Adam Pietrzykowski

Open source jako kult(ura) paratekstu

Media, Kultura, Społeczeństwo nr 1 (6), 17-25

2011

ADAM PIETRZYKOWSKI
Uniwersytet im. Adama Mickiewicza w Poznaniu
apiet@amu.edu.pl

OPEN SOURCE JAKO KULT(URA) PARATEKSTU

Przemierzając globalne przestrzenie Internetu, łatwo natknąć się na obszerne pokłady twórczości pochodnej. Nieprzerwanie rosnące stosy owej twórczości nie funkcjonują bowiem w izolacji. Ujmując przestrzeń Internetu za pomocą kategorii wypracowanych w obrębie analizy literackiej, można stwierdzić, że ukazuje się ona jako nieprzebrane morze intertekstualnych odniesień do obecnych w całym medialnym uniwersum tekstów. Rodzaj odniesienia pomiędzy tekstami, który będzie mnie szczególnie interesował, to paratekst.

Zgodnie z terminologią jednego z twórców teorii tekstu, Gérarda Gennette'a, paratekst oznacza wszystko to, co otacza tekst właściwy i umożliwia mu stanie się książką (Genette 1991: 261). Będzie to każdy dodatek do tekstu właściwego, jak stopka redakcyjna, wstęp czy przedmowa, a także tytuł dzieła i kompozycja okładki. W przypadku zastosowania pojęcia paratekstu do aktywności „usieciowionych" jednostek przeniesionych w obszar interaktywnego ekranu, pojęcie to wymagałoby przedefiniowania do szerokiego kulturologicznego znaczenia. I tak paratekst w obrębie nowych mediów określić można jako: rekomendująco-prezentujące preteksty do innych tekstów (Loewe, 2004: 183), zależne od tekstu postfabrykaty kultury (Krzysztofek, 2010: 13) czy ramę dyskursywną, która określa horyzont funkcjonowania tekstu bazowego (Gwóźdź, 2010: 36). W pierwszym przypadku paratekst będzie wszystkim, co zapowiada inny tekst, np. strona internetowa z recenzjami literatury czy filmu, komentarze do owych recenzji, ciekawostki, wywiady. Z kolei ujęcie paratekstu jako postfabryktu akcentować będzie zależność nowych wytworów kultury od swoich poprzedników, którzy posłużyli za swego rodzaju budulec. Te dwa ujęcia paratekstu jako pre-tekstu i postfabrykatu cechuje różne temporalne usytuowanie podmiotu w stosunku do tekstu – przed lub po zapoznaniu się z nim. Perspektywa ramy dyskursywnej zorientowana jest bardziej horyzontalnie. Określa wiązkę praktyk dyskursywnych, które poszerzają bazę medialną (tekstową) paratekstu, udostępniając go na różne sposoby i różnymi drogami (Gwóźdź, 2010: 36).

W każdym z powyższych przypadków przedrostek para- odczytany dosłownie jako słowo języka polskiego ma tę dodatkową zaletę, iż stanowi doskonałą analogię obrazującą funkcjonowanie paratekstu w kulturze. Wskazuję na efemeryczność i uzależnienie tekstu od pewnej materii wyjściowej. Im „gorętszy" jest tekst, a zatem im bardziej kontrowersyjny, doniosły, odkrywczy, zabawny, użyteczny, tym samym większe jest jego „parowanie", czyli kwantytatywny wymiar paratekstów okalających go. Powyższe ujęcia paratekstu w nowych mediach nie wyczerpują oczywiście wszystkich możliwych sposobów jego ujmowania.

W niniejszym artykule spróbuję odnaleźć relacje paratekstualności w miejscu dotychczas nieopisywanym, stawiając tezę, iż będąca wierzchnią tkanką informatycznej infrastruktury warstwa oprogramowania podlega podobnym procesom intertekstualnych odniesień. Wymagać to będzie ukazania kodu źródłowego programu komputerowego jako swoistego tekstu. Zadanie to będzie o tyle łatwiejsze, iż perspektywa teoretyczna wnikająca w odmęty technologicznej nieświadomości nowych mediów ma już za sobą swoje pierwsze artykulacje. Namysł prowadzony w jej horyzoncie myślowym dotyczyć będzie zwłaszcza otwartego oprogramowania oraz społeczności zgromadzonej wokół niego.

Kod, który stał się kulturą

Za prekursora humanistycznej refleksji nad oprogramowaniem uznać należy Lwa Manovitcha. W *Języku nowych mediów*, swojej najbardziej rozpoznawalnej pracy, stwierdza on, iż:

> By zrozumieć logikę nowych mediów, musimy zwrócić się do nauk informatycznych. Właśnie tam możemy oczekiwać odnalezienia nowych pojęć, kategorii i operacji, które charakteryzują media mogące zostać zaprogramowane. Od studiów nad mediami, przechodzimy do czegoś, co można by nazwać '*software studies*' – od teorii mediów do teorii oprogramowania (Manovitch, 2006: 117).

To kluczowy fragment, który można uznać za quasi-manifest nowego spojrzenia na medialność w epoce cyfryzacji, spojrzenia oddolnego zapatrzonego w miękką materię oprogramowania (*software*) w odróżnieniu od jej twardego podłoża (*hardware*), czyli elektroniki oraz fizycznych łącz, w których przemieszcza się sygnał[1].

Oprogramowanie stanowi właściwe medium, które odpowiedzialne jest za kształt obiektów nowych mediów. Mówiąc językiem informatyki „formatuje" artefakty kultury niematerialnej, nadając im specyficzny dla środowiska elektronicznego sposób bycia. Modułowość, wariacyjność oraz transkodowanie to podstawowe kategorie jakie wyróżnia Manovitch ukazujące *status quo* „kawałka kultury", który przyobleka się w cyfrową szatę. Zwłaszcza ostatnia kategoria nazwana dokładnie transkodowaniem

[1] Niektórzy teoretycy mediów, jak Friedrich Kittler, nie podzielają takiego ujęcia technologii informacyjnych uważając, że ontologiczny status oprogramowania jako immaterialnego i przeciwstawnego do elektroniki bytu nie ma uzasadnienia w faktycznym stanie rzeczy. Jest błędem czynionym przez informatycznych ignorantów. Zobacz: Kittler F. (1995), *There is No Software*, Ctheory, www.ctheory.net/articles.aspx?id=74.

kulturowym wskazuje na przenikanie się warstw komputerowej i kulturowej. To proces, który wpływa w istotny sposób na możliwości kreacji, modyfikacji i dystrybucji treści kultury, jak również na możliwość ich zapamiętywania. Zrozumienie tego oddziaływania, zwłaszcza zaś w odniesieniu do kultury audiowizualnej, wymaga krytycznej refleksji, swoistej „krytyki czystego oprogramowania".

Oprogramowanie jest dla Manovitcha na tyle istotnym składnikiem dzisiejszych społeczeństw, iż stwierdza on:

> (...) współczesne społeczeństwo można scharakteryzować jako społeczeństwo oprogramowania (*software society*), współczesną kulturę natomiast można zasadnie nazwać kulturą oprogramowania (*software culture*), gdyż oprogramowanie odgrywa dziś kluczową rolę zarówno w kształtowaniu rzeczywistości materialnej jak i wielu niematerialnych obiektów, które razem tworzą kulturę (Manovitch, 2008: 17).

Choć refleksja Manovitcha wyznaczyła odważny i ryzykowny dla humanistyki kierunek myślenia, nie penetruje bezpośrednio tego, co znajduje się pod powierzchnią ekranu – kodu, *logosu* komputera. Dokładnie rzecz biorąc kod źródłowy to zbiór instrukcji zapisany w pliku, w którym poprzez konkretny język programowania wyraża się działanie programu – interpretowanego (przetwarzanego za każdym razem podczas uruchamiania na język maszynowy zrozumiały dla komputera) bądź kompilowanego (przetworzonego jednorazowo) w celu użycia. Umieszczenie kodu w ognisku refleksji humanistycznej zakrawa na kuriozum bowiem intencjonalność w kodzie źródłowym programu jest jawna i jednoznaczna, a jej wymiar jest czysto performatywny. Jej odczytanie zaś sprowadza się do biegłości w języku programowania, w którym program jest napisany. Wąski, matematyczno-logiczny zakres środków wyrazu oraz brak niedookreśloności semantycznej kodu, która uprawniałaby do postawienia pytania: „o co autorowi chodzi?", sprawia, że wyglądać on może na mało atrakcyjny obiekt poznania. Sprawę zmienia nieco spojrzenie na kod w kontekście jego powstawania.

W kierunku problematyzacji kodu w kontekście społecznej praktyki jego wytwarzania wiedzie myśl brytyjskiego badacza kultury Adriana Mackenziego. W jego ujęciu kod źródłowy jest niezwykle istotnym obiektem dla zrozumieniu nowych mediów:

> (...) *software studies* nie powinno bezpośrednio przenosić terminów, kategorii i operacji z informatyki. Pomimo jego formuły jako kontrolowanego przez reguły wyrażenia, kod (...) jest przejściowym, zmiennym tworzywem. (...) co robi oprogramowanie i jak jest wykonywane, rozprowadzane, zmieniane i utrwalane nie może być zrozumiałe bez ujęcia go w całości jako kodu (Mackenzie, 2006: 6).

Kod pojmuje on nie jako raz dany i gotowy artefakt techniczny, ale jako dynamicznie zmieniającą się rzeczywistość osadzoną w społecznym procesie. Wynika to główne z praktyki programistycznej, która zakłada równorzędne istnienie wielu wersji oprogramowania, wielu osobnych modułów w jego obrębie opracowywanych przez różne osoby. Interesuje go zwłaszcza praktyka programistów tworzących w modelu produkcji partnerskiej *open source*. *Open source* ('otwarte źródło') to określenie zarówno modelu produkcji oprogramowania opartego na wolontaryjnej pracy zainteresowanych programistów jak i programu komputerowego, którego kod możliwy jest do wglądu,

modyfikowania, dystrybuowania i uruchamiania. Dla Mackenziego sprowadzanie *software studies* do analizy elementu użytkowego jakim jest interfejs użytkownika oraz operacje w nim dostępne, które kreują coraz bardziej rzeczywistość kultury, nie ujmuje całej złożoności zdywersyfikowanej przestrzeni jaką jest oprogramowanie. Konieczne jest ukazanie kontekstu powstawania, ramy społeczno-kulturowej, która determinowała proces jego tworzenia, a która pozostaje z nim w stałym sprzężeniu zwrotnym.

Dzięki temu możliwe staje się uchwycenie motywów, jakie przyświecały tysiącom użytkowników Sieci biorących udział w przedsięwzięciu tworzenia przeglądarki internetowej Mozilla Firefox, pakietu biurowego OpenOffice, serwera HTTP Apache czy systemu operacyjnego GNU\Linux.

W przypadku społeczności *open source* sama jawność kodu źródłowego stanowi przesłankę o konstytutywnej dla niego wartości. Perspektywa Mackenziego skupia się zatem na analizie kodu jako dynamicznej rzeczywistości, która stanowi pole do realizacji określonych wartości przez cyberspołeczność. Choć ten socjologiczny punkt widzenia ukazuje niesłychanie istotny kontekst powstawania, do pełnego obrazu oprogramowania potrzebne będzie spojrzenie hermeneutyczne, które pozwoli nam wreszcie uchwycić oprogramowanie w sposób właściwy dla postawionej na wstępie tezy.

Kod źródłowy oprogramowania jako swoisty tekst *sensu stricto* sproblematyzował dopiero amerykański badacz Mark Marino inspirowany, jak przyznaje, esejami Matthew Fullera *Behind the Blip: essays on the culture of software.* W swoich esejach Fuller stawia pytanie o możliwą teorię krytyczną dotyczącą oprogramowania. Uwzględniając wcześniejszą refleksję na tym polu, stara się „uczynić widzialną dynamikę, struktury, tryby działania i energię, która stoi za każdym zdarzeniem, z którym się ono – oprogramowanie – wiąże" (Fuller, 2003: 32). Marino podejmuje tak określony problem, lecz rozwija go w innym kierunku. Obszar i metodę badawczą określa jako: „(...) podejście, które stosuje krytyczną hermeneutykę do interpretacji kodu komputerowego, architektury programu i dokumentacji w społeczno-historycznym kontekście" (Marino, 2006: 5). Jego starania, by podjąć próbę krytycznego odczytywania owego kodu za pomocą narzędzi humanistyki (ideologia, władza, *gender* itd.) podchwyciło kilkudziesięciu badaczy pogranicza humanistyki i nauk o informacji tworząc *critical code studies* – krytyczne studia nad kodem. Od badania wiarygodności kodu sond internetowych, poprzez analizę komentarzy w kodzie, czy sposób modelowania podmiotowości użytkownika badacze ci starają się (...) lepiej rozumieć programy i sieci różnych programów i ludzi, na których one oddziałują, podporządkowując, reprezentując, manipulując, transformując innymi słowy mówiąc, angażując" (Marino, 2006: 8).

Kod może komunikować coś, czego w interfejsie dostrzec nie sposób. Prowadzi to do pytania, dlaczego kod wygląda właśnie tak, jak wygląda? Z jednej strony kod zdradza umysłowość programisty, która za nim stoi, ukazując jeden z możliwych sposobów implementacji algorytmu. Z drugiej, ujawnia funkcje programu, które są niedostrzegalne na poziomie interfejsu użytkownika, a których istnienie umotywowane jest jakimś celem (często niegodziwym jak próbą manipulacji danymi, nieuprawnionym

uzyskaniem dostępu). Przedstawia zatem komponent intencjonalny niedostrzegalny w obcowaniu z gotowym programem. Niemniej performatywna funkcja kodu, która realizuje się podczas wykonania programu, ukazuje jego właściwą istotę, gdyż języki programowania, w których kod jest wyrażony, faktycznie robią to, co mówią (Gallaway, 2006: 31).

Odwołując się do teorii aktów mowy i pojęcia illokucji, wykonany kod stwarza realne skutki w rzeczywistości, przez co staje się równie ważny jak język naturalny (Hayles, 2005: 49). Definiując zatem kod źródłowy jako tekst, można stwierdzić, że jest to tekst *stricte* performatywny, który „użyty" działa w świecie. Działa na różne sposoby. Zamiast mówić o prawie, jest prawem, zakreślając możliwości użytkowników i reżim postępowania z informacją (Lessing, 2006). Zamiast komunikować określone intencje, jest ich realizacją. Oczywiście zakres jego oddziaływania jest z reguły dość ograniczony. Zważywszy jednak na postępujące „oprogramowywanie" wszelkich obszarów działalności ludzkiej, zakres ten ciągle się poszerza, skłaniając coraz częściej do pytania, czy zaufanie jakim darzymy „programy życia codziennego" jest uzasadnione?

Parateksty oprogramowania

Uznawszy kod źródłowy programu za swoisty tekst, którego specyfiką jest performatywność, ukazane zostaną obecnie jego przykłady w świecie oprogramowania. Paratekst będzie tu rozumiany na sposób bliski zaproponowanemu przez Andrzeja Gwoździa:

> jako wiązki praktyk dyskursywnych, dzięki którym bazowy tekst (audio)wizualny lub samo medium, w obrębie którego on funkcjonuje, włączone zostają do środowiska szerszego niż to, które zapewnia paratekstowi jego baza tekstowa (medialna) (Gwóźdź, 2010: 36).

Oczywiście tekst oznaczał tu będzie kod. Z racji podstawowego dualizmu tkwiącego w oprogramowaniu relację paratekstualności odnaleźć można na dwóch poziomach.

Pierwszy, wewnętrzny rodzaj paratekstualności określić można jako autoreferencyjny. Taką rolę spełniać będą komentarze obecne w kodzie źródłowym programu. Komentarze do kodu pozwalają programistom wyrażać pewne intencje odnośnie kodu: stworzyć „logiczną mapę" aplikacji opisującą działanie wybranych fragmentów kodu, ale także dokonywać estetycznych sądów czy dodawać treści zupełnie niezwiązane z działaniem programu, jak opis subiektywnych stanów doświadczanych przez autora programu podczas jego tworzenia. Komentarze nie stanowią części wykonywalnej kodu, są więc transparentne zarówno dla maszyny, na której program jest uruchamiany jak i jego użytkownika. Paratekstualny charakter komentarzy w kodzie wskazał Jerremy Douglas, współtwórca *critical code studies* oraz współpracownik Manovitcha przy *software studies initiative*. Śledząc literaturę nauk o informacji, dostrzegł, że istnienie komentarzy w kodzie jest sprawą dyskusyjną i poza ujmowaniem ich jako korzystnych wtrętów ułatwiających nawigowanie po materii kodu, są one również odbierane jako swego rodzaju niepowodzenie programisty, który musiał wprowadzić

metajęzyk do opisu niezbyt precyzyjnie napisanego programu (Douglas, 2010). Dobrą analogią są tu przypisy w przekładach, które stanowią egzemplifikację paratekstu w literaturze. W recenzji do niedawno wydanej pracy dotyczącej zagadnienia przypisów prof. dr hab. Teresa Giermak-Zielińska zapytuje:

> Czy przypis, zwłaszcza w przekładach literackich, jest świadectwem niedostatecznej biegłości tłumacza, czy też, przeciwnie, jego skrupulatności i troski o jasny przekaz przełożonego na inny język dzieła? (Giermak-Zielińska, 2010).

To pokrewieństwo problemowe ze światem literatury potwierdza paratekstualny charakter komentarzy w kodzie oraz stwarza przestrzeń dialogu między tymi dwiema dyscyplinami.

Drugi, zewnętrzny rodzaj paratekstu sprowadzić można do relacji pomiędzy kodem źródłowym programu bazowego (tekstu) a innymi, wybranymi programami, które zostają niejako nadbudowane poszerzając przestrzeń jego „informatycznego oddziaływania". Z punktu widzenia tego artykułu interesować mnie będzie właśnie ten rodzaj paratekstualności, który – jak uważam – ma bezpośrednie przełożenie na kształt środowiska nowych mediów i tym samym na stan kultury zanurzonej w oprogramowaniu.

Za najbardziej znaną egzemplifikację paratekstu w świecie oprogramowania uznać można system operacyjny GNU/Linux[2]. Od czasu swojego powstania w 1991 roku stał się flagowym przykładem rewolucji w sposobie produkcji dóbr niematerialnych (Reymond, 1997; Benkler, 2008), argumentem na rzecz tezy o rewitalizacji ekonomii daru w społeczeństwie informacyjnym (Castells, 2003; Zawojski, 2006) czy dowodem na sukces fińskiego modelu polityki społecznej (Castels, Himanen, 2009). Wedle wcześniejszego stwierdzania Linux to system operacyjny, a więc zbiór programów, których przeznaczeniem jest stworzenie środowiska pracy dla użytkownika oraz platformy komunikującej uruchamiane przez niego programy z zasobami komputera. Takie definiowanie Linuksa jest jednak niewystarczające dla zrozumienia złożoności jego paratekstualnej natury. Trzymając się definicyjnej precyzji, to, co rozumiemy pod pojęciem Linux, to w istocie program komputerowy działający na najniższym poziomie – tzw. jądro systemu (*kernel*) – którego zadaniem jest wspomniana obsługa żądań programów wobec zasobów komputera. Zatem właściwa definicja Linuksa brzmieć może następująco: to system operacyjny bazujący na jądrze Linuksa, wyróżniający się specyficznym doborem oprogramowania instalującego, zarządzającego i użytkowego, do którego dodana została pewna wartość symboliczna. Całość określana mianem dystrybucji. Dokonać trzeba zatem podstawowego rozróżnienia na *kernel* (jądro) i dystrybucję (partykularną kompozycję). Relacja między jądrem, a dystrybucją jest relacją zawierania się. Każdy system określony jako Linux musi zawierać w sobie jądro Linuksa. Dystrybucja zaś jest każdorazowo nowym kolażem aplikacji, konfiguracji i estetyki interfejsu użytkownika. To właśnie w niej upatrywać można informatycznej praktyki paratekstualnej. Jądro systemu ujęte jako tekst nie może funkcjonować w izolacji. Tak

[2] Choć potocznie używa się określenia Linux, dokładna nazwa uwzględniająca pochodzenie najistotniejszych aplikacji składających się na system brzmi GNU/Linux. W dalszej części pracy używał będę jednak określenia potocznego.

jak tekst literacki umieszczony zostaje w strukturze nazwanej książką, która posiada okładkę, tytuł, tekst wprowadzającego itd., tak jądro Linuksa występuje w paratekstualnym środowisku odpowiednich aplikacji generujących interfejs użytkownika oraz zarządzających zasobami komputera. Paratekstem będzie tu zatem wszystko, co okalając jądro, pozwala mu stać się systemem operacyjnym, który z kolei istnieje zawsze jako dialogiczna relacja oprogramowania wewnątrz niego.

Łatwość paratekstualnej praktyki doboru oprogramowania sprawiła, że istnieje dziś kilkaset dystrybucji systemu Linux. Tych liczących się z pewnym wkładem własnym w produkcję „oprogramowania paratekstualnego" jest zaledwie kilka. Niemniej każda dystrybucja Linuksa stanowi odmienny paratekst, dla którego materią (tekstem) wyjściowym jest jądro systemu. System Linux i jego dystrybucje nie wyczerpują oczywiście całej gamy paratekstów w świecie oprogramowania. Wszelkie pluginy (wtyczki), mody, wykorzystywanie dostępnych API, a więc praktyki, które udostępniają tekst poprzez stworzenie nowego kontekstu jego funkcjonowania, można również określić mianem pratekstu w zastosowanym dyskursywnym rozumieniu.

Cyberkomunizm czy kult(ura) paratekstu?

W świecie oprogramowania kod źródłowy stanowi nieocenioną wartość. Jego posiadanie umożliwia wgląd w zastosowane rozwiązania algorytmiczne, pozwalając tym samym kontrolować wizualno-funkcjonalny kształt aplikacji. Dwie najczęściej spotykane strategie twórców oprogramowania określone będą stosunkiem do jawności kodu. Pierwsza, na której Bill Gates zbudował swoją fortunę, to restrykcyjny model dostępu, w którym nadaje się użytkownikowi wyłącznie prawo do uruchamiania programu, pozbawiając go możliwości wglądu „pod powierzchnię" ekranu. Drugą, przeciwstawną strategią jest wywodzący się z kultury hakerskiej idealistyczny model wolnego oprogramowania, który zapewnia użytkownikowi niemalże każdą możliwą informatyczną wolność. Z zainicjowanego przez Richarda Stallmana w pierwszej połowie lat 80. ruchu wolnego oprogramowania dekadę później wyłonił się ruch otwartych źródeł. To właśnie ten nastawiony utylitarnie obóz nazywany *open source* odniósł niebywały sukces w tworzeniu oprogramowania, które funkcjonuje w innym niż merkantylny porządku aksjologicznym. Sukces zdeterminowany głównie synergią dwóch czynników: zastosowaniem otwartego modelu partycypacji, który umożliwiał przyłączenie się do projektu każdej zainteresowanej osobie oraz upowszechnianiem się w tym czasie nowego ładu komunikacyjnego za sprawą sieci Internet.

O ile parateksty w świecie oprogramowania powstawać będą zarówno w obszarze oprogramowania komercyjnego, jak i wolnego/otwartego, to w przypadku tego pierwszego natrafiamy na podstawową trudność. Z przyczyn osiągnięcia przewagi konkurencyjnej, różnego rodzaju patentów czy zagadnień bezpieczeństwa, twórcy oprogramowania komercyjnego z reguły pozbawiają użytkownika jakichkolwiek praw do kodu źródłowego. Możliwości paratekstualnej praktyki ograniczać się będą zatem do wariacyjności w obrębie określonej „funkcji prosumenckiej" programu, jak np. tworzenia skórek, modów czy pluginów.

Wolne i otwarte oprogramowanie stanowi furtkę dla twórczych praktyk mash-upu czy remake'u kodu źródłowego, ale także tworzenia aplikacji zewnętrznych, wykorzystujących bądź odwołujących się do wewnętrznych właściwości programów z udostępnionym kodem źródłowym. Używana przez Stallmana analogia kodu i przepisu kulinarnego, który można udoskonalić, by następnie podzielić się nim z przyjaciółmi, dobrze obrazuje sposób myślenia, który stoi za ideą *copyleft*. Niemalże dokładnie przeciwstawnej względem tradycyjnej optyki praw autorskich. „Narzędziem wykonawczym" *copyleft* jest licencja GPL (GNU *Public Licence*) najczęściej stosowany quasi-dokument prawny, określający zasady korzystania z wolnego i otwartego oprogramowania. Reguluje ona zakres i charakter praw użytkowników, definiując tzw. cztery wolności: używania, modyfikowania oraz dystrybuowania w postaci oryginalnej jak i zmodyfikowanej. Owe wolności otrzymuje się w zamian za uznanie dwóch nakazów licencji GPL: zachowanie informacji o twórcach kodu oraz konieczność wydania wszelkich dzieł pochodnych na tej licencji. Ostatni z nich sprawił, że licencja ta zyskała znaczne grono przeciwników, dorobiwszy się pejoratywnego określenia GPV – *General Public Virus*. Ten zaraźliwy przymus dzielenia się to dla CEO *Microsoftu* Steve'a Ballmera wręcz „(...) rak pojęcia własności intetelektualnej, który przytwierdza się do wszystkiego, czego dotknie" (Newbart, 2001).

W tym sensie *open source* widziany będzie jako irracjonalny kult, który ustanawia się w opozycji do rynkowej racjonalności *homo oeconomicusa*. Patrząc jednak z drugiej strony, gdyby nie owa wymuszona przez licencje zasada wzajemności, siła napędowa idei FLOSS[3] mogłaby się już dawno wyczerpać. Zachęceni ekonomiczną wartością swoich projektów twórcy oprogramowania otwartoźródłowego mogliby ulec pokusie spieniężenia dotychczasowego wysiłku i zamknięcia go w formie komercyjnego produktu o niedostępnych źródłach. Taką możliwość stwarzają inne licencje otwartego oprogramowania m.in. licencje MIT czy BSD. Licencja GPL – przeciwnie. Skoro uniemożliwia przekształcenie programu w wersję zamknięto-źródłową, jest z założenia proparatekstualna. Stwarza niezmienną bazę tekstów (kodu) umożliwiającą ich dowolne wiązanie czy przetwarzanie, a w efekcie stwarzanie nowych paratekstów. Skoro zatem pewna grupa osób decyduje się przeciwstawić tendencji komercjalizującej kod, co wiąże się z uznaniem pewnej wartości za istotną (wolność użytkownika programu) i dobraniem środków do jej realizacji (tworzenie licencji, organizowanie fundacji etc.), to można zaryzykować tezę, iż mamy tu do czynienia z pewną kulturą w rozumieniu normatywno-dyrektywnym, postulowanym np. przez Jerzego Kmitę (Banaszak, Kmita 1994). Biorąc pod uwagę fakt, że realizacja tej wartości prowadzi bezpośrednio do zwiększenia ilości paratekstów (kodu), określić ją można mianem kultury paratekstu. Będzie to kultura daru, która krzewi się i utrzymuje w sposób świadomy. Dzięki niej świat oprogramowania drastycznie się zmienił, a z nim również nasze „wzory cyberkultury". Od oprogramowania jako towaru do oprogramowania jako twórczej praktyki społecznej *pro publico bono*. Od monolitycznego systemu prawa autorskiego skupionego na interesie jednostki do perspektywy umożliwiającej regulowanie zakre-

[3] Zbiorcze określenie na wolne i otwarte oprogramowanie (*Free Liber*|*Open Source Software*).

su tych praw, jak Creative Commons. A wszystko to z ducha krzemowych obwodów, które znalazły swoich wyzwolicieli.

Bibliografia

Banaszak C., Kmita J. (1994), *Społeczno-regulacyjna koncepcja kultury*, Instytut Kultury, Warszawa.

Benkler Y. (2008), *Bogactwo sieci. Jak produkcja społeczna zmienia rynki i wolność*, Wydawnictwo Akademickie i Profesjonalne, Warszawa.

Castells M. (2003), *Galaktyka Internetu. Refleksje nad Internetem, biznesem i społeczeństwem*, Dom Wydawniczy Rebis, Poznań.

Castells M., Himanen P. (2009), *Społeczeństwo informacyjne i państwo dobrobytu. Model fiński*, Wydawnictwo Krytyki Politycznej, Warszawa.

Douglas J. (2010), *Comments on Comments in Code*, Critical Code Studies Conference, University of Souther California, lipiec 2010, http://vimeo.com/iml/videos [dostęp: 28.03.2011].

Fuller M. (2003), *Behind the blip, Essays on the Culture of Software*, Autonomedia.

Gallaway A. (2008), *Język chce, by go nie dostrzegać*, „Kultura Popularna" nr 4 (22)/2008, Wydawnictwo Szkoła Wyższa Psychologii Społecznej, Warszawa.

Genette G. (1991), *Introduction to Paratext*, [w:] *New Literary History*, vol 22, nr 2.

Giermak-Zielińska T. (2009), *Okładka*, [w:] Skibiniewska E. (red.), *Przypisy tłumacza*, Księgarnia Akademicka, Kraków.

Gwóźdź A. (2010), *Obok filmu, między mediami*, [w:] Gwóźdź A. (red.), *Pogranicza audiowizualności. Parateksty kina, telewizji i nowych mediów*, Towarzystwo Autorów i Wydawców Prac Naukowych Universitas, Kraków.

Hayles N. K. (2005), *My Mother was a Computer: Digital Subjects and Literary texts*, Chicago.

Kittler F. (1995), *There is No Software*, www.ctheory.net/articles.aspx?id=74 [dostęp: 1.04.2011].

Krzysztofek K., *Paratekst jako postfabrykat kultury*, [w:] Gwóźdź A. (red.), *Pogranicza audiowizualności*, Towarzystwo Autorów Tekstów i Wydawców Prac Naukowych Universitas, Kraków.

Lessing L. (2006), *Code: Version 2.0*, http://pdf.codev2.cc/Lessig-Codev2.pdf [dostęp: 28.03.2011].

Loewe I. (2004), *Paratekst w Internecie*, [w:] Sokołowski M. (red.), *Edukacja medialna. Nowa generacja pytań i obszarów badawczych*, Wydawnictwo Kastalia, Olsztyn.

Mackenzie A. (2006), *Cutting Code. Software and sociality*, New York.

Manovitch L. (2006), *Język nowych mediów*, Warszawa.

Manovitch L. (2008), *Software takes command*, http://www.softwarestudies.com/softbook [dostęp: 05.03.2011].

Marino M. (2006), *Critical code studies, Electronic book review*, http://www.electronicbookreview.com/thread/electropoetics/codology [dostęp: 28.03.2010].

Newbart D. (2001), *Microsoft CEO takes launch break with the Sun-Times*, Sun-Times, Chicago, http://web.archive.org [dostęp: 28.03.2010].

Zawojski P. (2006), *Cyberkulturowa rewitalizacja ekonomii daru*, http://www.zawojski.com/2006/11/15/cyberkulturowa-rewitalizacja-ekonomii-daru/ [dostęp: 28.03.2010].