Rafał Ilnicki

Od programu komputerowego do kultury jako programu. Perspektywa studiów nad oprogramowaniem (software studies)

Media, Kultura, Społeczeństwo nr 1 (6), 27-39

2011

RAFAŁ ILNICKI
Uniwersytet im. Adama Mickiewicza w Poznaniu
ilnicki.r@gmail.com

OD PROGRAMU KOMPUTEROWEGO DO KULTURY JAKO PROGRAMU. PERSPEKTYWA STUDIÓW NAD OPROGRAMOWANIEM (*SOFTWARE STUDIES*)

Celem artykułu jest ukazanie w jaki sposób oprogramowanie komputerowe stanowi wyposażenie kultury, jej przedmiot, pełniąc także znaczącą rolę kulturotwórczą. Przestaje być zatem wygodną metaforą na opisanie zjawisk technicyzacyjnych oraz efektem redukcji oprogramowania do poziomu jedynie technologicznego. Dziedziną, która bada wpływ oprogramowania na kulturę są studia nad oprogramowaniem (*software studies*). Jest to podejście transdyscyplinarne łączące takie dziedziny jak kulturoznawstwo, filozofię, informatykę, socjologię, antropologię, kognitywistykę i psychologię. Trudno mówić jednoznacznie o ich początku – wymagałoby to wyodrębnienia refleksji nad oprogramowaniem komputerowym z całej tradycji refleksji poświęconej technice. Tak rozumiana perspektywa genetycznego wywodzenia źródeł studiów nad oprogramowaniem zdecydowanie wykracza poza ramy tej pracy. Współcześnie możemy mówić o tym, że autorem terminu *software studies* jest Lev Manovich.

Najpierw przedstawię powody, dla których warto zajmować się oprogramowaniem komputerowym, wskazując jego podstawowe aspekty kulturotwórcze oraz wprowadzając technologiczne i kulturoznawcze określenie programu komputerowego. Następnie przejdę do opisu wybranych propozycji rozumienia poszczególnych zagadnień przez teoretyków studiów nad oprogramowaniem. Zostaną zaprezentowane wybrane problemy oraz najważniejsze zagadnienia i sposoby formułowania propozycji badawczych. W podsumowaniu ukażę możliwe rozumienie kultury jako programu w ujęciu Siegfrieda Schmidta, co ma potwierdzić przydatność teoretycznego i praktycznego aparatu *software studies* analizy zjawisk kulturowych. Podsumowaniem tych rozważań będzie odniesienie się do rozumienia kultury jako programu o właściwościach samoorganizacyjnych.

Kulturowe znaczenie oprogramowania

George Grant twierdzi, że „Komputer nie narzuca nam, w jaki sposób musimy się nim posługiwać" (za: Barney 2008: 50). Jeśli przyjmiemy tę tezę, to należałoby także uwzględnić perspektywę oprogramowania, bowiem to ono tworzy interfejs w układzie człowiek-maszyna. Sądzę, że tak zmodyfikowane twierdzenie brzmiałoby następująco: „Komputer nie narzuca nam, w jaki sposób musimy się nim posługiwać, jednak oprogramowanie komputera już tak". Oczywistym jest, że sam komputer pojęty tutaj jako sprzęt, materialnie obecny, posiadający określone gabaryty, wagę oraz materiał, z którego został wykonany z pewnością nie zmusza użytkownika do określonego użytkowania. Jeśli jednak włączymy komputer, to koniecznie działa on za pośrednictwem oprogramowania: systemu operacyjnego oraz rozszerzających jego funkcjonalność aplikacji i modułów, które można doinstalować. Współcześnie oprogramowanie komputerowe stało się przezroczyste, co wynika bezpośrednio z jego kulturowego użycia. Do identyfikacji programu, jego określenia i refleksji nad nim dochodzi najczęściej w przypadku awarii, uszkodzenia, wyświetlenia na ekranie błędu. Przeważnie rozpoczynając pracę na komputerze nie zastanawiamy się co zostaje uruchomione, w jaki sposób realizuje się sekwencja startowa oprogramowania, ani też nie myślimy o strukturze takiego systemu operacyjnego. Jedna z wielu metafor, którymi posługiwał się Marshall McLuhan przyczynia się do zrozumienia tego procesu. Kanadyjski teoretyk mediów twierdził, że ludzie w stechnicyzowane środowisko wchodzą „jak do gorącej kąpieli", to znaczy, że media techniczne są niezauważalne (McLuhan 2001: 328). Podkreślona zostaje tutaj moc przyzwyczajenia oraz oczywistości tego, że dane medium funkcjonuje w zupełnie przewidywalny sposób. Nie chodzi o to, że media przestają być odczuwalne, tylko w trakcie posługiwania się nimi ludzie nie zwracają uwagi na samo zapośredniczenie. Należy pamiętać, że:

> Technologia na ogół ucieleśnia i wprowadza pewien określony sposób bycia w świecie, określoną koncepcję stosunków międzyludzkich. Posługując się językiem Heideggera, możemy powiedzieć, że technologia otacza nas lub, jak pisał Grant, technologia jest „całościowym sposobem patrzenia na świat, podstawowym sposobem w jaki człowiek Zachodu doświadcza swojej obecności w świecie" (Barney 2008: 50).

Odnosząc to twierdzenie do programów komputerowych, należy podkreślić, że nie są one neutralne. Podobnie jak kąpiel mogą one mieć różne stopnie gorąca, wanny mogą przybierać różne kształty, a także wieloaspektowo działać na użytkownika. Mam tutaj na myśli przede wszystkim sposób organizacji programu (zarówno jego architektonikę w postaci kodu, jak i reprezentacje na ekranie komputera), które przez sam fakt strukturyzowania przestrzeni, stawiania określonych wymogów dotyczących zachowań względem użytkowników w pewien sposób ich dyscyplinują, zmuszając do podjęcia przez nich określonych działań. Już samo wskazanie kursorem na dany element jest działaniem, tak jak każde kliknięcie myszką, czy wpisanie znaku z klawiatury. Właśnie z tej perspektywy warto przyjrzeć się programom komputerowym oraz ich funkcjonowaniu – od procesu powstawania, poprzez użytkowanie, a skończywszy na ich statusie jako artefaktów kulturowych, akcentując także wpływ procesów technicyzacji na kulturę.

Zwracam uwagę na to, że wpływ oprogramowania nie jest tutaj rozumiany na sposób dystopiczny, na wzór demonicznego technokratyzmu, lecz jako urządzanie ludzkiej fizycznej i kognitywnej przestrzeni. Dlatego też oddziaływanie programów komputerowych odczytuję w perspektywie miękkiego wpływu, to znaczy takiego, który nie determinuje wszystkich działań użytkownika. Już sam fakt obcowania z danym interfejsem wpływa w określony sposób na jednostkę. Musimy spojrzeć w określone miejsce, kliknąć w inne. Można postawić zastrzeżenie, że przecież programy można personalizować i że ktoś o dostatecznie dużej kompetencji informatycznej jest zdolny do samodzielnego tworzenia programów nie będąc tym samym podatnym na ich wpływ. Przedmiotem zainteresowania jest kulturowe „życie" programów, które nie sprowadza się jedynie do użycia i tworzenia oprogramowania przez wykwalifikowanych programistów.

Program komputerowy wywiera wpływ na użytkowników kultury w zależności od swej struktury, przeznaczenia, użytkowania, komputera, czy też innych zainstalowanych programów. Jednak nie sposób zatem nie uwzględnić dziś kulturotwórczego wpływu programów komputerowych, które zdają się być czymś znacznie mniej neutralnym, niż zwykło się uważać, ograniczając je do narzędzi codziennego użytku. Oczywistym jest, że współcześnie to użytkownicy kultury mają względny wpływ na programy, ponieważ mogą oni je konfigurować, zmieniać różne opcje, przez co nie użytkują gotowego produktu. Zamieniają się oni w prosumentów (kategorii tej używam w sensie, jaki nadał jej Alvin Toffler wskazujący na to, że współcześnie jednostki na różnych poziomach konstruują przedmioty oraz swoją aktywnością dopełniają usługi, których używają i z których korzystają) – przez swoje uczestnictwo jednocześnie konsumują dany program, jak i go konstruują. Nie muszą znać języków programowania, wystarczy, że swoim zachowaniem i pragnieniami wpływają na producentów, którzy zabiegają o ich uwagę. Oprogramowanie jest silnie skorelowane z przemysłem informatycznym, czy też przemysłem programów (Stiegler 2009: 3). Jednak nawet gotowy produkt w postaci programu komputerowego najczęściej jest tak stworzony, by wymagał ciągłego rozwijania przez użytkownika. Prosumenci są bowiem celem (targetem) producentów oprogramowania oraz sami też tworzą własne programy komputerowe. Przykładem są tutaj dystrybucje systemu operacyjnego Linux, których kod jest publicznie udostępniony i dzięki temu użytkownicy mogą zaangażować się aktywnie w jego rozwijanie oraz korygowanie odnalezionych w nim błędów. Można tutaj wyróżnić inne społeczności prosumenckie, czy też mikrokultury, ze względu na tworzenie się wspólnotowości wokół danego programu, jak ma to miejsce w przypadku najpopularniejszych programów z racji ich niezbędności dla działania komputera, czyli systemów operacyjnych. Zjawiska te jednoznacznie wskazują na wzrastającą kulturotwórczą rolę programów komputerowych. Przykładem są fora i portale internetowe, społeczności fanowskie, zloty, wydarzenia naukowe i popularyzatorskie, ogniskujące się wokół konkretnych technologii informatycznych. Działania te tworzą wspólnoty oprogramowania, które opierają się na tworzeniu relacji międzyludzkich skupionych wokół danej aplikacji. Mogą one przybierać bardzo różne formy organizacyjne ze względu na poziom zaawansowania i informatycznej kompetencji uczestniczących w nich jednostek. Nie muszą

być one w żaden sposób sformalizowane. Podejmowane przez użytkowników aktywności w obrębie wspólnot oprogramowania przyczyniają się do rozpowszechnienia myślenia o programach komputerowych jako o pełnoprawnych obiektach kultury.

Takiemu rozumieniu tej kulturotwórczej roli aplikacji odpowiada konstruktywistyczna wizja oprogramowania, według której kodowanie przebiega na przynajmniej dwóch poziomach. Pierwszym z nich jest płaszczyzna zespołu developerskiego. Na tym poziomie zostają przygotowywane odpowiednie programy. Jest to poziom producentów oprogramowania. Możemy mówić tutaj o celowym ograniczaniu indeterminacji, czyli o stosowaniu wszystkich zabiegów przez firmy developerskie, które spowodują, że wytwór informatyczny będzie kontrolowalny na każdym etapie wprowadzania go na rynek. Zatem produkt finalny, ujawniany wcześniej w różnych wersjach (np. alfa, beta) mających przygotować go na wejście na rynek, co jest tożsame z jego insercją w tkankę kultury, ma cechować się niezawodnością, doskonałością kodu, wysokim poziomem zabezpieczeń przed nieautoryzowanym przez dane przedsiębiorstwo sposobem rozpowszechniania.

Drugim poziomem jest perspektywa prosumenta, gdzie z wielu różnych użytkowań danego programu (prosumenckiego lub producenckiego) zostają kodowane różne reprezentacje społeczne. Przykładem są tutaj stereotypy dotyczące systemów operacyjnych, gdzie Windows jest uważany za prosty w obsłudze, a Linux za trudny. Kodowanie na poziomie kulturowym realizuje się oddolnie, w opozycji do kodowania producenckiego, które oferuje użytkownikom kompletny program lub też taki posiadający zaprogramowane cechy niekompletności, które jednostka ma wypełnić w trakcie korzystania z niego. Zatem każdy program może mieć szereg paralelnych żyć społecznych, ponieważ producenci nie są w stanie kontrolować w pełni tego, co dzieje się z danym oprogramowaniem. Kategorie użytkownika oraz prosumenta oprogramowania są nieostre. Można równocześnie korzystać z obu tych opcji w różnym wymiarze. Zatem nie ma kulturowej sprzeczności pomiędzy byciem użytkownikiem w pracy, gdzie posługuje się oprogramowaniem producenckim, a wróciwszy do domu korzysta się ze współtworzonych przez daną wspólnotę oprogramowania darmowych aplikacji. Napięcie pomiędzy tymi perspektywami wyznacza kulturotwórczą rolę programu komputerowego, który uwikłany jest w gęste sieci współzależności pomiędzy rynkiem i użytkownikami.

Studia nad oprogramowaniem (*software studies*) – wprowadzenie

Terminu oprogramowanie używam synonimicznie z programem komputerowym oraz aplikacją. Andrew Mackenzie przytacza definicję oprogramowania:

> Ogólny termin dla tych komponentów systemu komputerowego, które są raczej niematerialne niż fizyczne. Jest najczęściej odnoszony do używania programów komputerowych jako różnych od fizycznego sprzętu tego systemu komputerowego oraz do zawierania symbolicznych i wykonalnych postaci takich programów (Mackenzie 2006: 1).

Zatem przedmiotem zainteresowania są instrukcje wykonywane przez komputer. W takim wypadku należy określić, czym jest programowanie.

Przyjmuję szeroką definicję Alexandra Gallowaya, który twierdzi, że użytkowanie jest programowaniem (Galloway 2009). Propozycja ta pozwala nam przestać myśleć o programowaniu jako zagadnieniu technologicznym, którego realizacji podjąć mogą się jedynie wykwalifikowani programiści. Zamiast tego dotyczy ono wszystkich, którzy wykorzystują dane oprogramowanie. Wybierając różne opcje, pracując z danym programem, jest realizowany także proces wykonywania przez komputer poszczególnych instrukcji na poziomie maszynowym. W nim dochodzi do przetworzenia komend na impulsy elektryczne. Zatem płaszczyzna interfejsu, styku człowieka z komputerem jest to raczej poziom kodu kulturowego niż kodu maszynowego, ponieważ użytkownik nie musi znać technologicznych aspektów przetwarzania informacji, by sprawnie obsługiwać daną aplikację. Tutaj Galloway odnosi się do rozumienia cybernetycznego, gdzie programowanie jest sterowaniem. Przede wszystkim jednak program komputerowy jest obiektem technicznym szczególnego typu. Odnosi się on do rezultatów[1] – programować oznacza wprowadzać pewne zależności, których rezultat jest przewidywalny i dostępny przez uruchomienie danego programu komputerowego. Użytkowanie programów rządzi się pewnymi algorytmami kulturowymi, to jest zespołami czynności, które prowadzą do realizacji danego celu, czyli do spowodowania, by maszyna w odpowiedni sposób przetwarzała dane.

Zatem między *software studies* (studiami nad oprogramowaniem) a *platform studies* (studiami nad sprzętem) nie istnieje twarda i nieprzekraczalna sprzeczność. Uważam, że są to dwie perspektywy, z jakich możemy przyglądać się aplikacjom, które muszą być wykonywane na komputerze. Dlatego też kwestia platformy sprzętowej jest tutaj istotna. Jako cyfrowe obiekty techniczne zarówno oprogramowanie, jak i sprzęt pozostają niezależne, jednak takie ich rozumienie odnosi się do pewnej abstrakcji, ponieważ w przypadku kulturowego funkcjonowania programów komputerowych zostają one sprowadzone do jednej płaszczyzny użytkowania.

Należy teraz wskazać na podstawowe związki oprogramowania z kulturą. Pozwoli to zrozumieć zarówno kulturotwórczy charakter oprogramowania, który stanie się podstawą wyróżnienia poszczególnych propozycji ujmowania jego znaczenia w perspektywie studiów nad oprogramowaniem (*software studies*), jak i okaże się pomocny w określaniu kultury jako programu.

Oprogramowanie to głównie kwestia rynku. Możemy mówić o swoistej metafizyce przemysłu programów, które tworzą gotowe produkty doświadczenia ludzkiego, a nie tylko

[1] To określenie wpisuje się w szerszą refleksję nad cywilizacyjną i kulturową obecnością techniki, odnosząc się do mechanizmów determinujących skutek danych rozwiązań. W perspektywie oczekiwania na zaistnienie określonego efektu, rezultatu kryje się istota techniki według Jacquesa Ellula. Oprogramowanie komputerowe stanowi egzemplifikację tego procesu towarzyszącego źródłowo technicyzacji. Możemy bowiem od określonych informatycznych obiektów technicznych spodziewać się natychmiastowości, ale i współcześnie za sprawą personalizacji – intuicyjności, przyjazności użytkownikowi oraz domyślności. Stanowi to skutek pozytywnej waloryzacji w kulturze systemów eksperckich oraz systemów sztucznej inteligencji.

multimedialne obiekty techniczne. Jest to szczególnie wyraźne, jeśli spojrzymy na gry komputerowe, których sprzedaż jest podstawą funkcjonowania przedsiębiorstw, a nie sprzęt, który jest najczęściej jednorazową inwestycją. Istnieje także oprogramowanie darmowe lub to występujące na różnych licencjach dostępu. Programy także wyznaczają nasz sposób dostępu do informacji, a zatem mają wpływ na ukontekstowienie informacji, począwszy od czcionki, przez konstrukcję interfejsu, po ilość włączonych programów. Zatem użytkownik poprzez działanie w obrębie aplikacji rekontekstualizuje cyfrowo zdekonstekstualizowaną informację, ponieważ wprowadza własny kontekst użycia. Powstają ontologie informatyczne służące do zarządzania informacją, które są odrębne od ontologii ujmowanej na sposób historyczno-filozoficzny. Prowadzi to do postawienia pytań, na ile realność organizacji (hierarchii i heterarchii) danych jest realnością społeczną, czyli zyskującą swoje istnienie poprzez systemy katalogowania oraz indeksowania treści przez użytkowników, a na ile odnosi się do samych procesów technologicznych? W jakim stopniu program jest realny, a w jakim wirtualny? Jednak to nie tylko ingerencja informatyki w porządek ontologiczny. To także ontologizacja informatyki – uznanie programów komputerowych za prawowite obiekty wewnątrzświatowe, a nie tylko narzędzia. Programy komputerowe stanowią doskonałe laboratorium do badania aspektów informacji fizycznej, semantycznej, ekonomicznej (Floridi 2010), które wskazują na różne poziomy przejawiania się i krzyżowania aspektów technologicznych i kulturowych w poszczególnych aplikacjach. To także pytanie o język – czy jeśli użytkowanie jest programowaniem, to język naturalny staje się językiem programowania jeśli jest używany do obsługi maszyn? Oprogramowanie jest także istotne z perspektywy kognitywnej – w jaki sposób kierujemy uwagą, jakie procesy mentalne są wykorzystywane do obsługi danego programu, w jaki sposób oprogramowanie wpływa na nasze procesy poznawcze? Koniecznym wydaje się także podjęcie kwestii wykluczenia z dostępu do oprogramowania, metod jego dystrybucji, uwzględnienie szeregu kontekstów kulturowych i społecznych w perspektywie ingerencji podmiotów państwa czy prawa w dane programy. Odniesienie do tych zagadnień pozwoli lepiej zrozumieć w jaki sposób oprogramowanie kształtuje kulturę. Omawianie ich z pewnością przekroczyłoby ramy tego artykułu, jednak wymienienie tych wybranych kwestii ma za zadanie zwrócenie uwagi na to, że pełnią one doniosłą rolę.

Autorzy wskazują na różne obszary, w których można badać oprogramowanie. Nie wykształciło się jedno uniwersalne podejście, raczej możemy mówić o szeregu komplementarnych ujęć, zwłaszcza, ze dyscyplina ta dopiero powstaje wraz z jej podstawowymi założeniami teoretycznymi oraz metodologicznymi. Ukażę propozycje badawcze kilku autorów, a także krótko prześledzę ich myśli.

Studia nad oprogramowaniem – przedstawiciele, zagadnienia, problemy

Za fundatora dyscypliny studiów nad oprogramowaniem uznaje się medioznawcę oraz teoretyka kultury Lva Manovicha, jednak refleksja nad oprogramowaniem istniała już na początku lat dziewięćdziesiątych i związana była z literaturoznawcą

i medioznawcą Fridrichem Kittlerem. Rolę oprogramowania uchwycił także Umberto Eco, przyrównując system operacyjny Macintosh do systemu katolickiego, bowiem jest on jezuicki, kontrreformatorski, elastyczny, „każdy ma prawo do zbawienia", podczas gdy system operacyjny DOS, który jest surowy, ascetyczny, posiada ograniczoną ilość komend – zachowań, został określony jako protestancki, ponieważ dopuszcza interpretację pisma (kodu), jest wymagający, „nie każdy może osiągnąć zbawienie" (Eco 1994). W tych porównaniach bardziej istotna jest sama kulturowa analogia niż bezpośrednie odniesienie do danej religii, która funkcjonuje jedynie jako podstawa metaforycznego ujęcia kulturowej roli systemów operacyjnych. W konwencji przyjętej w eseju Eco można porównać systemy Uniksowe do religii gnostyckich, heretyckich w stosunku do monizmu Microsoftowskiego. W ten sposób unaocznia się cała warstwa znaczeń i skojarzeń, która posiada walor intuicyjnych określeń systemów operacyjnych, które należy traktować z dystansem. Tak pojęta refleksja nad oprogramowaniem zwraca uwagę na wszystkie kwestie odnoszące się do filozofii oprogramowania oraz na zawartą w nich ideologię. Przez te ustalenia prześwituje intuicja, że program komputerowy odczytywany z perspektywy medioznawczej, kulturoznawczej i filozoficznej zdecydowanie wykracza poza technologiczną płaszczyznę obowiązywania.

Na istotowy związek kultury i oprogramowania wskazuje Lev Manovich mówiąc o tym, że kreatywność realizuje się dziś za pośrednictwem danego interfejsu (Manovich 2008). Oprogramowanie to silnik współczesnych społeczeństw, gdzie remiks różnych elementów jest współcześnie podstawową formą produkcji kulturowej. Lev Manovich podkreśla także rolę informatyki w opisywaniu kultury. Dzieje się tak, ponieważ „oprogramowanie umożliwia globalne społeczeństwo informacyjne" (Manovich 2008: 4). Adrian Mackenzie mówi o „społeczeństwie oprogramowania" – to już nie programy w kulturze, ale kultura zogniskowana wokół oprogramowania. Wynika to bezpośrednio z faktu, że programy komputerowe są zainstalowane w bardzo wielu urządzeniach przez co wyznaczają one sposoby dostępu do informacji w większości dziedzin ludzkiego życia. Bankomaty, terminale, systemy rejestracji, kamery, elektroniczne gadżety są wyposażone w różne aplikacje. Podczas ich użytkowania oraz obcowania z nimi jednostki nie zwracają najczęściej uwagi na to, że naciśnięcie przycisku na słupie sygnalizacji świetlnej także inicjuje pewien program komputerowy. Można powiedzieć, że kultura jest wysoko nasycana obiektami i procesami informatycznymi, co prowadzi do tego, że możemy uznać ją za „społeczeństwo oprogramowania".

Mówiąc o performatywności kodu Mackenzie podkreśla, że Linux jest scentralizowany, ponieważ jest wyposażony w *kernel* (jądro systemu), które pozostaje niezmienne, podczas gdy można dowolnie dodawać i usuwać do niego inne aplikacje. Natomiast w społecznych dyskursach temu systemowi operacyjnemu przypisuje się zdecentralizowanie jako właściwość, która miałaby odróżniać go od Windows. Dlatego też Linux reprezentuje zbiorowe działanie w procesie tworzenia się własnego performatywnego krążenia (Mackenzie 2005: 3). System operacyjny Linux ciągle się zmienia, jego kod źródłowy jest rozproszony w szeregu dystrybucji i uaktualnień, podczas gdy w Windowsie efekt podobnego rozproszenia osiągamy dzięki dodatkowym programom, mając jednak zdecydowanie mniejszy wpływ na funkcjonowanie jądra systemu. To,

co czyni systemy operacyjne z rodziny Linux pełnoprawnymi obiektami kultury, to ich sposób organizacji (McKenzie 2005: 12). Podkreślając to, autor wskazuje na to, że mity oraz interpretacje oprogramowania w perspektywie ich kulturowego użycia zdecydowania przekraczają samą specyfikację techniczną tych systemów operacyjnych. Powracając do metaforycznego języka Umberto Eco, a przekładając go na kategorie epistemologiczne, można powiedzieć, że Windows jest systemem dla realistów, a Linux dla antyrealistów i konstruktywistów, bo wymaga kolektywnego działania do dalszego funkcjonowania (Mackenzie 2005: 21). Linux jest bardziej zdecentralizowany od programów Microsoftu, jednak w potocznym dyskursie często określany jest jako absolutnie zdecentralizowany. Cały czas pamiętajmy, że mówimy o kulturowej interpretacji, a nie właściwościach samego kodu, który jest definiowany jako „zasada zamiany wiadomości z jednej formy symbolicznej (alfabetu źródłowego) w inną (alfabet docelowy) (Mackenzie 2006: 3).

Niemiecki teoretyk mediów Friedrich Kittler twierdzi, że aby sprawnie funkcjonować we współczesnym świecie należy znać co najmniej jeden język naturalny oraz jeden język programowania. Nie chodzi o uzyskanie wysokiej sprawności w posługiwaniu się tym drugim, ale zrozumienie ogólnych zasad funkcjonowania programu, który nie został napisany w postaci narracji, lecz za pomocą specyficznego języka określającego instrukcje jakie komputer ma wykonać. Jest to kulturowa hermeneutyka języków oprogramowania. Użytkownik musi zawiesić myślenie o języku naturalnym, jego składni, a zacząć myśleć tak jak to czyni komputer. Nie oznacza to, że mentalnie powtarza w wyobraźni te same operacje logiczne, ale ujmuje ich sens i znaczenie. Kittler twierdzi także, że istotne jest odnajdywanie w kodzie wyrażeń języka naturalnego (za: Mackenzie 2006: 24).

Mark C. Marino proponuje powołanie do życia dyscypliny o nazwie krytyczne studia nad kodem (*critical code studies*). Kod rozumiany jest jako osobny język z właściwą mu retoryką. Paradoksalnie właściwości retoryczne posiada także to, co pozajęzykowe i nie dające się opisać w kategoriach języka naturalne. Odnosi się to do funkcji kodu, jaką jest „programowanie znaczącego" (John Cayley za: Marino 2006). Kod zawiera zatem jednostki sensu, które mogą być odczytywane, ale dopiero po przekroczeniu wymiaru technologicznego, w którym rozumiane są jako instrukcje danego języka programowania:

> Krytyczne studia nad kodem to podejście, które aplikuje krytyczną hermeneutykę do interpretacji kodu kulturowego, architektury programu i dokumentacji wewnątrz społeczno-historycznego kontekstu. KSND (krytyczne studia nad kodem) zakładają, że linie kodu nie mają neutralnej wartości i mogą być analizowane poprzez używanie teoretycznych podejść stosowanych do innych systemów semiotycznych w odniesieniu do poszczególnych metod interpretacji rozwijanych szczególnie w celu dyskusji nad programami (Marino 2006).

Zatem w tym podejściu oprogramowanie na poziomie kodu jest nośnikiem sensów kulturowych.

Alexander Galloway mówi o ideologiach kodu. Jeśli przyjrzymy się, jak działają aplikacje, jakim decyzjom użytkownika przypisują istotność, w jaki sposób ustanawiają one

relacje pomiędzy elementami wewnątrz programów, to spostrzeżemy, że zakładają one określone zachowania i działania jednostki. W ten sposób, korzystając z danego oprogramowania, do pewnego stopnia przejmujemy jego logikę funkcjonowania. Jest to szczególnie widoczne w przypadku programów komunikacyjnych oraz gier wideo. Za przykład podaję grę *Colonization*, w której jedyną strategią gwarantującą sukces jest eliminacja przeciwników. Możemy zdecydować się na inny rodzaj zarządzania swoimi koloniami, jednak wtedy po określonym czasie mamy pewność co do tego, że wojska pozostałych państw z pewnością nas zaatakują. Żeby przeciwstawić się im, należy prowadzić politykę militarną lub też inwestować w obronę, ale to umożliwia jedynie przetrwanie. Zwycięstwo gospodarcze lub dyplomatyczne nie jest możliwe. Interfejs gry jest bardzo przyjazny, grafika zupełnie pozbawiona jest reprezentacji przemocy poza ikonicznymi reprezentacjami wojsk. Dlatego też możemy mówić o ideologii zawartej w kodzie, bowiem regulacje do jakich zmusza nas program nie są widoczne, co nie przeszkadza w tym, żeby implikowały określoną wizję świata i użytkowania programu. Poziom ideologiczności może być bardzo zróżnicowany, jednak to właśnie w grach komputerowych, które są popularnymi aplikacjami i w tym sensie stanowią nośniki kulturowych sensów, dochodzi najczęściej do bardzo wyraźnych ideologicznych rozstrzygnięć.

Galloway podkreśla też materialny wymiar programowania pojętego jako użytkowania, twierdząc, że „ręce (są) ubrudzone kodem". Zatem kod nie byłby czymś nieuchwytnym i niepodlegającym ucieleśnieniu. Zatem program komputerowy funkcjonujący w danej sieci zależności kultury technicznej wykazuje własności performatywne – staje się materialnym w rękach programisty-użytkownika. Prowadzi to do tego, że ideologia zawarta w oprogramowaniu przegrywa z informatycznymi problemami kodu, to znaczy, że sama konstrukcja programu narzuca pewną ideologię – użytkownik może programować, ale tylko w zakresie dozwolonym przez dany program. Każde działanie, które wykracza poza ten rodzaj sprawczości określam mianem *hackingu*, który może przybrać postać modyfikowania, rekonfiguracji, czy też twórczego wykorzystywania błędów danej aplikacji. W takiej perspektywie należy zapytać o to w jaki sposób funkcjonuje podmiot w zestawieniu z kodem. Analityk mediów twierdzi, że:

> Nigdy nie zadaje się pytania gdzie jest podmiotowość, ponieważ komputer jest podmiotowością; i tylko tym. Innymi słowy, jeżeli weźmiemy polityczną koncepcję podmiotowości i przeniesiemy ją do realnie istniejącej materialnej kultury, otrzymamy właśnie komputer (Galloway 2009: 20).

Jest ona zatem określona przez interaktywność, czyli zdolność do działania i wpływania na różne elementy systemu kultury mediów technicznych, gdzie „Interaktywność jest tylko kolejną formą organizacji. Tragedia polega na tym, że jesteśmy zmuszani do bycia interaktywnymi" (Galloway 2009: 21). Na tym polega podstawowy paradoks programowania – podporządkowując się oprogramowaniu działamy w określony sposób. Nie oznacza to totalnego zdeterminowania użytkownika przez obiekty informatyczne, ale dotyczy poddania go określonym wpływom w postaci konieczności dostosowania się do logiki danej aplikacji. Programowanie może przybrać postać uczestnictwa w danej wspólnocie wirtualnej, zdalnej komunikacji, pracy w specjalistycznym programie

graficznym. Wszystkie te aktywności – zarówno programisty, jak i programowanych relacji i zdarzeń – posiadają określoną ideologię, kulturowy sens oraz szereg zależności, które ukrywa przed użytkownikami interfejs. Celem nie jest doszukiwanie się kolejnych płaszczyzn technologicznej głębi, z których można odczytać ideologiczne performansy oprogramowania, ale zwrócenie uwagi na wszelkie kulturowe zależności wbudowane w dane oprogramowanie, które jest abstrakcyjne na poziomie kodu maszynowego, natomiast zyskuje materialny wymiar podczas jego uruchomia przez dany podmiot:

> Kiedy używam Google'a, umieszczam wpis na blogu, wysyłam e-maila tworzę – wraz z milionami innych osób – zachowania o komercyjnym potencjale, z którym Google może wchodzić w interakcje, zarabiając na tym. A ja oczywiście nie mam z tego ani grosza (Galloway 2009: 21).

Zatem ekonomiczne, społeczne i polityczne funkcjonowanie jest już wpisane w kulturową logikę oprogramowania.

Matthew Fuller odwołuje się do ustaleń Gillesa Deleuze'a oraz Felixa Guattariego, którzy uważają tworzenie pojęć za zadanie filozofii. Autor wnioskuje, że oprogramowanie tworzy pojęcia, zatem tworzy „oprogramowanie" dla naszego sensorium, określając horyzont doświadczenia (Fuller 2003: 19). Zatem kodowanie i programowanie to równocześnie odkrywanie ukrytej konstrukcji użytkownika (Fuller 2005: 23).

Kulturotwórczy charakter oprogramowania obejmuje szereg złożonych i wewnętrznie połączonych procesów, które przede wszystkim odnoszą się do płaszczyzny technologicznej, która daje umocowanie kulturze. W tym sensie programy komputerowe mogą być odczytywane jako osobne medium, które jednak zawsze wymaga ucieleśnienia w konkretnym sprzęcie. Dlatego też możemy mówić o wirtualnych społecznościach, o fanach, miłośnikach danych systemów operacyjnych, ponieważ związki międzyludzkie zapośredniczone przez poszczególne aplikacje tworzą kulturę. Jednak sama infrastruktura techniczna nie wyczerpuje kulturotwórczej roli oprogramowania, ponieważ dotyczy ono w dużym stopniu tworzenia sensów, które muszą zostać poddane interpretacji. Programy komputerowe są podstawą rozumienia współczesna świata, ponieważ dostarczają ram myślenia współczesnego człowieka. Jest to szczególnie widoczne w powszednim użyciu języka, w którym zwroty pochodzące ze słownika informatycznego takie jak: „wylogować", „zainstalować", „zresetować" służą do opisania czynności oraz stanów mentalnych ludzi. Należy także podkreślić, że samo oprogramowanie jest już produktem kulturowym, może być nośnikiem ideologii, obalać lub wpisywać się w społeczne stereotypy, ponieważ zostaje ono wytworzone przez podmioty funkcjonujące w określonych symbolicznych ramach. Programowanie zatem nie jest jedynie abstrakcyjnym procesem. Kulturotwórcza rola aplikacji dotyczy także wszelkich trybów wykorzystania programów w działalności egzystencjalnej, komunikacyjnej, hobbystycznej, artystycznej.

Kultura jako program

Po wyróżnieniu kulturotwórczych własności oprogramowania należy zadać istotne pytanie: Czy istnieje możliwość rozpatrywania i analizowania kultury jako opro-

gramowania? W myśl powyższych ustaleń należy uwzględnić wszystkie procesy technicyzacji oraz informatyzacji, które na planetarną skalę zmieniają warunki życia jednostek i wspólnot. Sądzę, że można na wyżej postawione pytanie odpowiedzieć twierdząco odwołując się do ustaleń Siegfrieda Schmidta. Mówi on: „Kulturą nazywam program społecznej, całościowej interpretacji i oceny modelu rzeczywistości jakiegoś społeczeństwa" (Schmidt 2006: 322). Procesy te realizują się za pośrednictwem programów komputerowych, które dostarczają nie tylko narzędzi do pracy, ale programują doświadczenie ich użytkowników. Jednak takie określenie wymaga doprecyzowania, które odnosiłoby się do właściwości samoorganizacji kultury (por. Fleischer 2003: 33). Zatem kultura programuje się, powtarzając i aktualizując zastane połączenia i jednocześnie jest programowana przez użytkowników wprowadzających nowe obwody autoregulacji.

Oprogramowanie odnosi się do podstawowego wymiaru egzystowania człowieka:

> Kultura jako program umożliwia rozwiązywanie problemów kognitywnych i komunikacyjnych w zakresie konstrukcji sensu na względnie długi czas. W ten sposób rozwiązuje dwa centralne zadania determinujące trwałość społeczeństw: kontrolę i reprodukcję. Reprodukcja społeczeństwa dokonuje się bowiem poprzez przekazywanie jednostkom programu kulturalnego w trakcie socjalizacji. Obecne w tym procesie zobowiązania członków społeczeństwa wobec wyznaczonych przez kulturę opcji rozwiązywania problemów (wraz z ich emocjonalnym i normatywnym wymiarem) regulują relacje między porządkami społecznymi a przestrzenią indywidualnej wolności. Kontrola jednostek odbywa się nie poprzez bezpośrednie oddziaływania przyczynowo-skutkowe, lecz poprzez kulturowo zaprogramowane znaczenia – język pełni tu funkcję szczególnie efektywnego instrumentu kulturowej kontroli (Schmidt 2006: 323).

Jednak z racji tego, że różne programy komputerowe są „zainfekowane" różnymi ideologiami, realizują różne strategie polityczne i społeczne, to trudno mówić o jednym globalnym programie komputerowym, który byłby podstawą doświadczenia wszystkich użytkowników kultury. Aplikacje przestają funkcjonować wyłącznie jako obiekty techniczne, stając się programami doświadczenia. Wyznaczają one już nie tylko dostęp do określonej przestrzeni informatycznej, ale do kultury. Programy komputerowe nadal posiadają swój wymiar technologiczny, który jest równoprawny ze społecznym wykorzystaniem. Siegfried Schmidt wskazuje na to, że nie możemy wyróżnić jednego metaprogramu, który obejmowałby różnorodności zjawisk. Twierdzi, że:

> Z reguły programy kulturowe składają się z połączonych ze sobą programów cząstkowych. Tak rozwijają się na przykład funkcjonalnie zróżnicowane programy cząstkowe dla wyróżnicowania systemów społecznych („kultura gospodarcza", „kultura sportowa", które całkowicie nie mogą zostać uzgodnione i połączone). Konflikty między programami cząstkowymi rozwijają zwykle abstrakcyjne przepisy prawne (na przykład prawa nienaruszalności albo prawo własności). W zależności od stopnia wyróżnicowania [*Ausdifferenzierungsgrad*] programu kulturowego powstaje pytanie, czy można jeszcze sensownie mówić o „kulturze" danego społeczeństwa (Schmidt 2006: 323–324).

W myśl tej tezy możemy wyobrazić sobie bardzo silnie spluralizowane kultury, które skupiają się wokół danego programu. Doświadczenia użytkowników takich kultur pozostawałoby nieweryfikowalne – brak byłoby podstaw dla intersubiektywności (programy doświadczenia nie byłyby ze sobą kompatybilne), jeśli specjalizacja progra-

mów byłaby tak wielka, że nie pozwalałaby na wzajemną translację, będącą podstawą wzajemnego zrozumienia. Współcześnie możemy mówić o tym, że logowanie się do aplikacji komunikacyjnych, systemów zarządzania treścią, blogów, dotyczy nie tylko używania określonej technologii, ale wzbudzania szczególnego typu doświadczenia właściwego dla danej przestrzeni elektronicznej. W tym sensie porozumienie oraz uzgadnianie perspektyw zachodzi poprzez wykorzystywanie przez użytkowników podobnego oprogramowania.

Podsumowując te twierdzenia możemy za Siegfriedem Schmidtem powiedzieć, że:

> Kultura (...) jest programem społecznej (Re)konstrukcji zbiorowej wiedzy „w" i „poprzez" kognitywnie autonomiczne jednostki. Jako program materializuje się w swych użyciach, które tylko wtedy stają się społecznie ważne, gdy (zwykle) dotrą do dużej publiczności i są w stanie zdobyć na wystarczająco długi czas trwałe miejsce. Tak wyjaśnia się zasadnicze znaczenie mediów i komunikacji dla każdej kultury (Schmidt 2006: 325).

Zatem to co najważniejsze, to użycie, czyli programowanie. Jeśli kultura jest programem, człowiek jest programowany, to możemy zapytać o to kim jest człowiek? Czy tylko kolejnym programem „cząstkowym"? Oprogramowanie komputerowe ucieleśnia przywołane twierdzenia Siegfrieda Schmidta – realizuje jednocześnie zglobalizowaną i silnie skontekstualizowaną kulturę programów cząstkowych. Jednak wiedza zbiorowości w takim ujęciu nie jest już rekonstruowana w oparciu o materialną obecność, bowiem lokalność programu wyznacza liczba jego użytkowników, a nie teren jego użytkowania, choć i w przypadkach regulacji politycznych i społecznych ta kwestia może posiadać także istotny wpływ na jego funkcjonowanie. Program komputerowy zmusza do nawigowania kulturowego, do kulturowej „włóczęgi" między programami doświadczenia. Należą do nich programy komputerowe obecne w komputerach osobistych, ale także telefonach komórkowych, bankomatach, konsolach do gier oraz pozostałych urządzeniach elektronicznych wykorzystujących oprogramowanie komputerowe. Studia nad oprogramowaniem podejmują krytyczną refleksję nad codziennym programowaniem, które powoduje, że poszczególne dziedziny kultury stają się sferami programów.

Bibliografia

Barney D. (2008), *Społeczeństwo sieci*, Wydawnictwo Sic!, Warszawa.

Eco U. (1994), *Holy War: Mac. vs Dos*, http://www.themodernword.com/eco/eco_mac_vs_pc.html [dostęp: 23.04.2011].

Fleischer M. (2003), *Corporate identity i public relations*, Dolnośląska Szkoła Wyższa Edukacji Towarzystwa Wiedzy Powszechnej, Wrocław.

Floridi L. (2010), *Information a Very short introduction*, Oxford University Press, New York.

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

Fuller M. (2005), *Media Ecologies: Materialist Energies in Art and Technoculture*, MIT Press, Cambridge.

Galloway A. R. (2009), *Język chce, by go nie dostrzegać*, „Kultura Popularna", nr 4 (22).

Mackenzie A. (2005) *The Performativity of Code: Software and Cultures of Circulation*, http://www.lancs.ac.uk/staff/mackenza/papers/code_performativity.pdf.

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

Manovich L. (2008), *Software Takes Command*, http://softwarestudies.com/softbook/manovich_softbook_11_20_2008.pdf [dostęp: 23.04.2011].

Marino M. (2006), *Critical code studies*, Electronic book review, http://www.electronicbookreview.com/thread/electropoetics/codology [dostęp: 23.04.2011].

McLuhan M. (2001), *Wybór tekstów*, Zysk i S-ka, Poznań 2001.

Schmidt S. (2006), *Konstruktywizm jako teoria mediów*, [w:] Kuźma E., Skredno A., Madejski J. (red.), *Konstruktywizm w badaniach literackich*, Towarzystwo Autorów i Wydawców Prac Naukowych Universitas, Kraków.

Stiegler B. (2009), *Technics and Time 2: Disorientation*, Stanford University Press, Stanford.