Michał Owsiak

Wpływ różnic kulturowych na interfejs użytkownika w aplikacjach komputerowych

1. Interfejs użytkownika jako forma komunikacji.

Rozpoczynając pracę z dowolnym programem komputerowym, musimy zdawać sobie sprawę z tego, że poprzez interfejs aplikacji „przemawia" do nas nie bezduszna maszyna, ale człowiek. Tym człowiekiem jest inżynier oprogramowania będący twórcą aplikacji, z którą przyszło nam pracować. Projektanci aplikacji komputerowych poprzez interfejs użytkownika starają się przekazać użytkownikowi najważniejsze dla niego informacje - te, które sprawią, że praca z programem będzie przebiegała w jak najbardziej niezakłócony sposób. Informacje te są oferowane w takiej postaci, aby były dla pracujących z komputerami osób jak najbardziej zrozumiałe i łatwe do przyswojenia. Oczywiście nie zawsze się to udaje. Często intencje projektantów oprogramowania mijają się z rzeczywistymi potrzebami odbiorców aplikacji komputerowych. Bardzo często grupa docelowa odbiorców programu nie jest jednoznacznie określona i to, co łatwe dla jednego, jest niezwykle skomplikowane dla innego użytkownika programu[1]. Ale bywają również sytuacje, w których problem nie leży w samej naturze komputerów - ich znacznej nieprzystępności - ale w naturze człowieka. W jego problemach z poprawnym wyrażaniem się oraz problemach z doborem poprawnych form komunikacji. Jest również tak, że komunikaty kierowane do odbiorców są dla nich niezrozumiałe w wyniku różnic kulturowych, istniejących między twórcami aplikacji komputerowych, a ich użytkownikami. Kolejną, bardzo istotną kwestią, jest brak podstawowej wiedzy twórców o społeczności, do której trafi produkt lub po prostu niechęć do weryfikacji podstawowych informacji, które mogą mieć wpływ na przebieg komunikacji. Dean Leffingwell przytacza wyniki badań Standish Group dotyczące tej problematyki: „Badania zespołu Standish Group wykazały, że «brak danych wejściowych od użytkownika» jest najczęściej wymienianym czynnikiem w kwestionowanych przedsięwzięciach. Chociaż 13% respondentów wskazywało

[1] A. Cooper: *Wariaci rządzą domem wariatów*, WNT, Warszawa 2001, s. 174.

na ten czynnik jako podstawową przyczynę niepowodzenia, to dodatkowo 12% respondentów wybrało «niepełne wymagania i specyfikacje»"[2].

W trakcie tworzenia aplikacji komputerowych, mamy więc do czynienia z wieloma czynnikami mogącymi zaburzyć ten proces. Mogą pojawić się różnice kulturowe uniemożliwiające zrozumienie się twórców i odbiorców produktu, możemy mieć do czynienia ze źle dobranymi metaforami czy wreszcie całkowitym niezrozumieniem potrzeb użytkowników.

2. Symbole i komunikaty.

Graficzny interfejs użytkownika stanowi zbiór metafor. To co w życiu codziennym wykorzystujemy w trakcie powszednich zajęć, znajduje swoje odwzorowanie w programach komputerowych. Mamy do czynienia z metaforą biurka, kiedy pracujemy z pulpitem zawierającym uruchomione programy. Okna aplikacji są metaforą kartek, które zwykle umieszczamy na biurku - stąd, gdy jedno z nich jest na wierzchu, pozostałe leżą pod spodem. Z metaforą mamy do czynienia pisząc elektroniczny list. W zasadzie większość elementów ze świata cyfrowego wyrażana jest w postaci metafor, mających jak najwierniej przybliżać przedmioty ze świata rzeczywistego. Jeżeli jakaś metafora zostanie niefortunnie dobrana, będzie stanowiła problem dla osób korzystających z komputera. Wyobraźmy sobie sytuację, w której ruch myszką w lewo sprawiałby, że kursor porusza się w prawo. Większość użytkowników byłaby zdezorientowana i najpewniej miałaby kłopoty z przyzwyczajeniem się do takiej wizji świata. Właśnie takie, źle dobrane metafory sprawiają użytkownikom aplikacji komputerowych kłopoty w użytkowaniu programów.

Interfejs użytkownika możemy traktować również jako swego rodzaju formę komunikacji *projektant aplikacji - użytkownik*. Cechuje się ona m.in. ograniczeniem, wynikającym z braku bezpośredniej relacji pomiędzy twórcą aplikacji, a jej użytkownikiem. Projektant oprogramowania ma tutaj do czynienia z komunikacją przypominającą model komunikowania masowego. Jego przekaz jest prezentowany różnym odbiorcom, ale nigdy nie wiadomo, która grupa społeczna stanowi największą ich część. Sprzężenie zwrotne wprawdzie istnieje, ale jest w tym przypadku bardzo ograniczone. O wystąpieniu sprzężenia decydują tylko i wyłącznie odbiorcy produktu. Twórca aplikacji nie ma możliwości wpłynięcia na użytkowników tak, aby ci przekazali informację zwrotną dotyczącą wykorzystywanego oprogramowania. Wydaje się, że najlepiej opisującym tę relację modelem jest *Interakcyjny model komunikowania masowego*[3]. Mamy tutaj do

[2] D. Leffingwell, D. Widrig: *Zarządzanie wymaganiami,* WNT, Warszawa 2003, s. 92.
[3] B. Dobek-Ostrowska: *Podstawy komunikowania społecznego*, Astrum, Wrocław 2004, s. 108.

czynienia z wzajemnym interpretowaniem komunikatów słanych zarówno przez twórcę oprogramowania (interfejs aplikacji, instrukcja obsługi) oraz jego użytkowników - w postaci sprzężenia zwrotnego. Sprzężenie zwrotne wykorzystuje zwykle kanały udostępnione przez twórcę oprogramowania: strony WWW, adresy e-mail, linie telefoniczne.

W związku z tym, że interakcja *programista - użytkownik* jest w znaczny sposób ograniczona, twórcy produktów programowych muszą bardzo dokładnie dobierać formę przekazu do osób, które stanowią grupę docelową produktu. Ten wybór dotyczy doboru odpowiedniego języka, formy przekazu oraz odpowiedniej symboliki[4].

Dobór symboli, zbioru metafor oraz języka, którym porozumiewa się program z użytkownikiem muszą być rezultatem wcześniejszych badań przyszłych użytkowników. Badając ich rzeczywiste potrzeby, stwierdzając kim są, możemy zaoferować im to, co będzie dla nich najbardziej odpowiednie. *Język* programu dla nastolatków może dalece odbiegać od tego co zostanie zaproponowane pracownikom biurowym. *Język gier* może znacznie się różnić od *języka programów użytkowych*. Niemniej jednak to na twórcach spoczywa obowiązek takiego doboru przekazu, aby nie był niestosowny, czy wręcz obraźliwy.

3. Ten „inny".

Bardzo ważną kwestią w trakcie prac nad oprogramowaniem jest dbałość o szacunek dla innych kultur. Wytwarzając produkt możemy narazić się na śmieszność czy wręcz wrogość ze strony naszych klientów, jeżeli nie będziemy zwracać należytej uwagi na aspekt kulturowy. Interesującym przykładem jest między innymi błąd, jaki popełniła firma Microsoft. W trakcie prac nad swoim oprogramowaniem (system operacyjny Windows XP) nie ustrzegła się dosyć zabawnego, a dla naszych rodaków wręcz gorszącego, błędu - opracowana przez firmę mapa świata nie uwzględniała terytorium Polski. Skutkiem tego Polska stała się częścią Morza Bałtyckiego, a Europa zmniejszyła swoją powierzchnię. Takie działanie wydaje się być mocno niestosowne. Innym przykładem może być gra *Codename Panzers*, której twórcy twierdzą, że stroną odpowiedzialną za rozpoczęcie drugiej wojny światowej była Polska. Takie błędy wydają się o tyle niebezpieczne, że mogą wpłynąć w znaczny sposób nie tylko na reputację firmy, ale nawet prowadzić do konfliktów na szczeblu dyplomatycznym. Wprawdzie producenci tego produktu próbowali wybrnąć z całej sytuacji obronną ręką, twierdząc iż gra miała dokładnie odwzorowywać realia wojny (fałszywy komunikat o ataku polskich oddziałów na niemiecką radiostację), niemniej jednak niesmak

[4] A. Cooper: *Wariaci rządzą domem wariatów,* WNT, Warszawa 2001, s. 222.

pozostał. Nieszczęściem w tej całej sytuacji jest to, że bez jasnego przekazu oraz informacji dlaczego takie fakty znalazły się w grze, dokonywana jest kreacja historii w umysłach graczy. Jest to o tyle niebezpieczne, że kraje nie znające historii regionu, mogą wyrobić sobie na jej temat błędne mniemanie.

Odrębny problem stanowi oprogramowanie użytkowe, w którym bardzo często umieszcza się tzw. ikony, symbolizujące określone akcje użytkownika (np. symbol nożyczek jako określenie akcji „wytnij"). Okazuje się bowiem, że całkiem neutralne dla nas symbole, mogą w innej kulturze znaczenie nacechowane negatywnymi emocjami. Czy symbol „świnki" jako skarbonki jest dobrze dobraną metaforą? Przecież w niektórych krajach świnia jest uważana za zwierzę nieczyste. Wykorzystywany przez nas symbol uniesionej ręki - „stop" - w niektórych krajach jest gestem obraźliwym[5]. Wydaje się więc, że dobór symboli, które umieszczamy w programie, może mieć znaczenie większe, niż nam się wydaje. Kwestia dotyczy nie tylko „ikon". Ważna jest również wtedy, gdy rozpatrujemy język, jakim posługują się użytkownicy produktów programowych.

Języki różnią się między sobą alfabetem, ale również kierunkiem stawiania znaków (chodzi mi tutaj tylko i wyłącznie o graficzną reprezentację języka). Jeżeli teraz wyobrazimy sobie program, który ma służyć do wprowadzania informacji w postaci tekstu (np. formularz adresowy), błyskawicznie zauważymy, że mogą pojawić się znaczne problemy. Nie w każdej kulturze pisze się „od lewej do prawej", nie wspominając o tym, że istnieją języki, w których używa się całkiem innego alfabetu niż łaciński. Jak więc ma sobie poradzić np. Azjata z programem, który został napisany dla Europejczyków? Użytkownik z innego kręgu kulturowego zostaje postawiony w sytuacji, w której jego „świat językowy" zostaje wywrócony do góry nogami. Mało tego, taki człowiek może po prostu poczuć się urażony, uznając, że producent oprogramowania wykazał się ignorancją i brakiem znajomości jego kultury.

Opracowanie uniwersalnej formy przekazu wydaje się wręcz niemożliwością. Niemniej jednak producenci oprogramowania powinni dbać o to, by jego użytkownicy nie czuli, że ich świat został postawiony na głowie. Producent powinien dbać również o to, by dobrane metafory nie tylko były trafne, ale również nie niosły ze sobą negatywnych treści.

4. Problemy natury prawnej.

Problemy natury prawnej są w dzisiejszych czasach szczególnie istotne ze względu na łatwy dostęp do treści oferowanych w internecie. Tworząc program komputerowy musimy zdawać sobie sprawę z tego, że nie wszystko co jest legalne w naszym własnym kraju, musi być legalne na całym świecie. Stawia to twórców

[5] J. Spolsky: *Projektowanie interfejsu użytkownika,* MIKOM, Warszawa 2001, s. 120.

oprogramowania w szczególnej pozycji. Oferując swój produkt poprzez internet muszą sobie zdawać sprawę, że to, co jest w pełni legalne w kraju ich działalności, może prowadzić do reperkusji prawnych na drugim końcu globu. Znakomitym przykładem są tutaj tak zwane metody silnego szyfrowania danych. Twórcy oprogramowania, stosujący takie algorytmy, muszą zdawać sobie sprawę z tego, że ustawodawstwo różnych krajów podchodzi w różny sposób do tej problematyki. Kraje takie jak Stany Zjednoczone Ameryki bardzo restrykcyjnie podchodzą do kwestii eksportowania tego typu algorytmów - narażony na reperkusje jest tutaj twórca oprogramowania, który udostępniając produkt w sieci, może narazić się na oskarżenie złamania prawa. Z kolei w Chińskiej Republice Ludowej, gdzie korzystanie z tego typu mechanizmów nie jest mile widziane przez władze, na reperkusje narażeni są wszyscy użytkownicy komputerów. Umieszczając więc w kodzie produktu mechanizmy szyfrowania danych, narażamy odbiorcę produktu z tego regionu na nieprzychylności ze strony aparatu państwa. Najprostszym przykładem jest aplikacja pocztowa pozwalająca na szyfrowanie danych. Jeżeli, jako twórca oprogramowania, pozwolimy na niczym nieskrępowany dostęp do naszego produktu, narażamy się na łamanie prawa eksportowego USA. Z drugiej strony, używając programu w Chinach, możemy zostać oskarżeni o działalność szpiegowską. Równie ciekawą wydaje się również kwestia patentów na oprogramowanie. To, co legalne w Europie, może się okazać chronionym prawnie elementem oprogramowania w Stanach Zjednoczonych Ameryki. Znakomitym przykładem może być algorytm szyfrowania RSA[6], opatentowany w USA, ale nie chroniony prawnie w Europie[7].

Odrębną kwestię stanowią przepisy dotyczące legalności określonych treści. Różne ograniczenia wiekowe dotyczące legalności prezentowanej treści mogą stawiać użytkowników naszego oprogramowania w co najmniej niezręcznej sytuacji. To, co w Europie jest legalne od osiemnastego roku życia, w USA jest zwykle legalne od dwudziestego pierwszego roku życia. Jeżeli nasz produkt zawiera takie treści, które podlegają ograniczeniom wiekowym, możemy narazić odbiorców naszego produktu na konsekwencje prawne.

Problem kwestii prawnych jest niezwykle skomplikowany i stanowi wyzwanie raczej dla prawników, niż twórców oprogramowania. Wydaje się wręcz niemożliwym dostosowanie wszystkich możliwych produktów programowych do wszystkich możliwych ograniczeń wynikających z ustawodawstwa poszczególnych krajów. Samo zawarcie wszystkich obostrzeń w umowach zawieranych pomiędzy twórcami oprogramowania a jego użytkownikami, wydaje się nierealne. Musiałyby to być opasłe tomy zawierające wszystkie szczególne przypadki związane

---

[6] T. H. Cormen: *Algorytmy Teorio-liczbowe we Wprowadzenie do algorytmów*, WNT, Warszawa 2005, s. 902-909.

[7] http://www.cyberlaw.com/rsa.html

z wykorzystaniem oprogramowania. Na dzień dzisiejszy, wydaje się, że wszyscy po trosze znajdujemy się poza prawem.

5. Podsumowanie.

Wydaje się, że uzyskanie jednolitego, jednoznacznie rozumianego interfejsu użytkownika jest skomplikowane, jeżeli wręcz nie niemożliwe. Należałoby znaleźć taki podzbiór wszystkich języków, który zawierałby treści identycznie rozumiane przez wszystkie kultury. To samo dotyczy symboli oraz znaków. Już sam fakt różnego sposobu zapisu tekstu sprawia, że tworząc program musimy mieć na uwadze przyszłą konieczność jego modyfikacji. Wydaje się więc, że twórcom produktów programowych pozostaje dbać o to, by dostarczany na dany rynek program był za każdym razem dostosowany do wymogów danej kultury.

Bibliografia:

1. B. Dobek-Ostrowska: *Podstawy komunikowania społecznego,* Astrum, Wrocław 2004

2. A. Cooper: *Wariaci rządzą domem wariatów,* WNT, Warszawa 2001

3. J. Spolsky: P*rojektowanie interfejsu użytkownika,* MIKOM, Warszawa 2001

4. T.H. Carmen*: Algorytmy Teorio-liczbowe,* w: *Wprowadzenie do algorytmów,* WNT, Warszawa, 2005, s. 902–909.

5. P .J. Flinn, J. M. Jordan III: *Can You Encrypt, Decrypt, Sign and Verify without Infringing the RSA Patent?*, Alston & Bird LLP (http://www.cyberlaw.com/rsa.html)

6. D. Leffingwell, D. Widrig: *Zarządzanie wymaganiami*, WNT, Warszawa 2003