Studia i Materiały Informatyki Stosowanej, Tom 5, Nr 11, 2013
str. 33-39

ZASTOSOWANIE WIRTUALNEJ TAŚMY PRODUKCYJNEJ DO ANALIZY PROCESU TWORZENIA OPROGRAMOWANIA

Julia Siderska

Politechnika Białostocka
Wydział Zarządzania
ul. Tarasiuka 2, 16-001 Kleosin
e-mail: j.siderska@pb.edu.pl

Streszczenie: *Celem artykułu jest pokazanie, w jaki sposób wirtualna taśma produkcyjna (WTP) może być wykorzystywana jako model do analizy procesu tworzenia oprogramowania open source, czyli takiego, do którego kodów źródłowych mamy pełen, wolny i swobodny dostęp. Termin wirtualna taśma produkcyjna (WTP), zaproponowany został w 2006 roku przez S. Walukiewicza i jest efektem uogólnienia pojęcia klasycznej taśmy produkcyjnej, wprowadzonej przez H. Forda w przemyśle samochodowym Obecnie otwarte aplikacje są budowane, modyfikowane, poprawiane i rozpowszechniane dzięki wspólnej pracy twórczej programistów z całego świata. To właśnie kapitał ludzki każdego z tych twórców/programistów, czyli ich pasja, umiejętność programowania oraz chęć współpracy na WTP (kapitał społeczny) decydują o dynamicznym rozwoju i sukcesie takich rozwiązań..*

Słowa kluczowe: *Oprogramowanie open source, wirtualna taśma produkcyjna., kapitał społeczny.*

Virtual Production Line as a model to analyze the process of creating open source software

Abstract: *The main objective of this paper is to emphasize the essential role of social capital in IT companies and to demonstrate how Virtual Production Line (VPL) can be used as a model to analyze the process of creating open source software. VPL was introduced in 2006 by Walukiewicz as a model to analyze creative process and it is a natural extension of well-known Classical Production Line. Currently, open applications are built, modified, corrected and distributed owing to joint creative work of programmers. The human capital of each of these creators/programmers, their passion, programming skills and willingness to cooperate in the Virtual Production Line (social capital) decide of dynamic growth and success of such solution.*

Keywords: *open source software, virtual production line, social capital.*

1. WSTĘP

Na początku przedstawimy w zarysie genezę powstania wolnego i otwartego oprogramowania oraz pokażemy, jak kształtowała się potęga społeczności linuksowej, rozwijającej otwarte aplikacje oraz jak hackerzy z całego świata zbudowali nowy rynek – rynek oprogramowania open source.

W 1965 roku zespół naukowców z Bell Labs firmy AT&T (New Jersey, USA), Massachussets Institute of Technology i General Electric Company rozpoczął prace nad systemem operacyjnym MULTICS w ramach projektu MAC. Jego zadaniem miało być umożliwienie wielu użytkownikom równoczesnego dostępu do komputera i równoczesnego korzystania ze znacznej ilości danych. System ten nie spełniał jednak oczekiwań i założeń przyjętych przez jego twórców, dlatego też Bell Labs odstąpił od udziału w projekcie (patrz Bach, 1995). W rzeczywistości niezadowalające okazały się wyniki prac nad Multicsem, dlatego ponownie podjęto próby poprawy programowania, zgodnie z poprawionymi założeniami. I tak

w Bell Labs około 1969 roku powstała pierwsza wersja systemu operacyjnego Unix na komputery architektury PDP-7 i PDP-9. Napisana została przez Dennisa Ritchie i Kena Thompsona w asemblerze i nazwana UNICS (*Uniplexed Information and Computing System)*.
Powodzenie tego systemu operacyjnego skłoniło autorów do rozpoczęcia prac nad drugą wersją systemu na komputery PDP-11, która została uruchomiona w 1971 roku. System cieszył się dużą popularnością, zwłaszcza w instytucjach naukowych, które modyfikowały go według własnych potrzeb (patrz Silvester, 1991). W 1972 roku autorzy przepisali cały kod w wysokopoziomowym języku C, stworzonym przez Dennisa Ritchie. Decyzja ta zapewniła systemowi przenośność oraz pozwoliła na przetrwanie wszystkich przemian, które dokonały się w ciągu następnych kilkudziesięciu lat (Bach, 1995). W 1975 roku powstała szósta wersja systemu (Unix Sixth Edition), którą rozprowadzano w uczelniach dla zastosowań akademickich, a firma AT&T nie domagała się za nią żadnych opłat. Rok później John Lions, wykładowca z Uniwersytetu Nowej Południowej Walii w Australii, napisał legendarny komentarz do kodu Uniksa, tzw. Lions Book (właściwie: Source Code and Commentary on Unix Level 6), zawierający pełny kod w wersji 6.

Jedną z najważniejszych instytucji, do której trafiła nieodpłatnie kopia tego systemu był Uniwersytet Kalifornijski w Berkeley. W 1979 roku AT&T zaczęła pobierać opłaty licencyjne za kody źródłowe, co skłoniło inne firmy czy uniwersytety, do tworzenia własnych wersji systemu UNIX. To wtedy powstał system BSD (Berkeley Software Distribution) opracowany przez Uniwersytet Kalifornijski w Berkeley, SunOS firmy Sun Microsystems, Xenix Microsoftu. AT&T również wprowadziło do sprzedaży nowy system, nazwany System V, który w latach 80-tych zaczął konkurować z systemem BSD.

2. POCZĄTKI IDEI WOLNEGO DOSTĘPU DO KODU ŹRÓDŁOWEGO.

Przeciwko próbom zamykania kodu źródłowego i skomercjalizowania systemu UNIX wystąpił Richard M. Stallman z Massachussets Institute of Technology. Zaczął on rozwijać alternatywne oprogramowanie o otwartym kodzie źródłowym. W 1984 roku rozpoczął pracę nad otwartą, darmową wersją Unixa, którą nazwał GNU. Rok później Stallman założył Free Software Foundation (Fundację Wolnego Oprogramowania) promującą, chroniącą i tworzącą wolne oprogramowanie oraz rozwijającą projekt GNU. Dwa lata później ogłosił on tzw. Manifest GNU, zawierający podstawowe założenia jego projektu, w którym jasno określił na jakich zasadach takie oprogramowanie będzie udostępniane. Według Stallmana wolne oprogramowanie to takie, do którego źródeł mamy pełen dostęp i które można swobodnie używać, modyfikować i rozpowszechniać. Do lat 90-tych jedynym brakującym elementem systemu było jądro. W Manifeście GNU Stallman wspominał, że "*istnieje znajdujące się w początkowym stadium rozwoju jądro, ale jeszcze brakuje mu wiele, aby emulować Uniksa*".

Jednym z klonów systemu był także Minix, napisany w 1987 roku przez Holendra Andrew Tanenbauma, jako komercyjna wersja na platformę x86 i rozprowadzana na podstawie licencji GNU. System stał się inspiracją dla fińskiego studenta informatyki Linusa Torvaldsa (Helionica). Wykorzystał on wiele gotowych rozwiązań, używanych dotychczas w systemie Minix, do prac nad własnym jądrem nowego systemu. Torvalds zaczął pracę w Asemblerze, jednak później zdecydował się pisać jądro w języku C, modyfikując kod Minixa w oparciu o Uniksa. Pierwsza wersja 0.01 ukazała się w sierpniu 1991 roku, jednak była bardzo uboga, więc nie wzbudziła szerokiego zainteresowania. Otwartość kodu źródłowego pozwoliła użytkownikom na wprowadzanie uzupełnień i modyfikacji, dzięki czemu w szybkim czasie powstawały kolejne, coraz bardziej doskonałe wersje jądra. Torvalds dopracował system, a także upowszechnił kod źródłowy na wielu serwerach FTP, co spopularyzowało go wśród szerokiej rzeszy odbiorców. Najprawdopodobniej nazwa Linux stanowi kombinację słów „Linus" i „Unix" albo „Linus" i „Minix". Niektóre źródła wskazują, że autorzy tej nazwy użyli akronimu rekurencyjnego, aby zwrócić uwagę, że nie jest to system Unix (Linux Is Not UNIX).

Pełen system operacyjny, oprócz jądra Linux napisanego przez Torvaldsa, potrzebował jeszcze kompilatora, powłoki systemowej, bibliotek wywołań jądra, aplikacji itp. Autor postanowił wykorzystać do tego oprogramowanie i narzędzia GNU rozwijane przez Stallmana, dlatego też najbardziej popularną i poprawną nazwą dla całego systemu wraz z jądrem jest GNU/Linux. Używanie nazwy Linux do określenia kompletnego systemu operacyjnego jest nieprawidłowe, ale często używane potocznie. W tym artykule pełen system operacyjny GNU/Linux nazywany będzie w uproszczeniu Linux.
Linux rozprowadzany jest na licencji GNU GPL (General Public Licence), której celem jest przekazanie użytkownikom następujących wolności (Stallman, 2002):

- ✓ wolność uruchamiania programu w dowolnym celu (wolność 0)
- ✓ wolność analizowania, jak program działa i dostosowywania go do swoich potrzeb (wolność 1)
- ✓ wolność rozpowszechniania niezmodyfikowanej kopii programu (wolność 2)
- ✓ wolność udoskonalania programu i publicznego rozpowszechniania własnych ulepszeń, dzięki czemu może z nich skorzystać cała społeczność (wolność 3).

3. DWA MODELE ROZWOJU OPROGRAMOWANIA.

W 1997 roku Eric Raymond, jeden z czołowych ekspertów projektu GNU, zaniepokojony brakiem zainteresowania środowiska biznesowego wolnym oprogramowaniem, opublikował esej „The Cathedral and the Bazaar". Praca ta została uznana za jeden z najbardziej fundamentalnych tekstów w historii open source oraz twórcze podejście do nowego postrzegania wolnego oprogramowania.
W swojej publikacji Raymond zaprezentował środowisku hackerskiemu[16] dwa zupełnie nowatorskie spostrzeżenia dotyczące sposobu tworzenia oprogramowania. Klasyczny model powstawania oprogramowania własnościowego, prawnie zastrzeżonego (ang. proprietary software), przyrównał on do monumentalnej katedry budowanej w skupieniu i ciszy. Raymond był przekonany, że duże projekty „powinny powstawać jak katedry: budowane zręcznymi palcami samotnych czarodziei lub grupy dostojnych magów, pracujących w pełnym namaszczenia odosobnieniu, bez wersji beta udostępnianych przed czasem". Zauważył on, że społeczność linuksowa zamiast pracować w ciszy, „przypominała jeden wielki hałaśliwy bazar, pełen różnych poglądów i planów (co doskonale reprezentują serwery z archiwami linuksowymi, przyjmujące programy od każdego)" (patrz Raymond, 1999). Kod źródłowy takich programów powstaje publicznie, dzięki wspólnej pracy wielu ochotników-pasjonatów. Dopiero powodzenie rozwoju oprogramowania Linusa Torvaldsa przekonało Raymonda, że tzw. rozwiązanie bazarowe, ze względu na ogrom pracy włożonej przez dużą liczbę programistów, może przynieść wymierne korzyści zarówno twórcom, jak i użytkownikom aplikacji.

Fundamentalnym sformułowaniem definiującym modele bazarowe okazało się zdanie E. Raymonda: „given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix will be obvious to someone", (patrz, Raymond 1997), rozpowszechnione w mniej formalnej postaci jako „given enough eyeballs, all bugs are shallow". Stwierdzenie to, nazwane od nazwiska twórcy jądra Linuxa Prawem Linusa, wyjaśnia, że prawie każdy błąd (bug[17]) w działaniu oprogramowania może zostać dostatecznie szybko dostrzeżony
i naprawiony, jeśli nad projektem pracuje wystarczająca liczba testerów – użytkowników.

4. ISTOTA WIRTUALNEJ TAŚMY PRODUKCYJNEJ

Z całą pewnością można stwierdzić, że aplikacje open source są obecnie najprężniej rozwijającym się oprogramowaniem na świecie. Rynek oprogramowania jest bardzo dynamiczny i podlega ciągłej ewolucji.

Wolne i otwarte oprogramowanie powstaje jako owoc pracy wielu pasjonatów, pracujących jednocześnie, w sposób równoległy. Otwartość kodu źródłowego takich aplikacji powoduje,
że programiści z całego świata angażują się w jego tworzenie, modyfikowanie, udoskonalanie i dystrybuowanie. Można powiedzieć, że takie oprogramowanie jest efektem udostępniania twórczości informatycznej programistów dla szerokiej społeczności (na zasadzie opisanego wcześniej bazaru). Zatem w tym kontekście tworzenie programów o otwartych źródłach, w tym zwłaszcza systemu operacyjnego Linux, jest procesem twórczym.

Programiści rozwijający oprogramowanie open source to ludzie kreatywni, którzy dzielą się swoją wiedzą, dokonują konstruktywnej krytyki pracy innych profesjonalistów poprzez poprawianie i udoskonalanie takich aplikacji.

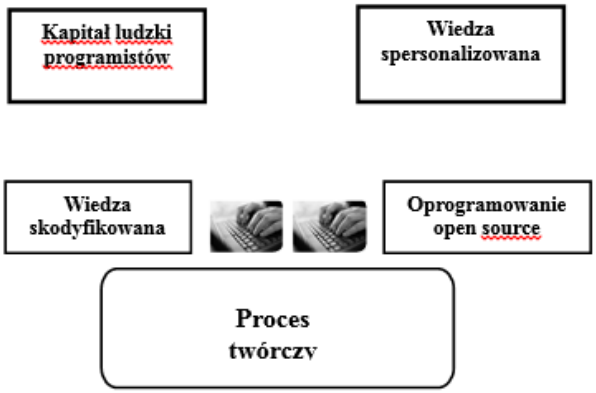
---

16 Hacker – w terminologii komputerowej: osoba posiadająca doskonałą znajomością języków programowania, systemów operacyjnych, w tym zwłaszcza systemów uniksopodobnych

17 W żargonie informatycznym: bug = błąd oprogramowania wynikający z pomyłki podczas programowania

Kapitał ludzki każdego z nich, tj. ich wiedza informatyczna, talent, umiejętności programowania, doświadczenie tworzą łącznie kapitał społeczny całej tej społeczności. Ich wspólna praca, pomysły, relacje nieformalne między nimi powodują, że powstają produkty bardziej doskonałe, bezpieczne mniej awaryjne niż te znajdujące się w ofercie gigantów branży informatycznej (np. system operacyjny Windows rozwijany przez firmę Microsoft). Co ważne, nagrodą dla tych pasjonatów – ochotników nie jest zazwyczaj wynagrodzenie, czy kontrakt, ale produkt doskonały.

Ideą rozwijania otwartych i wolnych aplikacji jest wspólna, kreatywna praca zespołu ekspertów, komunikujących się ze sobą za pomocą Internetu, zatem proces taki jest procesem twórczym, gdyż takie aplikacje są unikalne, tworzone po raz pierwszy.



Rysunek 1. WTP w analizie procesu tworzenia oprogramowania open source (źródło: Walukiewicz S.,)

Rozważmy wirtualną taśmę produkcyjną przedstawioną na rysunku 1. Programiści pracują twórczo, rozwiązują zauważone problemy w działaniu danego oprogramowania, uzupełniają, poprawiają i wprowadzają udoskonalenia w kodzie źródłowym, pracując na odpowiednio skonstruowanej wirtualnej taśmie produkcyjnej. W tym miejscu należy podkreślić, że w rzeczywistości taka taśma nie istnieje, dlatego też nazwana została wirtualną oraz oznaczana jest na rysunku linią przerywaną. Programiści pracując na takiej wirtualnej taśmie produkcyjnej wykorzystują swoją wiedzę skodyfikowaną (uzyskaną z książek, dokumentów itp.) oraz wiedzę spersonalizowaną, tj. doświadczenie, talent, pomysłowość, itp., (patrz Walukiewicz, 2010).

Proces tworzenia takiego oprogramowania podzielony jest na zadania realizowane przez poszczególnych programistów. Mogą oni dowolnie organizować sobie pracę, np. zmieniać kolejność wykonywanych zadań lub ich liczbę. Zmiany takiego procesu twórczego, rozwiązywanego na danej wirtualnej taśmie produkcyjnej, jego przeformułowanie dokonywane przez programistów podczas ich pracy nazywać będziemy samoorganizacją (patrz Walukiewicz, 2012).

Powyższe rozważania są podstawą do podania za Walukiewiczem definicji wirtualnej taśmy produkcyjnej:
Wirtualna taśma produkcyjna (WTP) to podział procesu twórczego (projektu) na zadania połączony z teleinformatyką. Zarówno sam podział, jak i liczba zadań mogą być zmieniane przez ekspertów, pracujących na danej WTP. Nazywamy to jej samoorganizacją. Może być ona wielokrotnie powtarzana.
Każda WTP składa się zatem z trzech elementów: podziału pracy twórczej na zadania, teleinformatyki (Internet) oraz samoorganizacji.

5. ZASTOSOWANIE WTP W ANALIZIE OPROGRAMOWANIA OPEN SOURCE.

Wirtualna taśma produkcyjna jest odpowiednim modelem do analizy pracy twórczej społeczności programistów rozwijających oprogramowanie open source. Ochotnicy budujący otwarte systemy podejmują się zadań, które potrafią i chcą wykonywać, a nie takich, do których zostają przymuszeni. To niezwykle istotna kwestia, gdyż to właśnie ich pasja, zaangażowanie i zapał decydują w dużej mierze o sukcesie takich aplikacji. W tym miejscu warto także podkreślić, jak ważną rolę pełni tu Internet. To możliwość sprawnej komunikacji i szybkiej wymiany pomysłów pomiędzy programistami ma istotny wpływ na postęp w rozwoju takich projektów. Wpisuje się to doskonale w założenie jednego z tzw. Praw Lehmana[18], mówiących, że aby program komputerowy stosowany w rzeczywistym środowisku był użyteczny, musi być stale modyfikowany i udoskonalany.

[18] Meir Lehman – brytyjski profesor, twórca teorii i praw ewolucji oprogramowania

Najszybciej rozwijającym się projektem open source jest jądro (*ang*. kernel) systemu operacyjnego Linux, napisane w 1991 roku przez Linusa Torvaldsa. Jest ono najważniejszą częścią uniksopodobnych systemów operacyjnych, rozwijaną przez programistów - wolontariuszy z całego świata w ramach The Linux Foundation. Pierwsze wersje jądra rozwijane były przez kilkuset koderów, a najnowsze (tzw. v. 3.2) – przez ponad 1300. Wspólnie napisano już ponad 15 mln linii kodu. Jednak od imponującej liczby linii kodu źródłowego ważniejsze jest to, jak szybko powstają kolejne wersje jądra systemu Linux i jak szybko są rozwijane i udostępniane. Jądro powstaje w oparciu o model *release – time*, co w praktyce oznacza, że jego nowa wersja pojawia się co 2-3 miesiące. Pomimo znaczącej liczby programistów – ochotników rozwijających jądro Linux, wciąż jest stosunkowo niewielu programistów, których praca w sposób istotny przyczynia się do jego udoskonalania. Przez ostatnie 5 lat twórcza praca dziesięciu czołowych ekspertów doprowadziła do 9% wszystkich zmian. Wśród nich jest Polak - Bartłomiej Żołnierkiewicz, który do tej pory wprowadził 2 074 zmiany w kodzie źródłowym systemu, co stanowi 0,8% wszystkich zmian dokonanych przez całą społeczność linuksową (patrz Corbet i in., 2012).

Z danych udostępnionych przez The Linux Foundation wynika, że prawie 18% zmian kodzie źródłowym Linuxa dokonywanych jest przez pasjonatów, nie zatrudnionych na co dzień w sektorze IT. W tym miejscu warto zaznaczyć, że wciąż rośnie liczba przedsiębiorstw informatycznych, których programiści pracują nad rozwijaniem jądra systemu Linux. Zatem znaczna część z nich, pracując na wirtualnej taśmie produkcyjnej, otrzymuje za to wynagrodzenie. Oczywiste jest przy tym, że firmą wnoszącą największą liczbę zmian do źródeł jądra Linuxa jest Red Hat – światowy lider w dostarczaniu rozwiązań open source do biznesu. Do tej pory pracownicy tej firmy wprowadzili 31 261 poprawek do kodu tego systemu, co stanowi 11,9% wszystkich dokonanych modyfikacji (patrz Corbet i in., 2012).
W tabeli 1 zebrano dane dotyczące liczby zmian w kodzie źródłowym systemu Linux, wprowadzonych przez programistów kilku przedsiębiorstw. Z uwagi na to, że zainteresowaniem naukowym autorki jest systemowa analiza największych przedsiębiorstw działających w branży IT, wybrano i przeanalizowano informacje odnoszące się tylko do kilku, najciekawszych podmiotów z punktu widzenia autorki.

Firma	Liczba wprowadzonych poprawek w kodzie	Procent wszystkich poprawek
Red Hat	31 261	11,9%
Novell	16 738	6,4%
Intel	16 219	6,2%
IBM	16 073	6,1%
Oracle	5 542	2,1%
Nokia	3 272	1,2%

Tabela 1. Liczba wprowadzonych poprawek do kodu źródłowego Linuxa przez programistów wybranych firm IT
Źródło: Corbet J., Kroah-Hartman G., McPherson A. (2012), *Linux Kernel Development: How Fast it is Going, Who is Doing It, What They are Doing, and Who is Sponsoring It*, The Linux Foundation

Dobrym przykładem ilustrującym proces rozwijania otwartego projektu na wirtualnej taśmie produkcyjnej jest także Apache – najczęściej wykorzystywany serwer http, dostępny dla wielu systemów operacyjnych. Zespół koderów pracujących nad poszczególnymi modułami był początkowo niewielki, jednak w miarę rozwoju projektu rozrósł się. Do prowadzenia projektu wykorzystywane są listy mailingowe i grupy dyskusyjne. Każdy moduł systemu nadzorowany jest przez tzw. opiekuna, którzy do prac nad daną częścią systemu dobierani są zgodnie z ich umiejętnościami. Jeżeli dany problem można rozwiązać na kilka sposobów, alternatywy takie rozsyłane są drogą internetową do wszystkich osób z listy mailingowej z prośbą o wsparcie. Na podstawie otrzymanych recenzji główny programista opracowuje ostateczne rozwiązanie, a następnie udostępnia i rozpowszechnia gotowy projekt. I jest to przykład samoorganizacji WTP.
Nie bez znaczenia pozostaje również fakt, że twórcy otwartego oprogramowania są zazwyczaj też jego użytkownikami (patrz Mockus, 2005). Dzięki temu doświadczają osobiście każdego błędu w działaniu takich aplikacji, co determinuje ich do rzetelnego modyfikowania, poprawiania i przez to udoskonalania działania tychże programów.

6. PODSUMOWANIE

Rozwiązania open source zrewolucjonizowały zarówno rynek oprogramowania, jak i cały przemysł informatyczny, stały się źródłem nowych rynków i wymusiły pojawienie się na nich innowacyjnych narzędzi. Ewolucja rynku technologii informacyjnych, jaka dokonała się w ciągu ostatnich kilkunastu lat, nie miała by szans na powodzenie, gdyby nie wspólny trud i zapał pasjonatów, wspólnie rozwijających otwarte projekty.

Tradycyjny model tworzenia oprogramowania komercyjnego, własnościowego zabrania użytkownikom na swobodne jego wykorzystywanie i rozpowszechnianie. Model otwarty jest bardziej współczesny, demokratyczny, a tym samym dostosowany do reguł rządzących na wolnym rynku.

W tym miejscu warto jeszcze raz przypomnieć i podkreślić istotę fundamentalnego sformułowania definiującego model tzw. bazarowego rozwijania oprogramowania open source. Jeżeli nad projektem pracuje twórczo duża liczba programistów, pasjonatów, wszystkie problemy w działaniu programu i błędy w kodzie źródłowym mogą być w dostatecznie szybki sposób rozwiązane. Producenci oprogramowania własnościowego, zamkniętego, na etapie programowania, nie są w stanie przewidzieć ile i jakie błędy proponowane przez nich aplikacje posiadają. Taka ograniczona liczba inżynierów oprogramowania i jego testerów z pewnością nie wnosi w rozwój projektów tak wielkiego kapitału, jak międzynarodowa społeczność pasjonatów rozwijająca oprogramowanie open source.

Ze statystyk opublikowanych w marcu 2012 roku przez The Linux Foundation wynika, że firma Microsoft po raz pierwszy znalazła się w pierwszej dwudziestce firm najaktywniej uczestniczących w rozwoju jądra systemu Linux. Programiści zatrudnieni w firmie kierowanej przez Steve'a Ballmera wprowadzili 1,2 % wszystkich zmian w kodzie jądra Linuksa (v. 2.6.36 oraz 3.2). Co ciekawe to właśnie pracownik firmy Microsoft - K. Y. Srinivasan, jako indywidualny ekspert, wprowadził największą liczbę poprawek do kodu źródłowego systemu w wersji 3.0 (patrz Corbet i in., 2012).

Jednak nawet najwięksi gracze rynku informatycznego, w tym m. in. Microsoft, nie mogą sobie pozwolić na zatrudnienie tak dużej liczby programistów, jaka wspólnie pracując na WTP, tworzy opisany wyżej kapitał społeczny o wielkiej wartości.

Literatura

1. Bach M.J., „Budowa systemu operacyjnego UNIX", Wydawnictwo Naukowo – Techniczne Warszawa, 13-14, 1995.
2. Corbet J., Kroah-Hartman G., McPherson A., "Linux Kernel Development: How Fast it is Going, Who is Doing It, What They are Doing, and Who is Sponsoring It", The Linux Foundation, 2012.
3. Mockus A., Fielding R.T., Herbsleb J., Two Case Studies of Open Source Software Development: Apache and Mozilla, [w:] Feller J., Fitzgerald B., Hissam S.A., Lakhani K.R. (red.), Perspectives on Free and Open Source Software, The MIT Press, Cambridge, MA, 157 –175, 2005.
4. Raymond E., Release early. Release often. And listen to your customers, 1997.
5. Raymond E., The Cathedral and the Bazaar; tłum.: Artur Skura, 2001.
6. Silvester Peter P., „System operacyjny Unix", Wydawnictwo Naukowo – Techniczne Warszawa, 15, 1991.
7. Stallman R., Free Software Definition, [w:] J. Gay (red.), Free Software, Free Society, wstep L. Lessig, GNU Press, Boston, MA, 2002.
8. Walukiewicz S., „Kapitał ludzki", Instytut Badań Systemowych IBS PAN, Warszawa, 2010.
9. Walukiewicz S.. „Kapitał społeczny", Instytut Badań Systemowych IBS PAN, Warszawa, 2012.