 2.0, opublikowana w 2004 roku, znacząco się od nich różni. Apache License 2.0, podobnie jak w/w licencje akademickie, zezwala na używanie, modyfikowanie i redystrybucję programu w postaci źródłowej, lub binarnej, bez obowiązku udostępnienia kodu źródłowego. Oznacza to, że kod na tej licencji można włączyć do zamkniętych programów, pod warunkiem zachowania zgodności z warunkami tej licencji. Podobnie jak poprzednie wersje, Apache License 2.0 nie udziela zezwolenia na używanie nazw i znaków handlowych licencjodawcy, z wyjątkiem użycia ich w informacji o pochodzeniu produktu.

140 http://www.opensource.org/licenses/mit–license.php
141 Licencja stworzona przez Internet Systems Consortium.
142 http://www.openbsd.org/cgi–bin/cvsweb/~checkout~/src/share/misc/license.template
143 http://www.gnu.org/philosophy/bsd.pl.html
144 http://www.fsf.org/licensing/licenses/#X11License
145 Tłumaczenie własne.
146 Ibidem.
147 L. Rosen, op. cit., s. 73–102.
148 http://www.apache.org/licenses/

Najważniejszą zmianą, w porównaniu do typowych licencji akademickich, jest klauzula o patentach. Apache License 2.0 określa, że licencjobiorca otrzymuje od wszystkich autorów i współautorów licencję na użycie ich patentów mających zastosowanie w licencjonowanym programie. Jednakże, jeśli licencjobiorca wytoczy komukolwiek proces o to, że ów program narusza jego patenty, to w/w licencja na użycie patentów autorów i współautorów zostanie mu cofnięta. Apache License 2.0 jest niekompatybilna z GPLv2, ale jest kompatybilna z GPLv3[149]. Ze względu na obejmowanie patentów i znaków handlowych, licencja ta stanowi ciekawą alternatywę dla licencji typu BSD.

Domena publiczna

Przy omawianiu licencji akademickich warto także wspomnieć o domenie publicznej. Domena publiczna nie jest licencją. Jest to ogół twórczości, do której wygasły autorskie prawa majątkowe, i która jest przez to dostępna dla wszystkich, i do dowolnych zastosowań. Do domeny publicznej należą także dzieła, które od początku ich powstania nie były objęte tymi prawami (np. teksty ustaw, statystyki sportowe, formuły matematyczne). W/w licencji akademickich, ani licencji copyleft nie należy mylić z domeną publiczną. Zarówno licencje copyleft, jak i akademickie, opierają się na prawie autorskim – w każdej z nich jest wyraźne zastrzeżenie praw autorskich. Dopiero po zastrzeżeniu tych praw autor udziela zezwolenia na używanie, kopiowanie i modyfikowanie, pod ściśle określonymi warunkami.

2.3. Podwójne licencjonowanie

Właściciel praw autorskich może licencjonować swoje dzieła wielokrotnie i na odmiennych warunkach. Firmy sprzedające oprogramowanie własnościowe często stosują kilka typów licencji jednocześnie – np. licencje dla użytkowników indywidualnych, dla użytkowników grupowych, oraz dla instytucji akademickich. Licencje te

149 http://www.fsf.org/licensing/licenses/#apache2

różnią się zakresem uprawnień oraz ceną. W przypadku wolnego oprogramowania jednoczesne wydawanie produktów na kilku licencjach jest stosowane ze względu na kompatybilność z innymi licencjami wolnego oprogramowania. Np. interpreter języka Perl jest wydany na Artistic License oraz na GPL[150], a przeglądarka Mozilla Firefox jest wydana jednocześnie na MPL, LGPL i GPL[151]. Użytkownik podczas modyfikacji programu ma możliwość wyboru jednej (lub wszystkich) z wymienionych licencji, ale po wybraniu nie może danej licencji zmienić.

Niektóre firmy stosują jednocześnie oba w/w modele licencjonowania dla swoich produktów – wydają je na licencji typu copyleft, oraz komercyjnej. Najbardziej znanymi przykładami są baza danych MySQL[152] oraz biblioteka interfejsu użytkownika Qt[153]. Twórcy oprogramowania korzystającego z w/w produktów mogą wybrać wersję na GPL, ale wtedy ich oprogramowanie też musi być objęte tą licencją (zgodnie z warunkami modyfikacji i dystrybucji zawartymi w licencji), lub mogą kupić licencję komercyjną, i wtedy nie muszą ujawniać swojego kodu źródłowego. Użytkownicy wybierający wersję GPL nie mogą udzielać innym komercyjnych licencji.

Jednoczesne komercyjne i otwarte licencjonowanie programów pozwala lepiej finansować rozwój programu oraz otrzymywać wsparcie niezależnych twórców–ochotników. Aby móc udzielać licencji komercyjnych na programy rozwijane przez społeczność niezależnych twórców, firma rozwijająca dany program musi otrzymać od każdego z nich zgodę na relicencjonowanie ich kodu. Udostępnienie programu na wolnej licencji pozwala na szybsze zdobycie dużej liczby użytkowników i uzyskanie efektu sieciowego[154], co można porównać z modelem shareware[155].

150 http://dev.perl.org/licenses/
151 http://www.mozilla.org/MPL/
152 http://www.mysql.com/products/which–edition.html
153 http://trolltech.com/company/model
154 Efekt sieciowy – pojęcie w teorii ekonomii opisujące sytuację, w której wartość danego dobra jest bezpośrednio związana z liczbą jego użytkowników. Przykładem takiego dobra jest telefon – jego wartość dla nowych użytkowników jest większa, jeśli mogą za jego pomocą komunikować się z większą liczbą osób.
155 Zob. rozdz. 2.1.2.

Podwójne licencjonowanie można łączyć z innymi cechami produktu. Produkt na licencji komercyjnej może odróżniać się od otwartego dodatkową funkcjonalnością, posiadaniem gwarancji producenta, lub wsparciem technicznym. Np. komercyjna wersja bazy MySQL jest oferowana wraz z szerokim wachlarzem usług[156], zaś komercyjna wersja maszyny wirtualnej VirtualBox oferuje funkcje niedostępne w wersji otwartej[157].

Podwójne licencjonowanie może być także rozłożone w czasie. Firma Artifex Software przez wiele lat wydawała Ghostscript (interpreter języka PostScript) na licencji komercyjnej oraz Alladin Free Public License[158] (licencja zezwalająca na modyfikację kodu źródłowego tylko do celów niekomercyjnych), zaś wersję na GPL wydawała z rocznym opóźnieniem[159]. Obecnie Artifex Software zrezygnowała z licencji AFPL i wydaje bieżącą wersję Ghostscript na GPL oraz komercyjnej licencji[160].

Niemniej jednak, podwójne licencjonowanie ma dość wąskie zastosowanie. Jak podaje Mikko Välimäki, model ten jest praktyczny tylko w określonych warunkach[161]. Po pierwsze, nie ma sensu go stosować do licencji bez klauzuli copyleft, gdyż programy nimi objęte można bez problemu włączać do zamkniętych, komercyjnych produktów. Po drugie, do wielokrotnego licencjonowania potrzebne jest posiadanie pełni praw autorskich do kodu źródłowego programu. W dużych projektach z dużą liczbą niezależnych autorów (np. jądro Linux) uzyskanie zgody wszystkich na zmianę licencji może się okazać niemożliwe, bo np. niektórzy autorzy już nie żyją, a z kolejnymi nie ma kontaktu. Po trzecie, model ten ma największy sens wtedy, gdy potencjalni klienci mają interes w uzyskaniu licencji komercyjnej. Chodzi głównie o programy na licencji z silną klauzulą copyleft, które są podstawą do tworzenia innych programów (np. biblioteki,

156 http://www.mysql.com/products/enterprise/features.html
157 http://www.virtualbox.org/wiki/Editions
158 http://www.artifex.com/downloads/doc/Public.htm
159 L. Rosen, op. cit., s. 264–267.
160 http://ghostscript.com/awki/News
161 M. Välimäki, „*The Rise of Open Source Licensing. A Challenge to the Use of Intellectual Property in the Software Industry*", wyd. Turre Publishing, Helsinki 2005, rozdz. 7.3, „*Case Study 2: Dual Licensing and Embedded Software*", s. 206–216. Książka dostępna w Internecie: http://pub.turre.com/openbook_valimaki.pdf

bazy danych). Zastosowana w nich klauzula copyleft wymusza na twórcach programów zależnych udostępnienie kodu źródłowego ich dzieł na tej samej licencji, co używana biblioteka czy też baza danych. Dzięki możliwości zakupu licencji komercyjnej, w/w twórcy mogą za opłatą uwolnić się od wymagań klauzuli copyleft. W przypadku programów samodzielnych, nieingerujących w działanie innych programów, twórcy oprogramowania własnościowego mogą z nich korzystać bez ujawniania kodu swoich programów.

2.4. Shared Source

Shared Source[162] („współdzielony kod") jest inicjatywą Microsoftu stworzoną w celu udostępniania ich kodu źródłowego osobom i organizacjom. Inicjatywa ta powstała w maju 2001 roku jako odpowiedź na żądania dużych korporacji i organizacji rządowych, które, chcąc zapewnić sobie odpowiedni poziom bezpieczeństwa, potrzebowały wglądu do kodu źródłowego używanych programów. Shared Source można także uważać za pośrednią odpowiedź na ruch open source, zwłaszcza biorąc pod uwagę podobieństwo nazw. W ramach tej inicjatywy, organizacje (instytucje rządowe, instytucje akademickie, korporacje, producenci OEM) oraz niezależni twórcy mogą, po podpisaniu odpowiedniego kontraktu, uzyskać dostęp do kodu źródłowego wybranych programów Microsoftu. Do programu dopuszczane są tylko wybrane osoby i instytucje, po spełnieniu szeregu wymagań stawianych przez Microsoft[163].

W ramach Shared Source, Microsoft udostępnił także 5 licencji, którymi niezależni twórcy mogą objąć swoje oprogramowanie. 2 z nich, Microsoft Public License (Ms–PL) i Microsoft Reciprocal License (Ms–RL)[164], zostały zaakceptowane przez OSI jako licencje otwartego oprogramowania[165]. Z czasem FSF także je zaakceptowała, jako licencje wolnego

162 http://www.microsoft.com/resources/sharedsource/default.mspx
163 http://www.microsoft.com/resources/sharedsource/productsourceprogram.mspx
164 http://www.microsoft.com/opensource/licenses.mspx
165 http://opensource.org/node/207

oprogramowania[166], z zastrzeżeniem, żeby nie obejmować nimi nowego oprogramowania, ponieważ Ms–RL jest niekompatybilna z GPL, a Ms–PL duplikuje funkcjonalność licencji Apache 2.0.

Microsoft Public License można zakwalifikować do licencji akademickich[167], gdyż zezwala ona modyfikację i dystrybucję kodu źródłowego, oraz na włączanie kodu do własnych programów, zarówno komercyjnych jak i niekomercyjnych. Ms–PL, podobnie jak licencja Apache 2.0, zawiera także klauzulę o udzieleniu licencji na patenty oraz jej wygaśnięciu w przypadku wytoczenia autorom pozwu o naruszenie patentów. Ms–PL także wyraźnie nie udziela licencji na użycie znaków handlowych producenta. Licencja ta jest kompatybilna z trzecią wersją licencji GPL[168].

Microsoft Reciprocal License, podobnie jak Ms–PL zezwala na modyfikację i dystrybucję kodu źródłowego. Jednakże, Ms–RL jest licencją typu copyleft[169], gdyż wymaga dystrybucji zmodyfikowanych wersji pod tą samą licencją. Podobnie jak licencje MPL i CDDL, zezwala ona na włączanie do programu kodu na innych licencjach, pod warunkiem umieszczenia go w osobnych plikach. Tak samo jak MPL i CDDL, licencja Ms–RL jest niekompatybilna z GPL.

Kolejne 2 licencje, Microsoft Limited Public License (Ms–LPL)[170] i Microsoft Limited Reciprocal License (Ms–LRL)[171], są modyfikacjami w/w licencji, które ograniczają użycie programów nimi objętych tylko do systemu Microsoft Windows. Z tego powodu nie mogą one być uznane za licencje wolnego ani otwartego oprogramowania.

166 http://www.fsf.org/licensing/licenses/
167 Zob. rozdz. 2.2.2.
168 http://www.fsf.org/licensing/licenses/#ms-pl
169 Zob. rozdz. 2.2.1.
170 http://microsoft.com/resources/sharedsource/licensingbasics/limitedpubliclicense.mspx
171 http://microsoft.com/resources/sharedsource/licensingbasics/limitedreciprocallicense.mspx

Ostatnią licencją Shared Source jest Microsoft Reference Source License (Ms–RSL)[172]. Licencja ta zezwala tylko na podgląd kodu źródłowego, bez prawa do modyfikacji, oraz bez prawa do dystrybucji poza własną firmą. Jest to najbardziej restrykcyjna licencja z całej inicjatywy Microsoftu. Mimo tych ograniczeń, dany kod źródłowy może być użyty do zrozumienia działania programu i uzyskania dzięki temu poprawy działania własnych produktów współdziałających z danym programem na licencji Ms–RSL. Licencja ta również nie jest licencją wolnego ani otwartego oprogramowania.

W ramach Shared Source, Microsoft udostępnił na w/w licencjach kod źródłowy niektórych drobnych programów[173]. Microsoft wydał także kilka programów na indywidualnych licencjach, które nakładają szereg dodatkowych ograniczeń – np. zabraniają użycia danego programu do celów komercyjnych. Przykładami są Shared Source Common Language Infrastructure[174] (implementacja CLI, głównego komponentu platformy .NET) oraz system Windows CE 6.0[175].

Inicjatywa Shared Source spotkała się z szeroką krytyką ze strony środowiska wolnego oprogramowania. Richard Stallman określił program Shared Source jako „*inną nazwę dla umów o zachowaniu poufności (ang. NDA)*[176]*”*. Bruce Perens (współzałożyciel OSI) podsumował Shared Source słowami „*patrz, ale nie dotykaj – my wszystko kontrolujemy*[177]”. Michael Tiemann (prezes OSI) stwierdził, że *„Shared Source, używając podobnie brzmiących terminów i oferując podobne możliwości, służy odwróceniu uwagi i rozmyciu przekazu ruchu open source*[178]*”*. Tiemann uważa także, że Shared Source jest chwytem marketingowym, który może okazać się koniem trojańskim. Doskonałą ilustracją tego stwierdzenia jest fakt, iż twórcy wolnego oprogramowania, którzy oglądają kod udostępniony na licencji

172 http://microsoft.com/resources/sharedsource/referencesourcelicense.mspx
173 http://microsoft.com/resources/sharedsource/licensing/developer.mspx
174 http://msdn.microsoft.com/MSDN–FILES/027/002/097/ShSourceCLILicense.htm
175 http://msdn.microsoft.com/en–us/embedded/bb190212.aspx
176 Tłumaczenie własne. Źródło: http://news.zdnet.co.uk/itmanagement/0,1000000308,2109446,00.htm
177 Tłumaczenie własne. Źródło: http://www.perens.com/Articles/StandTogether.html
178 Tłumaczenie własne. Źródło: http://opensource.org/node/225

Microsoft Reference Source License mogą później zostać oskarżeni o bezprawne skopiowanie kodu objętego licencją Ms–RSL – jeśli ich kod źródłowy będzie wystarczająco podobny[179]. Miguel de Icaza, programista projektów GNOME i Mono, ostrzega twórców wolnego oprogramowania, żeby nawet nie przeglądali kodu źródłowego objętego tą licencją[180].

Nie da się ukryć, że Shared Source wykorzystuje wymienioną w rozdziale 2.2 dwuznaczność terminu open source, który potocznie może być rozumiany tylko jako możliwość zobaczenia kodu źródłowego, bez prawa do modyfikacji. Doskonale ilustruje to licencja Ms–RSL, która przez osobę nieobeznaną w temacie może zostać błędnie uznana za licencję otwartego oprogramowania. Chociaż Shared Source obejmuje 2 licencje wolnego oprogramowania (Microsoft Public License i Microsoft Reciprocal License), to całą inicjatywę należy zaliczyć do zamkniętego oprogramowania, bo większość jej licencji nie spełnia kryteriów definicji wg FSF i OSI (opisanych w rozdziale 2.2).

179 http://news.cnet.com/8301–13505_3–9790795–16.html
180 http://tirania.org/blog/archive/2007/Oct–03.html

3. Wartość modeli ze względu na wybrane kryteria porównawcze

Celem porównania opisanych wyżej modeli licencjonowania oprogramowania w niniejszej pracy zastosowano następujące kryteria: zakres uprawnień jakie nam daje autor oprogramowania, cenę oprogramowania i koszty jego utrzymania, jakość oraz organizację jego produkcji i dystrybucji.

3.1. Wolność i ograniczenia użytkowników

Zakres uprawnień udzielanych użytkownikowi oprogramowania przez jego autora jest określony w licencji oprogramowania. W momencie objęcia programu licencją pojawia się konflikt interesów autora i użytkownika – im więcej praw autor sobie zastrzeże, tym mniej praw zostanie udzielonych użytkownikowi.

Zamknięte kontra otwarte

W rozdziale 2 wyjaśniono, że oprogramowanie własnościowe opiera się na zastrzeżeniu większości praw przez autora, zaś wolne oprogramowanie opiera się na udzieleniu jak największej ilości praw użytkownikowi. Preambuła licencji GPL wyraźnie to opisuje: *„Licencje większości oprogramowania i innych dzieł zostały zaprojektowane po to, aby odebrać Ci wolność dzielenia się i zmieniania utworów. W odróżnieniu, Powszechna Licencja Publiczna GNU ma na celu zagwarantowanie Twojej swobody udostępniania i zmieniania wszystkich wersji programu – aby upewnić się, że pozostanie on wolnym oprogramowaniem dla wszystkich jego użytkowników*[181]*."*

Con Zymaris porównując licencję EULA z GPL podaje[182], że 45% treści licencji systemu Windows XP Professional służy ograniczeniu praw

181 Tłumaczenie własne, źródło: http://www.gnu.org/licenses/gpl.html
182 http://www.cybersource.com.au/about/comparing_the_gpl_to_eula.pdf

użytkownika, podczas gdy w przypadku GPLv2 jest to tylko 27%. Zaś jeśli wziąć pod uwagę rozszerzanie praw użytkownika, to tylko 15% treści licencji EULA jest temu poświęcone, a w przypadku GPL jest to aż 51%.

Dzięki posiadaniu prawa do nieskrępowanego używania, kopiowania i dystrybucji, użytkownik wolnego oprogramowania nie musi:

1. zastanawiać się na ile stanowisk wykupić licencję,
2. martwić się o dozwoloną liczbę kopii zapasowych,
3. przejmować innymi ograniczeniami, jakim podlega oprogramowanie własnościowe, a mianowicie:
    a) znika problem tzw. piractwa (nielegalnego kopiowania) – każdy może dany program kopiować i dystrybuować bez ograniczeń, i może nawet za to pobierać opłaty,
    b) użytkownik może za darmo w pełni przetestować wolne oprogramowanie przed jego szerszym zastosowaniem, lub wykupieniem związanej z nim usługi,
    c) dzięki posiadaniu prawa do modyfikacji, użytkownik może szybko i łatwo przystosować otwarty program do swoich wymagań – np.:
        - wprowadzić drobne poprawki,
        - załatać świeżo ujawnione błędy,
        - wysłać swoje modyfikacje do twórców programu.

W przypadku dużych, zamkniętych programów stworzonych przez wielkie korporacje, użytkownik ma nikłe szanse wywrzeć jakikolwiek wpływ na rozwój używanego oprogramowania – nie może go dostosować do swoich potrzeb, a na wszelkie dodatkowe funkcje lub poprawki błędów musi czekać, aż producent je zaimplementuje i udostępni.

Licencje zamkniętego oprogramowania odbierają większość praw użytkownikowi, ponieważ należą do modelu biznesowego, którego

zasadniczą częścią jest sprzedaż kopii oprogramowania. W przypadku wolnego oprogramowania stosowane są modele biznesowe opierające się na wartości dodanej (wsparciu technicznym, szkoleniach, aktualizacjach oprogramowania, i innych usługach dodatkowych). Typowi producenci zamkniętego oprogramowania traktują programy jak produkt fabryczny, zaś producenci wolnego oprogramowania traktują je jak usługę.

Zamknięte oprogramowanie często przywiązuje użytkownika do produktów jednej firmy (ang. *vendor lock–in*), podczas gdy wolne oprogramowanie daje użytkownikowi niezależność i elastyczność – w każdej chwili może on zmienić dostawcę lub samodzielnie przejąć jego rolę. Dzięki temu może on zmniejszyć lub zupełnie uniknąć kosztów ewentualnej migracji do nowego oprogramowania. Jest to szczególnie ważne, jeśli główny dostawca oprogramowania upadnie i trzeba będzie szybko znaleźć innego na jego miejsce. W przypadku produktów opartych na zamkniętym oprogramowaniu wraz z upadłością producenta kod źródłowy jego programów może zostać bezpowrotnie utracony i konieczna będzie kosztowna migracja do nowego oprogramowania. W przypadku upadku producenta otwartego oprogramowania wystarczy tylko znaleźć firmę, która będzie kontynuować jego dzieło.

Poza tym, wolnemu oprogramowaniu najczęściej towarzyszą otwarte standardy, zaś zamknięte oprogramowanie sprzyja tworzeniu zamkniętych, własnościowych rozwiązań, celowo stworzonych aby były niekompatybilne z rozwiązaniami konkurencji. Np. pakiet biurowy OpenOffice używa otwartego standardu plików ODF[183], dzięki czemu każda jego wersja jest w stanie otworzyć pliki w nim zapisane. Microsoft Office zaś, na przestrzeni lat używał różnych, niekompatybilnych ze sobą formatów plików, przez co niemożliwe jest np. otworzenie pliku z Worda 2000 w Wordzie 97. Warto także odnotować, że Microsoft wielokrotnie próbował niszczyć otwarte standardy poprzez swoją strategię „*obejmij, rozszerz, wygaś*" (ang. *embrace, extend, extinguish*). Strategia ta została opisana w (wymienionych w rozdziale

183 OpenDocument Format, http://www.oasis–open.org/specs/index.php#opendocumentv1.1

1.3.2) Dokumentach Halloween[184]. Polega ona na zaadaptowaniu otwartego standardu przez Microsoft, rozszerzeniu go o dodatkowe, nieudokumentowane cechy, i użyciu tych cech w celu zdobycia przewagi i wyeliminowania konkurencji. Przykładem zastosowania tej strategii jest protokół uwierzytelniania i autoryzacji Kerberos, którego rozszerzona implementacja w systemie Windows 2000 blokuje łączenie się z programami nie obsługującymi rozszerzeń Microsoftu[185]. Innym przykładem jest Microsoftowa implementacja platformy Java, która została zaprojektowana tak, żeby programy pod nią napisane nie mogły działać na innych systemach operacyjnych. Kolejnym przykładem jest przeglądarka Microsoft Internet Explorer, która zawiera specyficzną dla niej technologię osadzania obiektów binarnych ActiveX. Ponieważ jest ona przywiązana tylko do jednej przeglądarki i jednego systemu operacyjnego (Windows), użycie tej technologii w swojej stronie www uniemożliwia wyświetlenie jej na innych przeglądarkach i systemach operacyjnych. Oba te przykłady były przedmiotem procesu z Departamentem Sprawiedliwości USA[186].

Nie można też pominąć faktu, że to właśnie otwarte protokoły TCP/IP, SMTP oraz HTTP odpowiadają za szybkie upowszechnienie się Internetu. Dzięki ich otwartości i dostępności otwartych implementacji, każdy mógł je wykorzystać w swoich produktach bez ponoszenia jakichkolwiek opłat licencyjnych oraz bez stosowania się do jakichkolwiek restrykcji kontraktowych.

Zamknięte oprogramowanie sprzyja także tworzeniu się monopoli. Z powodu silnych efektów sieciowych, użytkownicy popularnego zamkniętego oprogramowania (np. systemu Windows) mogą nie być skłonni do korzystania z mniej popularnych produktów konkurencji. W przypadku wolnego oprogramowania monopole nie mają racji bytu, bo zawsze istnieje możliwość stworzenia niezależnej konkurencji poprzez utworzenie

184 http://www.catb.org/~esr/halloween/
185 Źródło: komentarz firmy Novell do ugody w (wspomnianym w rozdz. 1.3.2) procesie pomiędzy Departamentem Sprawiedliwości USA a Microsoftem: http://www.usdoj.gov/atr/cases/ms_tuncom/major/mtc-00029523.htm
186 Zestawienie faktów ustalonych w sprawie Departament Sprawiedliwości USA vs Microsoft: http://www.usdoj.gov/atr/cases/f3800/msjudgex.htm

rozwidlenia projektu (ang. *fork*). Na rynku może konkurować kilku producentów opierających się na tym samym wolnym kodzie, bo nie będą ze sobą konkurować na polu programistycznym, tylko na polu wartości dodanej. Podsumowanie porównania modeli biznesowych prezentuje Tab. 2.

Tab. 2: Porównanie modeli biznesowych zamkniętego i otwartego oprogramowania.

	Oprogramowanie własnościowe	**Wolne oprogramowanie**
Licencje	indywidualne, skomplikowane, rozległe	ustandaryzowane, uniwersalne, prostsze od licencji EULA
Główne źródło dochodów	sprzedaż licencji na używanie kopii oprogramowania	wartość dodana
Stosunek producenta do klienta	nacisk na zastrzeżenie praw producenta i ograniczenie wolności użytkownika	nacisk na wolność użytkownika
Oferta na rynku	jako produkt fabryczny	jako usługa
Uzależnienie od dostawcy	bardzo silne	słabe
Siła oddziaływania klientów i użytkowników	mała	średnia
Elastyczność	mała	bardzo duża
Stosunek do własności intelektualnej[187]	własnościowe rozwiązania, patenty, niejawne algorytmy	otwarte standardy, wolnodostępna dokumentacja, otwarty kod źródłowy
Kompatybilność z konkurencyjnymi rozwiązaniami	nacisk na kompatybilność z własnymi produktami, problemy z kompatybilnością z produktami konkurencji	duża przenośność, kompatybilność z większością dostępnych rozwiązań
Interes producenta	zmonopolizowanie rynku i wyeliminowanie konkurencji	tworzenie wspólnych rozwiązań infrastrukturalnych i konkurencja w obszarze wartości dodanej

Źródło: opracowanie własne.

187 Richard Stallman odrzuca termin „*własność intelektualna*", gdyż obejmując 3 różne dziedziny prawa (prawo autorskie, patentowe i prawo znaków handlowych) zniekształca ich znaczenie, sugeruje analogię do własności obiektów fizycznych oraz prowadzi do nadmiernych uogólnień. Zob. http://www.gnu.org/philosophy/not-ipr.html

Copyleft kontra licencje akademickie

W środowisku wolnego oprogramowania także istnieją rozbieżności w poglądach na ilość praw udzielanych użytkownikowi. Największy konflikt występuje pomiędzy zwolennikami licencji GPL i zwolennikami licencji akademickich (głównie BSD). Zwolennicy GPL argumentują, że ich licencja zapewnia jednakową wolność wszystkim użytkownikom, zaś ograniczenie prawa do zamknięcia kodu źródłowego traktują jak naturalną konsekwencję postawienia granicy pomiędzy własną a cudzą wolnością. Zwolennicy licencji typu BSD zaś twierdzą, że ich licencja daje twórcy więcej wolności, bo pozwala mu włączyć oprogramowanie nią objęte do zamkniętych programów. Zwolennicy GPL traktują to jako odebranie wolności kolejnym użytkownikom. Można więc stwierdzić, że GPL na pierwszym miejscu stawia wolność użytkownika (co też jest opisane w w/w preambule GPL), zaś dla licencji akademickich najważniejsza jest wolność twórcy oprogramowania (co można zaobserwować na podstawie historii systemów BSD – patrz rozdział 1.3.4). Można także stwierdzić, że GPL gwarantuje wolność od zamkniętego oprogramowania, a BSD daje wolność do tworzenia zamkniętego oprogramowania.

Zwolennicy BSD twierdzą, że ich licencja, dzięki nieskrępowanej możliwości wykorzystywania kodu źródłowego, jest bardziej przyjazna dla biznesu. Historia negatywnie weryfikuje tę tezę, gdyż żadna duża korporacja nie udostępniła żadnego swojego kluczowego produktu na licencji typu BSD. Za to istnieje wiele przykładów korporacji udostępniających swoje produkty na licencji GPL (lub innych licencjach typu copyleft): IBM, Sun, Netscape, Red Hat, Novell. Dzieje się tak dlatego, że licencje akademickie są przyjazne dla biznesu tylko „w jedną stronę". Kod na takich licencjach można bez problemu włączyć do swoich produktów bez ujawniania ich kodu źródłowego, ale żadna firma nie będzie chciała udostępnić swoich kluczowych produktów na licencji typu BSD, żeby nie tworzyć sobie w ten sposób konkurencji. W przypadku licencji typu copyleft firmy mają pewność, że żaden ich konkurent nie osiągnie znaczącej przewagi dzięki użyciu ich

kodu źródłowego, bo każde jego innowacje będą podlegały tej samej licencji, więc autorzy pierwowzoru będą mogli je włączyć do swojego kodu.

Przykładem porażki licencji akademickiej jest Wine[188] – implementacja API[189] systemu Windows dla systemów Uniksowych. Wine przez 8 lat był dystrybuowany na licencji MIT[190], ale w 2002 roku licencja została zmieniona na LGPL, ponieważ firma Transgaming Inc.[191], która na podstawie Wine stworzyła swój komercyjny produkt WineX (później nazwany Cedega), nie publikowała swoich zmian i przez to nie przyczyniała się do rozwoju programu. Transgaming wciąż rozwija i sprzedaje swój produkt, zaś od czasu zmiany licencji na LGPL, Wine pod pewnymi względami przewyższa funkcjonalność Cedegi. Warto także odnotować, że istnieje jeszcze jeden komercyjny program oparty na Wine – CrossOver. Tworząca go firma CodeWeavers[192] korzysta z Wine na licencji LGPL i aktywnie współpracuje z jego twórcami. Drugim przykładem jest system Mac OS X firmy Apple. Choć oparty został na systemie FreeBSD[193], to część stworzonych przez Apple komponentów została udostępniona na bardziej restrykcyjnej licencji Apple Public Source License[194], zaś kluczowe elementy, takie jak interfejs użytkownika Aqua, czy też interfejsy programowania aplikacji Carbon i Cocoa, są zamkniętym oprogramowaniem i ich kod źródłowy jest niedostępny.

Wolne licencje jako element strategii

Wolne licencje mogą być stosowane jako element strategii. Firmy stosujące podwójne licencjonowanie (patrz rozdział 2.3), oraz oferujące usługi dodatkowe, używają licencji typu copyleft w celu zwiększenia popularności swoich produktów i zachęcenia do kupna komercyjnej licencji na ich produkt lub usługę. Firmy mogą też udostępniać tylko część swoich

188 http://www.winehq.org
189 API (ang. *Application Programming Interface* – interfejs programowania aplikacji) – interfejs umożliwiający programom komputerowym komunikację z systemem operacyjnym lub biblioteką.
190 Zob. rozdz. 2.2.2.
191 http://www.transgaming.com/
192 http://www.codeweavers.com/
193 http://developer.apple.com/opensource/overview.html
194 http://www.opensource.apple.com/apsl/

programów na wolnej licencji, a pozostałe, będące z nimi komplementarne, mogą sprzedawać na licencji komercyjnej (np. komercyjny serwer i otwarty klient danej usługi).

Licencje z klauzulą copyleft mogą być także zastosowane w celu wyeliminowania danej dziedziny ze sfery konkurencji. Ponieważ copyleft nakłada na wszystkich twórców obowiązek udostępnienia swoich modyfikacji, to z czasem wszyscy programiści zaimplementują w swoich programach funkcjonalność z oprogramowania konkurentów. Dzięki temu, będące w obiegu wszystkie programy danego typu upodobnią się do siebie lub ich twórcy utworzą jeden wspólny projekt.

Copyleft może być także metodą na rozproszenie ryzyka projektu. Z punktu widzenia producenta, produkt na licencji copyleft zyskuje dodatkowych darmowych programistów i testerów (ochotników i pracowników innych firm). Z punktu widzenia klienta zaś, ryzyko zostaje ograniczone przez możliwość znalezienia tańszych dostawców oferujących podobny produkt lub wsparcie istniejącego.

Porównanie do wirusa

Zarówno zwolennicy zamkniętego oprogramowania, jak i zwolennicy licencji akademickich często określaną licencję GPL mianem „wirusowej licencji". Twierdzą oni, że GPL, tak jak wirus, zaraża inne programy. Tymczasem, jak zostało już wykazane w rozdziale 2.2.1, GPL nie obejmuje programów dystrybuowanych na tym samym nośniku, podobnie jak wirusy zarażają sąsiednie komórki. Licencji GPL nie można uznać za wirusa, ponieważ nie działa ona samodzielnie i nie może nigdzie wniknąć bez niczyjej wiedzy. Wyboru licencji oprogramowania oraz używanego kodu zawsze dokonuje autor, a zazwyczaj robi to świadomie. Programista może zmodyfikować i używać kod objęty licencją GPL i nie ujawniać swoich zmian, jeśli nie rozpowszechnia swojego programu ani w formie binarnej, ani w źródłowej. W przypadku dystrybucji programu niezgodnej z warunkami licencji GPL (np. po włączeniu kodu do programu na niezgodnej licencji, lub dystrybucji programu w wersji binarnej bez udostępniania źródeł) autor

kodu objętego licencją GPL może domagać się zaprzestania takich działań i skierować sprawę do sądu. Jednakże nawet w przypadku wyroku sądowego, programista łamiący licencję GPL ma wybór: może udostępnić źródła swojego programu (na licencji GPL) lub zaprzestać jego dystrybucji. Nie ma tutaj wirusowej działalności, tj. wbrew czyjejkolwiek woli. Nie ma tutaj też żadnego przymusu do wolności. Tak jak nikt nikogo nie zmusza do używania programu na licencji GPL, tak samo nikt nie zmusza nikogo do wykorzystywania kodu nią objętego. Użytkownik zawsze ma jakiś wybór.

3.2. Koszty zakupu i utrzymania

Na całkowity koszt użytkowania oprogramowania (ang. *total cost of ownership*, w skrócie *TCO*) składają się następujące składniki: koszt zakupu oprogramowania (koszt licencji), koszt wdrożenia oprogramowania (instalacja, szkolenie pracowników) oraz koszty jego późniejszego utrzymania (koszt pomocy technicznej, wynagrodzenia pracowników odpowiedzialnych za administrację oprogramowaniem).

Koszty zakupu

Typowe zamknięte oprogramowanie jest licencjonowane, gdzie klient płaci za licencję na każde stanowisko. Wolne oprogramowanie zaś jest dostępne za darmo, bądź za opłatą licencyjną, tak jak w przypadku zamkniętego oprogramowania[195]. Firmy sprzedające tzw. pudełkowe wersje wolnego oprogramowania oprócz samego oprogramowania w ramach produktu oferują także usługi – wsparcie techniczne, aktualizacje oprogramowania, certyfikację.

Firma Cybersource w opublikowanej w 2004 analizie porównawczej kosztów otwartego i zamkniętego oprogramowania podaje przykładowe koszty zakupu oprogramowania dla firmy, której infrastruktura sieciowa obejmuje 250 użytkowników[196]. W skład zestawu oprogramowania niezbędnego do funkcjonowania tej przykładowej firmy wchodzą systemy

195 Zob. rozdz. 2.2.
196 http://www.cyber.com.au/about/linux_vs_windows_tco_comparison.pdf

operacyjne (desktopowe i serwerowe), pakiety biurowe, baza danych SQL, serwer poczty, edytor grafiki oraz narzędzia dla programistów. W owej analizie całkowity koszt zakupu oprogramowania opartego na systemie Microsoft Windows wynosi 504712$, podczas gdy koszt rozwiązania opartego o system Red Hat Enterprise Linux wynosi 99279$.

Koszty utrzymania

Koszty utrzymania oprogramowania obejmują wynagrodzenia pracowników administrujących oprogramowaniem i udzielających pomocy technicznej pozostałym pracownikom korzystającym z tego oprogramowania. Mogą także obejmować straty powstałe w wyniku błędnego działania oprogramowania lub w wyniku przestojów w pracy firmy spowodowanych wyłączeniem oprogramowania podczas jego naprawy lub konserwacji.

W w/w analizie firmy Cybersource koszt utrzymania oprogramowania opartego na platformie Windows wynosi 444000$, zaś koszt utrzymania oprogramowania opartego na Linuksie wynosi 474000$. Chociaż koszty utrzymania Linuksa są nieznacznie wyższe od rozwiązania opartego na systemie Windows, to po dodaniu kosztów zakupu oprogramowania całkowity koszt utrzymania rozwiązania opartego o system Red Hat Enterprise Linux jest niższy o 255433$, co stanowi 18,69% całości kosztów.

Warto przy tym odnotować, że w w/w analizie firma Cybersource, chcąc uniknąć oskarżeń o ewentualną stronniczość (Cybersource zajmuje się dostarczaniem rozwiązań opartych na Linuksie i oprogramowaniu open source), wykonała swoje obliczenia z pominięciem kilku czynników niekorzystnych dla Microsoftu. W analizie nie uwzględniono kosztów związanych ze złośliwym oprogramowaniem (wirusy, trojany, spyware), które w praktyce dotyczą tylko systemu Windows. Aktualna lista wirusów na wolności prowadzona przez The WildList Organization International nie zawiera ani jednego wirusa na system Linux[197]. Wirusy komputerowe stanowią coraz większy koszt utrzymania oprogramowania. Washington Post

197 http://www.wildlist.org/WildList/200803.htm

podaje, że w latach 2004–2005 amerykańscy obywatele wydali 5,2 mld $ na usunięcie szkód spowodowanych przez wirusy[198]. W analizie nie uwzględniono także kosztów przestojów w pracy spowodowanych restartami lub zawieszeniami systemu. Firma Moni.tor.Us podaje, że statystycznie Linux oferuje większą dostępność niż Windows[199]. Wynika to z pełnej modularności systemu – w Linuksie każda usługa działa niezależnie od systemu i restart lub reinstalacja usługi nie wymaga restartu całego systemu, jak to często dzieje się w przypadku systemu Windows.

Inną analizę kosztów utrzymania systemów operacyjnych na zlecenie IBM–a przeprowadziła w 2005 roku firma konsultingowa Robert Frances Group[200]. W ramach tej analizy przeprowadzono wywiady w 20 firmach zatrudniających co najmniej 250 pracowników. Z owej analizy wynika, że całkowity koszt utrzymania serwerów aplikacji opartych o system Linux jest o 40% niższy, niż w przypadku systemu Windows i o 54% niższy od kosztów utrzymania serwerów opartych o system Solaris. Robert Frances Groups zwraca w swojej analizie zwraca uwagę na fakt, że koszty administrowania systemem Linux są niższe, bo 1 administrator może administrować większą liczbą systemów, niż w przypadku Windows, zaś sam system jest efektywniej wykorzystywany, bo więcej aplikacji jest na nim uruchamianych. Dzięki temu firmy mogą zaoszczędzić na sprzęcie i pensjach administratorów. Ów fakt nie został uwzględniony we wcześniej wymienionej analizie firmy Cybersource.

Na powyższych przykładach widać, że pod względem kosztów wolne oprogramowanie stanowi silną konkurencję dla zamkniętego oprogramowania. Jednakże, w powyższej analizie nie uwzględniono wszystkich możliwych kosztów związanych z wdrożeniem wolnego oprogramowania. W szczególności, nie uwzględniono kosztów migracji na nowe oprogramowanie. Koszty migracji są zależne od sytuacji firmy (liczba

198 http://washingtonpost.com/wp–dyn/content/article/2006/08/07/AR2006080701202.html
199 http://blog.mon.itor.us/?p=286
200 http://www–1.ibm.com/linux/whitepapers/robertFrancesGroupLinuxTCOAnalysis05.pdf

pracowników, ich umiejętności, przyzwyczajenia, dotychczas używane oprogramowanie) i dlatego są specyficzne dla każdego przypadku. Migracja na nowy system operacyjny może wymagać zmiany lub przepisania części dotychczas używanego oprogramowania. Z racji tego, że wiele firm stosuje oprogramowanie napisane na własne potrzeby, koszty jego przepisania pod nowy system operacyjny mogą okazać się wyższe od oszczędności uzyskanych dzięki zmianie dotychczas używanego systemu operacyjnego.

3.3. Jakość oprogramowania

Jakość oprogramowania nie jest łatwa do zmierzenia, bo zależy od wielu czynników. To co na pierwszy rzut oka wygląda na błąd używanego programu może w rzeczywistości być błędem innego programu z nim współdziałającego, systemu operacyjnego, albo nawet sprzętu. Pod pojęciem jakości oprogramowania należy rozumieć przede wszystkim liczbę błędów występujących w danym programie, a w szczególności liczbę luk bezpieczeństwa. W obszarze takiego właśnie znaczenia dokonano dalszej analizy porównawczej.

Otwartość a jakość

Wolne oprogramowanie ma zawsze jawny kod źródłowy. Ów kod jest dostępny dla każdego na takich samych warunkach, bez żadnych ograniczeń, zarówno do użytku prywatnego jak i komercyjnego. Skoro każdy ma dostęp do kodu źródłowego, to każdy może go przeanalizować i zgłosić błędy lub wysłać poprawki twórcom danego programu. Teoretycznie daje to też większe możliwości działania potencjalnym włamywaczom, bo mogą oni wykorzystać znalezione w kodzie dziury. W praktyce dostępność kodu źródłowego nie ma bezpośredniego wpływu na ilość znajdywanych błędów, bo takie poszukiwania najczęściej są przeprowadzane na skompilowanej wersji programu. Programy skompilowane do kodu maszynowego są funkcjonalnie identyczne z ich kodem źródłowym, a łatwiej jest znaleźć błąd sprawdzając jak program działa, niż czytając jego kod źródłowy.

Z otwartością kodu źródłowego wiąże się także błędne przekonanie, że skoro kod jest otwarty, to każdy może do niego wprowadzić złośliwą funkcjonalność (wirusy, konie trojańskie, tylne drzwi). Jest to niemożliwe, ponieważ każdy nowy kod jest sprawdzany przez innych programistów danego otwartego programu. Każdy może kod źródłowy przeczytać, każdy może go zmodyfikować na swoje potrzeby, ale nie każdy może dokonywać zmian w oficjalnej wersji programu. Dzięki otwartemu procesowi rozwoju, który zwykle towarzyszy otwartemu oprogramowaniu, jego autorzy częściej dostają zgłoszenia błędów od swoich użytkowników. Często dostają także poprawki błędów od niezależnych programistów (patrz Prawo Linusa[201]). Tacy programiści z czasem mogą dołączyć do projektu i stać się jego współautorami. Obserwując te zachowania programistów i użytkowników[202] Eric Raymond napisał swój esej „Katedra i bazar[203]". Stwierdził w nim, że bazarowy styl rozwoju Linuksa, w którym wszyscy mają głos i spośród ich propozycji wybierane są najlepsze, jest zadziwiająco efektywny. Pozornie chaotyczny, zdecentralizowany rozwój Linuksa, w którym nowe wersje są wypuszczane wcześnie i często, jest wg Raymonda lepszy od konwencjonalnego hierarchicznego modelu rozwoju komercyjnego oprogramowania.

Twórcy wolnego oprogramowania często porównują swoje działania do praktyk kryptologów. Sformułowana w XIX wieku przez holenderskiego kryptologa Augusta Kerckhoffsa zasada mówi, że system kryptograficzny powinien być bezpieczny nawet wtedy, gdy wszystkie szczegóły jego działania (oprócz klucza) są znane. Jej przeciwieństwem jest tzw. „*security through obscurity*" (bezpieczeństwo dzięki niejawności) – nieujawnianie zasad działania systemu w celu zapewnienia jego bezpieczeństwa. Idea ta jest promowana przez większość twórców zamkniętego oprogramowania, w tym Microsoft. W praktyce daje ona tylko złudne poczucie bezpieczeństwa, gdyż to, że dane luki w oprogramowaniu nie są publicznie znane nie

201 „*Przy wystarczającej liczbie oczu wszystkie błędy są proste*" – zob rozdz. 1.3.1.
202 Zachowania organizacyjne twórców wolnego oprogramowania szerzej opisuje rozdz. 3.4.
203 E. S. Raymond, op. cit.

oznacza, iż nie są one masowo wykorzystywane przez wybrane osoby, które o tych lukach wiedzą, ale nie dzielą się swoją wiedzą z innymi. W przypadku publicznego ujawnienia informacji o lukach bezpieczeństwa w danym programie, osoby go używające mogą przedsięwziąć odpowiednie środki zapobiegające wykorzystaniu danej luki, które w konsekwencji mogą pomóc w uchronieniu się przed zagrożeniem.

Statystyczna ilość błędów w kodzie

Jedną z najczęściej stosowanych miar oceny jakości oprogramowania jest ilość wykrytych błędów podzielona przez ilość linii kodu danego programu. Nie jest to miara doskonała, bo ten sam kod źródłowy może być rozciągnięty na wiele krótkich linii lub zmieszczony w jednej, za to bardzo długiej. Wg badań naukowców z Uniwersytetu Carnegie Mellon, typowe oprogramowanie komercyjne posiada 20–30 błędów na 1000 linii kodu[204]. Andrew Tanenbaum, powołując się na publikację Ostranda i Weyukera[205], twierdzi, że oprogramowanie zawiera od 2 do 75 błędów na 1000 linii kodu[206]. Firma Coverity od 2006 roku prowadzi regularne badania jakości kodu programów będących wolnym oprogramowaniem na zlecenie Departamentu Bezpieczeństwa Krajowego Stanów Zjednoczonych[207]. Coverity oferuje za darmo projektom open source swoje usługi oparte na oprogramowaniu do automatycznej analizy kodu źródłowego. Obecnie w usłudze Scan zarejestrowanych jest ponad 180 projektów. Firma oferuje także swoje usługi komercyjnie wszystkim producentom zamkniętego oprogramowania[208].

Przed założeniem firmy Coverity, jej założyciele przeprowadzili analizę kodu Linuksa w ramach pracy na Uniwersytecie Stanforda. Podczas

204 Cyt. za *Coverity's kernel code quality study*, LWN.net, 2004, http://lwn.net/Articles/115530/
205 T. J. Ostrand, E. J. Weyuker, *The distribution of faults in a large industrial software system,* Proc. Int'l Symp. on Software Testing and Analysis, ACM, 2002, s. 55–64.
206 A. S. Tanenbaum, J. N. Herder, H. Bos, *Can We Make Operating Systems Reliable and Secure?*, IEEE Computer, vol. 39, s. 44–51, V 2006, http://www.cs.vu.nl/~ast/publications/computer–2006a.pdf
207 http://scan.coverity.com/
208 http://www.coverity.com/

4–letniego badania trwającego od 2000 roku[209], w 5,7 mln linii kodu źródłowego znaleziono 985 błędów, co daje 5,787 błędu na 1000 linii kodu. Obecnie strona scan.coverity.com podaje, że w Linuksie jest 0,127 błędu na 1000 linii kodu[210]. Inne znane otwarte programy również plasują się poniżej w/w średniej ilości błędów. Dla serwera HTTP Apache średnia ilość błędów znalezionych przez usługę Scan wynosi 0,14 błędu na 1000 linii kodu. Dla przeglądarki WWW Mozilla Firefox wartość współczynnika ilości błędów do ilości linii kodu źródłowego wynosi 0,162. Zaś dla bazy danych PostgreSQL współczynnik ten wynosi 0,041. Oczywiście nie oznacza to, że w/w programy nie zawierają więcej błędów. Z biegiem czasu kolejne błędy są znajdowane, a wraz z rozszerzaniem funkcjonalności nieświadomie mogą być wprowadzane nowe błędy. Owe współczynniki powinny stanowić tylko punkt orientacyjny przy ewentualnych ocenach i porównaniach.

Chociaż firma Coverity prowadzi podobne badania na zamkniętym oprogramowaniu, to nie ujawnia ich wyników. Ben Chelf, współzałożyciel firmy Coverity, w wywiadzie dla tygodnika BusinessWeek podaje, że otwarte oprogramowanie przeciętnie jest wyższej jakości niż zamknięte oprogramowanie[211]. Jednocześnie Ben Chelf twierdzi, że spośród ponad 150 programów przeanalizowanych przez jego firmę, najwyższą jakość wykazało 15 programów, z których aż 11 to programy o zamkniętym kodzie źródłowym. Wg niego, najlepsze zamknięte programy można znaleźć w zastosowaniach krytycznych, takich jak lotnictwo, medycyna, telekomunikacja, czy też przemysł kosmiczny.

Warto zauważyć w tej argumentacji, że Chelf pomija bardzo istotny fakt, iż są to zastosowania szczególnego przeznaczenia, w których ilość sprzętu, na którym ma działać oprogramowanie jest ściśle ograniczona, tak samo jak funkcjonalność całego urządzenia. Porównywane przez niego otwarte programy mają zastosowanie ogólne, i dlatego muszą obsługiwać więcej sprzętu, a ich funkcjonalność jest ciągle rozwijana. Dlatego też

209 *Coverity's kernel code quality study*, LWN.net, 2004, http://lwn.net/Articles/115530/
210 http://scan.coverity.com/rung1.html
211 http://www.businessweek.com/technology/content/oct2006/tc20061006_394140.htm

w programach ogólnego przeznaczenia można znaleźć więcej błędów. Większość otwartego oprogramowania stanowią właśnie programy przeznaczone dla ogółu, gdyż dzięki swojej ogólności znajdują więcej użytkowników, a przez to także więcej osób chętnych je rozwijać. Szczególnie wyspecjalizowanym programom trudno jest znaleźć nowych użytkowników i współtwórców, dlatego w ich przypadku lepiej się sprawdza model zamkniętego oprogramowania.

Czas potrzebny do załatania błędów

Inną miarą jakości oprogramowania jest czas jaki upłynął od momentu odkrycia błędów do ich załatania. Również nie jest to miara doskonała bo zależy od poziomu zagrożenia jakie te błędy niosą. Praktyka pokazuje, że twórcy otwartego oprogramowania reagują znacznie szybciej na zgłoszone błędy niż twórcy zamkniętego oprogramowania. Microsoft publikuje swoje biuletyny bezpieczeństwa w drugi wtorek każdego miesiąca. Kiedy pojawią się jakieś krytyczne dziury w oprogramowaniu, na wydanie poprawek trzeba w najgorszym wypadku czekać co najmniej 30 dni. Twórcy wolnego oprogramowania publikują łaty na krytyczne błędy jak tylko zostaną ukończone, bez czekania do jakiejś z góry wyznaczonej daty. Raport 4 badaczy uniwersytetu Carnegie Mellon podaje, że twórcy otwartego oprogramowania są o 71% szybsi w łataniu znanych błędów niż twórcy zamkniętego oprogramowania[212].

Ostateczna ocena jakości oprogramowania zależy od interpretacji wszystkich dostępnych wskaźników. Na w/w przykładach widać, że wolne oprogramowanie może wykazywać porównywalną, jak i wyższą jakość od zamkniętego oprogramowania. Jednakże, nie ma tutaj reguły – każdy przypadek należy porównywać osobno.

212 A. Arora, R. Krishnan, R. Telang, Y. Yang, *An Empirical Analysis of Software Vendors' Patching Behavior: Impact of Vulnerability Disclosure*, 2006, http://www.heinz.cmu.edu/~rtelang/disclosure_jan_06.pdf

3.4. Organizacja produkcji i rozwoju oprogramowania

Specyficznym elementem odróżniającym zamknięty i otwarty model licencjonowania oprogramowania są towarzyszące im modele rozwoju oprogramowania. Tradycyjnie pojmowane oprogramowanie komercyjne jest tworzone przez firmę w zamkniętym procesie. Cały rozwój typowego zamkniętego oprogramowania odbywa się za zamkniętymi drzwiami: projektowanie, pisanie kodu, testowanie. Program jest udostępniany klientom dopiero po ukończeniu jego procesu produkcji. Ów model produkcji bywa także stosowany dla otwartego oprogramowania, jednak większość projektów będących wolnym oprogramowaniem jest oparta na otwartym modelu rozwoju. Model ten, zapoczątkowany przez Linusa Torvaldsa przy tworzeniu Linuksa, został opisany przez Erica Raymonda w eseju „Katedra i bazar[213]".

Raymond, obserwując rozwój Linuksa stwierdził, że przypomina on wielki, hałaśliwy bazar, pełen różnych poglądów i planów. Było to zupełnym przeciwieństwem klasycznego stylu tworzenia oprogramowania, który porównał do budowania katedry. Przed poznaniem Linuksa Eric Raymond uważał, że po przekroczeniu pewnego poziomu złożoności programu należy zastosować bardziej scentralizowane podejście do tworzenia oprogramowania. Wg jego ówczesnych poglądów, najważniejsze programy powinny powstawać jak katedry, budowane przez dostojnych architektów pracujących w pełnym odosobnieniu, bez udostępniania wersji testowych przed czasem. Linus Torvalds zaś wypuszczał nowe wersje Linuksa wcześnie i często, i był bardzo otwarty na wszelkie sugestie. Raymondowi wtedy wydawało się, że z tak chaotycznego procesu rozwoju nie może wyjść nic spójnego i stabilnego. Później z zaskoczeniem odkrył, że ten styl działa, i to działa bardzo dobrze. Dzięki częstemu wydawaniu nowych wersji Linux był szybciej testowany, a wykryte w nim błędy szybciej naprawiane. Zaś dzięki otwartej postawie Torvaldsa rozwój Linuksa nabrał niesamowitego tempa. Z całego świata napływały do niego kolejne fragmenty kodu dodające nową

213 E. S. Raymond, op. cit.

funkcjonalność lub naprawiające znalezione błędy. Niebagatelny wpływ na to miała rosnąca popularność sieci Internet, dzięki której taka luźna współpraca wielu osób z różnych krajów była możliwa.

Na otwarty model rozwoju oprogramowania składają się trzy podstawowe, publicznie dostępne komponenty: system śledzenia błędów[214], system kontroli wersji[215], oraz dowolne narzędzie komunikacyjne (lista dyskusyjna, forum, wiki[216], kanał IRC[217], itd.). Dostęp do każdego z nich jest otwarty: każdy może zgłaszać błędy, każdy może sprawdzić status prac nad zgłoszonymi błędami, każdy może śledzić rozwój programu w systemie kontroli wersji, każdy może zgłaszać swoje sugestie autorom programu. Jest to całkowite przeciwieństwo w stosunku do zamkniętego oprogramowania, gdzie przeciętny użytkownik nie ma żadnej możliwości aby wpłynąć na jego rozwój. Producenci zamkniętego oprogramowania nie oferują użytkownikom żadnej możliwości obserwowania rozwoju kodu źródłowego ich programów, ani żadnej możliwości sprawdzenia stanu zgłoszonych błędów. Zaś możliwości zgłaszania błędów i sugestii są ograniczone. Np. Microsoft umożliwia swoim użytkownikom zgłaszanie błędów tylko przez telefon[218], Corel nie oferuje żadnej możliwości zgłaszania błędów[219], Adobe zaś udostępnia formularz do zgłaszania błędów i sugestii[220], jednakże po wysłaniu zgłoszenia nie można nigdzie śledzić jego statusu, ani sprawdzić cudzych zgłoszeń. Przyczyną takiego stanu rzeczy jest założenie, iż wszystkie błędy zostaną wykryte na etapie produkcji. Praktyka pokazuje, że nie ma oprogramowania bez błędów, i że zawsze będą znajdywane nowe błędy. Otwarty model tworzenia oprogramowania zaś zakłada, że programy

214 System śledzenia błędów – oprogramowanie służące do zgłaszania błędów i śledzenia ich stanu. Przykład: system śledzenia błędów w Ubuntu (dystrybucji Linuksa) – https://bugs.launchpad.net/ubuntu/

215 System kontroli wersji – oprogramowanie służące do śledzenia i łączenia zmian w kodzie źródłowym dokonywanych przez wiele osób w różnych momentach. Przykład: system kontroli wersji jądra Linux – http://git.kernel.org/

216 Wiki – oprogramowanie umożliwiające wspólną pracę wielu użytkowników przy tworzeniu treści stron internetowych, zob. http://pl.wikipedia.org/wiki/Wiki

217 IRC – usługa umożliwiająca jednoczesną rozmowę wielu osób w czasie rzeczywistym, zob. http://pl.wikipedia.org/wiki/IRC

218 http://support.microsoft.com/gp/contactbug/

219 http://www.corel.com/servlet/Satellite/us/en/Content/1152796555474

220 http://www.adobe.com/cfusion/mmform/index.cfm?name=wishform

są w ciągłym rozwoju, i dlatego tak dużą rolę odgrywają w nim publiczne systemy śledzenia błędów i kontroli wersji.

Bardzo częstym problemem w rozproszonych projektach będących wolnym oprogramowaniem jest zarządzanie prawami autorskimi do kodu źródłowego. Przy dużej liczbie niezależnych twórców brak prowadzenia ścisłej kontroli nad własnością kodu źródłowego może przynieść spore problemy. Bez dokładnego sprawdzania autorstwa kodu dany projekt może się narazić na proces, podobny do tego jaki spotkał system BSD[221]. Brak informacji o autorstwie danych fragmentów kodu może być też problematyczny przy próbie zmiany licencji całego programu. Kod nieznanego pochodzenia w takim przypadku musiałby zostać napisany od nowa, gdyż nieznanego (lub nieżyjącego już) autora nie da się spytać o zgodę na zmianę licencji. Wiele projektów będących wolnym oprogramowaniem radzi sobie z tym problemem stosując tzw. CLA – ang. *Contributor License Agreement* – umowę licencyjną współtwórcy. Każdy programista biorący udział w takim projekcie musi podpisać taką umowę, poprzez którą przekazuje swoje prawa autorskie na rzecz fundacji bądź firmy zarządzającej projektem. Dzięki takim umowom dany rozproszony projekt uzyskuje 1 przedstawiciela, który może w razie potrzeby podjąć decyzję o uaktualnieniu lub zmianie licencji, i który będzie odpowiadał za ewentualne naruszenia cudzych praw autorskich lub patentów, zaś programiści udostępniający projektowi swój kod zyskują ochronę przed ewentualnymi roszczeniami prawnymi. Przykładami projektów używających takich licencji są: Apache[222], MySQL[223], OpenOffice.org[224], Fedora i Projekt GNU. Kontrprzykładem jest Linux, którego autorzy nie stosują takich umów.

Jednym z najpowszechniejszych mitów dotyczących rozwoju wolnego oprogramowania jest stwierdzenie, że jest ono rozwijane niekomercyjnie, głównie przez anonimowych, niezależnych twórców. Na przykładzie Linuksa można łatwo sprawdzić kto odpowiada za rozwój wolnego oprogramowania.

221 Zob. rozdz. 1.3.4.
222 http://www.apache.org/licenses/icla.txt
223 http://forge.mysql.com/contribute/cla.php
224 http://www.openoffice.org/licenses/sca.pdf

Raport opublikowany przez Linux Foundation podaje, że ponad 70% pracy w rozwijaniu Linuksa wykonują pracownicy różnych firm[225]. W ciągu ostatnich 3 lat 13,9% zmian w kodzie Linuksa zostało dokonanych przez niezależnych programistów. 12,9% zmian wykonali programiści, co do których nie można było ustalić czy ich praca była opłacana przez jakąś firmę. 11,2% zmian dokonali pracownicy firmy Red Hat; 8,9% – pracownicy firmy Novell; 8,3% – IBM; 4,1% – Intel; 2,6% – Linux Foundation; 2,5% – Consultant; 2,0% – SGI; 1,6% – MIPS Technologies; 1,3% – Oracle; 1,2% – MontaVista; 1,1% – Google; 1% – Linuxtronix; 0,9% – HP. Pozostałe niecałe 27% zmian dokonanych zostało przez inne firmy, których poszczególny udział w zmianach nie przekroczył 1%. Nawet przy założeniu, że wszyscy programiści, których pracodawcy nie udało się ustalić pracowali za darmo, z czysto altruistycznych pobudek, wyraźnie widać, że bezwzględna większość rozwoju Linuksa jest dziełem różnych firm, wśród których można znaleźć wielu znaczących przedstawicieli rynku IT.

225 https://www.linux-foundation.org/publications/linuxkerneldevelopment.php

Zakończenie

> *„Jeśli ty masz jabłko i ja mam jabłko i wymienimy się tymi jabłkami, to wtedy ty i ja wciąż będziemy mieli po 1 jabłku. Ale jeśli ty masz pomysł i ja mam pomysł i wymienimy się tymi pomysłami, to wtedy oboje będziemy mieli 2 pomysły."*
>
> George Bernard Shaw

Powyższa praca nie wyczerpuje tematyki licencjonowania oprogramowania. Wiele zagadnień, które wymagałyby dokładnego omówienia zostało potraktowanych skrótowo lub nawet pominiętych. Celem niniejszej pracy było jednak rozpoznanie i porównanie dzisiejszych modeli licencjonowania oprogramowania oraz udzielenie odpowiedzi na pytanie: jakie są ich zalety i wady?

Reasumując wszystkie analizy przedstawione w niniejszej pracy, można stwierdzić, że oba wyróżnione przeze mnie modele licencjonowania mają wiele zalet i doskonale się sprawdzają w konkretnych, zależnych od kontekstu sytuacjach.

Zdaniem autora licencje **wolnego oprogramowania** lepiej sprawdzają się w projektach infrastrukturalnych, takich jak: systemy operacyjne, przeglądarki www, serwery, popularne aplikacje ogólnego przeznaczenia. Wolne licencje doskonale nadają się do tworzenia referencyjnych implementacji algorytmów (np. obsługi protokołów internetowych, kryptografii, kodeków obrazu i dźwięku, formatów dokumentów). Bez otwartych programów takich jak Bind, Apache, czy OpenSSH – używanych na większości serwerów – nie byłoby Internetu w tej formie co dzisiaj. Wolne oprogramowanie nadaje się także do projektów rządowych i edukacyjnych, które z zasady mają dotrzeć do jak największej ilości odbiorców przy jak najniższym koszcie.

Licencje **zamkniętego oprogramowania** (EULA) lepiej sprawdzają się w projektach niszowych lub wymagających wysokich nakładów, takich jak specjalistyczne oprogramowanie, np. do projektowania samolotów. Wynika

to z faktu, że dla projektów niszowych trudno jest znaleźć szerokie grono użytkowników skłonnych do współpracy przy jego tworzeniu. Osobnym przypadkiem są gry komputerowe, które w większej części składają się z utworów graficznych i dźwiękowych (a nie z kodu źródłowego).

Warto także wspomnieć o wpływie, jaki wolne oprogramowanie wywarło na inne dziedziny. Licencje wolnej dokumentacji utorowały drogę organizacji Creative Commons, która zasady wolnego oprogramowania przeniosła na grunt kultury. Dzięki nim, twórcy utworów muzycznych, graficznych, audiowizualnych, literackich i innych, mogą w wielu przypadkach swobodnie korzystać z dorobku innych twórców, a konsumenci kultury mogą się nią swobodnie dzielić. W świecie nauki zaś, udostępnianie wyników swojej pracy bez ograniczeń jest na porządku dziennym. Dzięki swobodnej wymianie wiedzy osiągnęliśmy tak daleki rozwój cywilizacyjny. Wolne oprogramowanie można także uznać za naturalną odpowiedź na zbyt restrykcyjny system prawa autorskiego – kiedy ustawodawca dał twórcom zbyt duży zakres uprawnień, to część z nich świadomie z tych praw zrezygnowała na rzecz użytkowników.

Spisy rysunków i tabel

Spis rysunków

Spis tabel

Bibliografia

Źródła książkowe

1. R. Dixon, „*Open Source Software Law*", wyd. Artech House, Norwood, Massachusetts 2004.
2. M. K. McKusick, „*Twenty Years of Berkeley Unix: From AT&T–Owned to Freely Redistributable*", w: „*Open Sources: Voices from the Open Source Revolution*", wyd. O'Reilly Media, Sebastopol, California 1999.
3. „*Prawo Internetu*", red. P. Podrecki, Wydawnictwo Prawnicze LexisNexis, Warszawa 2007.
4. E. S. Raymond, „*The Cathedral & The Bazaar. Musings on Linux and Open Source by an Accidental Revolutionary*", wyd. O'Reilly Media, Sebastopol, California 1999.
5. L. Rosen, „*Open Source Licensing. Software Freedom and Intellectual Property Law*", wyd. Prentice Hall PTR, Upper Saddle River, New Jersey 2004.
6. A. M. St. Laurent, „*Understanding Open Source and Free Software Licensing*", wyd. O'Reilly Media, Sebastopol, California 2004.
7. R. M. Stallman, L. Lessig, J. Gay, „*Free Software, Free Society: Selected Essays of Richard M. Stallman*", wyd. GNU Press, Boston, Massachusetts 2002.
8. M. Välimäki, „*The Rise of Open Source Licensing. A Challenge to the Use of Intellectual Property in the Software Industry*", wyd. Turre Publishing, Helsinki 2005.
9. S. Williams, „*W obronie wolności. Krucjata hakera na rzecz Wolnego Oprogramowania*" (w oryg. „*Free as in Freedom: Richard Stallman's Crusade for Free Software*"), tłum. K. Masłowski, wyd. Helion, 2003.

Artykuły

1. J. Myers, „*Apple v. Microsoft: Virtual Identity in the GUI Wars*", Richmond Journal of Law & Technology, 1999.
2. T. J. Ostrand, E. J. Weyuker, „*The distribution of faults in a large industrial software system*", Proc. Int'l Symp. on Software Testing and Analysis, ACM, 2002.
3. A. S. Tanenbaum, J. N. Herder, H. Bos, „*Can We Make Operating Systems Reliable and Secure?*", IEEE Computer, vol. 39, V 2006.

Źródła prawne

1. Dyrektywa Rady EWG z dnia 14 maja 1991 r. w sprawie ochrony prawnej programów komputerowych (91/250/EWG).
2. Konwencja o udzielaniu patentów europejskich, Dz. U. 2004 nr 79 poz. 737.
3. Ustawa z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych, Dz. U. 1994 nr 24 poz. 83.

Źródła internetowe

Wszystkie odnośniki zostały sprawdzone 4 V 2008.

1. ftp://ftp.cs.berkeley.edu/pub/4bsd/README.Impt.License.Change
2. http://blog.mon.itor.us/?p=286
3. http://catb.org/~esr/halloween/
4. http://catb.org/~esr/writings/cathedral-bazaar/
5. http://cvsweb.netbsd.org/bsdweb.cgi/~checkout~/src/distrib/notes/common/legal.common?rev=1.47
6. http://dev.perl.org/licenses/
7. http://developer.apple.com/opensource/
8. http://developer.apple.com/opensource/overview.html
9. http://distrowatch.com
10. http://ec.europa.eu/comm/competition/antitrust/cases/microsoft/
11. http://en.wikipedia.org/wiki/Image:Copyleft.svg
12. http://en.wikipedia.org/wiki/Otherware
13. http://en.wikipedia.org/wiki/Software_patents_under_United_States_patent_law
14. http://encyklopedia.interia.pl/haslo?hid=92188
15. http://encyklopedia.pwn.pl/haslo.php?id=3951439
16. http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31991L0250:PL:HTML
17. http://freshmeat.net/stats/#license
18. http://futurist.se/gldt/
19. http://ghostscript.com/awki/News
20. http://gpl3.palamida.com
21. http://groups.google.pl/group/net.unix-wizards/msg/4dadd63a976019d7

22. http://kernel.org/#whatislinux
23. http://law.richmond.edu/jolt/v1i1/myers.html
24. http://legalpad.blogs.fortune.cnn.com/2007/05/13/msft-linux-free-software-infringe-235-of-our-patents/
25. http://lwn.net/Articles/115530/
26. http://microsoft.com/presspass/press/2006/nov06/11-02MSNovellPR.mspx
27. http://microsoft.com/resources/sharedsource/default.mspx
28. http://microsoft.com/resources/sharedsource/licensing/developer.mspx
29. http://microsoft.com/resources/sharedsource/productsourceprogram.mspx
30. http://money.cnn.com/magazines/fortune/fortune_archive/2007/05/28/100033867/
31. http://news.cnet.com/8301-13505_3-9790795-16.html
32. http://news.zdnet.co.uk/itmanagement/0,1000000308,2109446,00.htm
33. http://news.zdnet.co.uk/software/0,1000000121,39287338,00.htm
34. http://opensource.org/docs/certification_mark.html
35. http://opensource.org/docs/osd
36. http://opensource.org/history
37. http://opensource.org/licenses/category
38. http://opensource.org/node/207
39. http://opensource.org/node/225
40. http://opensource.org/trademarks/osi-certified/
41. http://oreilly.com/catalog/osfreesoft/book/
42. http://pl.wikipedia.org/wiki/Bulletin_Board_System
43. http://pl.wikipedia.org/wiki/Haker_(slang_komputerowy)
44. http://pl.wikipedia.org/wiki/IRC
45. http://pl.wikipedia.org/wiki/OEM
46. http://pl.wikipedia.org/wiki/Wiki
47. http://pub.turre.com/openbook_valimaki.pdf
48. http://rosenlaw.com/oslbook.htm
49. http://scan.coverity.com/
50. http://scan.coverity.com/rung1.html
51. http://stallman.helion.pl

52. http://support.microsoft.com/gp/contactbug/
53. http://tirania.org/blog/archive/2007/Oct-03.html
54. http://trolltech.com/company/model
55. http://washingtonpost.com/wp-dyn/content/article/2006/08/07/AR2006080701202.html
56. http://wiki.oldos.org/Dos/OriginOfDos
57. http://wp.netscape.com/newsref/pr/newsrelease591.html
58. http://www-03.ibm.com/ibm/history/history/decade_1960.html
59. http://www-1.ibm.com/linux/whitepapers/robertFrancesGroupLinuxTCOAnalysis05.pdf
60. http://www.adobe.com/cfusion/mmform/index.cfm?name=wishform
61. http://www.asp-shareware.org/users/history-of-shareware.asp
62. http://www.bell-labs.com/history/unix/
63. http://www.blinkenlights.com/classiccmp/gateswhine.html
64. http://www.businessweek.com/technology/content/oct2006/tc20061006_394140.htm
65. http://www.codeweavers.com/
66. http://www.corel.com/servlet/Satellite/us/en/Content/1152796555474
67. http://www.coverity.com/
68. http://www.cs.vu.nl/~ast/publications/computer-2006a.pdf
69. http://www.cyber.com.au/about/linux_vs_windows_tco_comparison.pdf
70. http://www.cybersource.com.au/about/comparing_the_gpl_to_eula.pdf
71. http://www.debian.org/social_contract#guidelines
72. http://www.dwheeler.com/essays/gpl-compatible.html
73. http://www.eff.org/wp/dangerous-terms-users-guide-eulas
74. http://www.epo.org/patents/law/legal-texts/html/epc/2000/e/ar52.html
75. http://www.everything2.com/index.pl?node=BSD%20Code%20in%20Windows
76. http://www.filetiger.com/articles/spyware.html
77. http://www.forbes.com/forbes/2005/0704/071.html
78. http://www.free-soft.org/mirrors/www.opensource.org/docs/osd-polish.php
79. http://www.fsf.org/licensing/licenses/
80. http://www.fsf.org/licensing/licenses/quick-guide-gplv3.html

81. http://www.fsf.org/licensing/sco/sco-preemption.html
82. http://www.gnu.org/copyleft/copyleft.pl.html
83. http://www.gnu.org/gnu/linux-and-gnu.html
84. http://www.gnu.org/gnu/thegnuproject.pl.html
85. http://www.gnu.org/licenses/gpl-faq.html
86. http://www.gnu.org/licenses/why-not-lgpl.html
87. http://www.gnu.org/philosophy/bsd.pl.html
88. http://www.gnu.org/philosophy/free-sw.pl.html
89. http://www.gnu.org/philosophy/fsfs/rms-essays.pdf
90. http://www.gnu.org/philosophy/not-ipr.html
91. http://www.gnu.org/philosophy/open-source-misses-the-point.html
92. http://www.groklaw.net/staticpages/index.php?page=20070810205256644
93. http://www.heinz.cmu.edu/~rtelang/disclosure_jan_06.pdf
94. http://www.informationweek.com/news/showArticle.jhtml?articleID=199600443
95. http://www.linux-foundation.org/publications/linuxkerneldevelopment.php
96. http://www.linux-watch.com/news/NS7235986827.html
97. http://www.math.utah.edu/ftp/pub/tex/bib/toc/dr-dobbs-1980.html#10(3):March:1985
98. http://www.mozilla.org/MPL/
99. http://www.mysql.com/products/enterprise/features.html
100. http://www.mysql.com/products/which-edition.html
101. http://www.netbsd.org/about/redistribution.html#default
102. http://www.oasis-open.org/specs/index.php#opendocumentv1.1
103. http://www.oreilly.com/catalog/opensources/book/kirkmck.html
104. http://www.os2bbs.com/os2news/OS2History.html
105. http://www.perens.com/Articles/StandTogether.html
106. http://www.softwarehistory.org/history/d_60s.html
107. http://www.sun.com/smi/Press/sunflash/2005-01/sunflash.20050125.1.xml
108. http://www.theregister.co.uk/2000/07/31/ms_ballmer_linux_is_communism/
109. http://www.theregister.co.uk/2001/06/02/ballmer_linux_is_a_cancer/

110. http://www.theregister.co.uk/2001/06/20/gpl_pacman_will_eat_your/
111. http://www.theregister.co.uk/2002/04/22/gates_gpl_will_eat_your/
112. http://www.thocp.net/companies/microsoft/microsoft_company_part2.htm
113. http://www.thocp.net/companies/microsoft/microsoft_company.htm
114. http://www.transgaming.com/
115. http://www.unix.org/what_is_unix/history_timeline.html
116. http://www.usdoj.gov/atr/cases/f3800/msjudgex.htm
117. http://www.usdoj.gov/atr/cases/ms_index.htm
118. http://www.usdoj.gov/atr/cases/ms_tuncom/major/mtc-00029523.htm
119. http://www.virtualbox.org/wiki/Editions
120. http://www.wildlist.org/WildList/200803.htm
121. http://www.winehq.org

Licencje

1. Microsoft Windows XP Professional EULA
   http://download.microsoft.com/documents/useterms/Windows%20XP_Professional_Polish_a675687b-f2e8-4c76-b108-9657210d2135.pdf
2. GNU General Public License 1.0
   http://www.gnu.org/licenses/old-licenses/gpl-1.0.txt
3. GNU General Public License 2.0
   http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
4. GNU Lesser General Public License 2.1
   http://www.gnu.org/licenses/old-licenses/lgpl-2.1.html
5. GNU General Public License 3.0
   http://www.gnu.org/licenses/gpl.html
6. GNU Lesser General Public License 3.0
   http://www.gnu.org/licenses/lgpl.html
7. GNU Affero General Public License 3.0
   http://www.gnu.org/licenses/agpl.html

8. Nieoficjalne tłumaczenie licencji GPLv2
   http://gnu.org.pl/text/licencja-gnu.html
9. Affero General Public License 1.0
   http://www.affero.org/oagpl.html
10. Mozilla Public License 1.1
   http://www.mozilla.org/MPL/MPL-1.1.html
11. Common Development and Distribution License
   http://www.sun.com/cddl/cddl.html
12. Licencja BSD (4-klauzulowa)
   http://www.freebsd.org/copyright/license.html
13. Licencja BSD (3-klauzulowa)
   http://www.opensource.org/licenses/bsd-license.php
14. Licencja BSD (2-klauzulowa)
   http://www.gnu.org/licenses/info/BSD_2Clause.html
15. Licencja MIT
   http://www.opensource.org/licenses/mit-license.php
16. Licencja ISC
   http://www.openbsd.org/cgi-bin/cvsweb/~checkout~/src/share/misc/license.template
17. Apache License 2.0
   http://www.apache.org/licenses/
18. Alladin Free Public License
   http://www.artifex.com/downloads/doc/Public.htm
19. Microsoft Public License, Microsoft Reciprocal License
   http://www.microsoft.com/opensource/licenses.mspx
20. Microsoft Limited Public License
   http://microsoft.com/resources/sharedsource/licensingbasics/limitedpubliclicense.mspx
21. Microsoft Limited Reciprocal License
   http://microsoft.com/resources/sharedsource/licensingbasics/limitedreciprocallicense.mspx

22. Microsoft Reference Source License
http://microsoft.com/resources/sharedsource/referencesourcelicense.mspx
23. Microsoft Shared Source CLI, C#, and JScript License
http://msdn.microsoft.com/MSDN-FILES/027/002/097/ShSourceCLILicense.htm
24. Microsoft Windows Embedded CE 6.0 Shared Source License
http://msdn.microsoft.com/en-us/embedded/bb190212.aspx
25. Apple Public Source License
http://www.opensource.apple.com/apsl/
26. Apache Software Foundation Individual Contributor License Agreement
http://www.apache.org/licenses/icla.txt
27. MySQL Contributor License Agreement
http://forge.mysql.com/contribute/cla.php
28. Sun Microsystems, Inc. Contributor Agreement
http://www.openoffice.org/licenses/sca.pdf