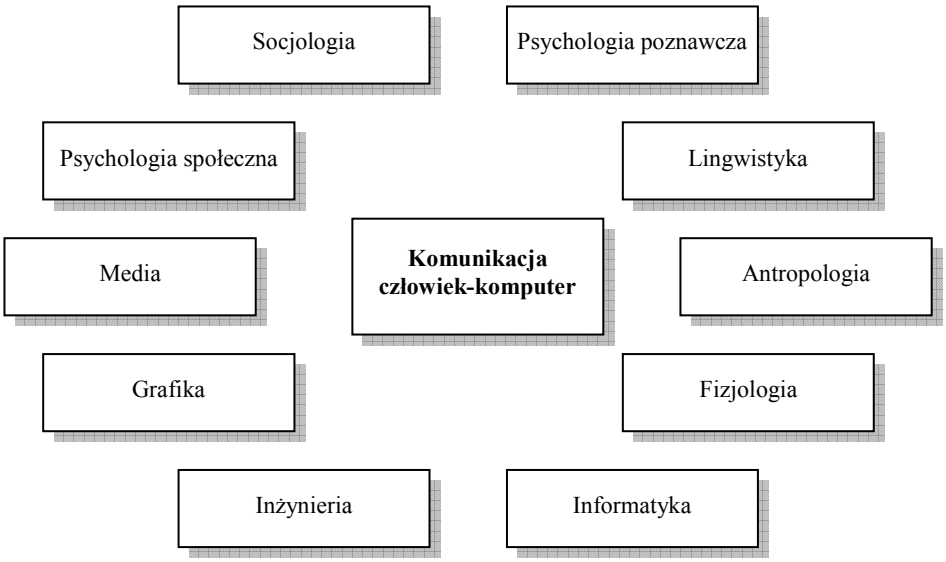
Sebastian Kostrubała

KOMUNIKACJA CZŁOWIEK-KOMPUTER

Wprowadzenie

Komunikacja człowiek-komputer (*human computer interaction*) jako termin naukowy pojawił się w literaturze we wczesnych latach 80. W tym okresie następowała zmiana w definiowaniu i podejściu do użytkownika komputera. Projektanci systemów komputerowych zauważyli i zaakceptowali zmianę głównego odbiorcy systemów z profesjonalistów na ludzi niezwiązanych zawodowo z informatyką. Oznaką takiego podejścia było zdefiniowanie terminu „naiwnego użytkownika" przez Easona w 1976 roku [Easo76, s. 3-7] oraz wypowiedź z 1983 roku wiceprezesa firmy IBM prof. Branscomba, odpowiedzialnego za badania naukowe, który stwierdził, że komputery przestają być ekskluzywnymi narzędziami dla naukowców, a stają się powszechne i niezbędne dla wszystkich. W tym okresie firma IBM poświęciła większość swoich badań zagadnieniom dotyczącym łatwego użycia komputera. Lata 80. to rozkwit komputerów osobistych, stąd nie może dziwić ilość prowadzonych w tym czasie badań. Sama problematyka komunikacji pomiędzy człowiekiem a maszyną pojawiła się jednak w badaniach naukowych znacznie wcześniej. Przykładem może być założone w 1949 roku towarzystwo naukowe Ergonomics Research Society. Zostało one powołane po to, by zająć się m.in. badaniami nad „dopasowaniem maszyny do człowieka". Członkowie tego towarzystwa już wtedy zauważali, że problem komunikacji człowiek-maszyna jest problemem interdyscyplinarnym, łączącym budowę maszyn z takimi dziedzinami, jak np. anatomia, czy psychologia [WWW8]. Wraz z rozwojem badań nad interakcją użytkownika z komputerem, wyróżniono znacznie więcej powiązanych ze sobą dyscyplin. Przykładowe powiązania pokazano na poniższym rysunku.



Rys. 1. Interdyscyplinarność problemu komunikacji człowiek-komputer

Źródło: [SaJB08, s. 535].

Przyglądając się rysunkowi, można dojść do wniosku, że idealny projektant interaktywnych systemów będzie dysponował wiedzą i doświadczeniem z zakresu: psychologii, nauk poznawczych, ergonomii i informatyki. Oprócz zrozumienia zasad i wymagań dotyczących samego procesu projektowania, będzie on musiał posiadać także umiejętności związane z inżynierią oprogramowania, potrzebne do zbudowania niezbędnych narzędzi. Naturalne w procesie projektowania i budowy interfejsów wydają się być umiejętności graficzne. Dobrze byłoby, gdyby projektant, oprócz umiejętności inżynierskich, potrafił zrozumieć szerszy kontekst interakcji użytkownika, czyli posiadał wiedzę z socjologii. Oczywiście znalezienie tak opisanego, idealnego projektanta systemów interaktywnych jest w rzeczywistości niemożliwe. Gorzej, że również stworzenie takiego zespołu projektowego, który posiadałby specjalistów z każdej z wymienionych dyscyplin, należy do rzadkości. W związku z tym, w praktyce obserwuje się tendencję do projektowania interakcji pomiędzy człowiekiem a komputerem z punktu widzenia określonej dyscypliny lub kilku głównych dziedzin, najczęściej: informatyki, psychologii i nauk poznawczych [DFAB04, s. 5]. Próbę opisania jaką rolę odgrywają poszczególne dziedziny w procesie projektowania można odnaleźć m.in. w pracy Paula Booth'a [Boot89, s. 7-12].

1. Modele projektowania systemów interaktywnych

Tradycyjne podejście do projektowania systemów komputerowych nie sprawdza się podczas projektowania interakcji pomiędzy człowiekiem a komputerem. Model kaskadowy oraz późniejsze modele uwzględniające iterację pomiędzy poszczególnymi działaniami są niewystarczające, głównie ze względu na to, że koncentrują się na technicznym aspekcie działania systemu [SmAt06, s. 48]. Potrzebę innego podejścia do projektowania systemów interaktywnych dostrzeżono już w latach 80., gdy rozpoczęto na szeroką skalę badania nad komunikacją człowiek-komputer. Podstawy postępowania przy projektowaniu tego rodzaju systemów opracowali Gould i Lewis, formułując 3 zasady:

- zbliżenie do użytkownika i zadań, które powinny być realizowane (zrozumienie użytkownika i jego zachowań),
- empiryczny pomiar wydajności użytkownika (poprzez analizę jego pracy z prototypami),
- projektowanie iteracyjne [GoLe85].

Zmiana podejścia projektantów z ukierunkowania się na aspekty techniczne systemu na zwrot ku problemom użytkownika stała się podstawą nowego podejścia zaproponowanego przez profesora Normana – projektowaniu zorientowanym na użytkownika (*user-centered design*) – [Norm88]. W ramach tego nurtu powstało wiele modeli i sposobów projektowania systemów opisanych przez naukowców zajmujących się tą problematyką (m.in. Nielsen – znany ze swych wieloletnich badań nad użytecznością, Kreitzberg, Dix, Preece, Beyer, Cooper). Warto przypomnieć model profesora Nielsena, ponieważ był on jednym z pierwszych i stał się inspiracją dla kolejnych autorów. Zaproponował on jedenaście etapów tworzenia interaktywnych systemów:

1. Poznanie użytkownika:
   a) indywidualne cechy użytkownika,
   b) bieżące i przyszłe zadania użytkownika,
   c) analiza funkcjonalna,
   d) ewolucja użytkownika i pracy.
2. Analiza konkurencyjności.
3. Określenie celów użyteczności.
4. Projektowanie równoległe.
5. Współprojektowanie.
6. Skoordynowane projektowanie całego interfejsu.
7. Stosowanie wytycznych i analizy heurystycznej.
8. Prototypowanie.

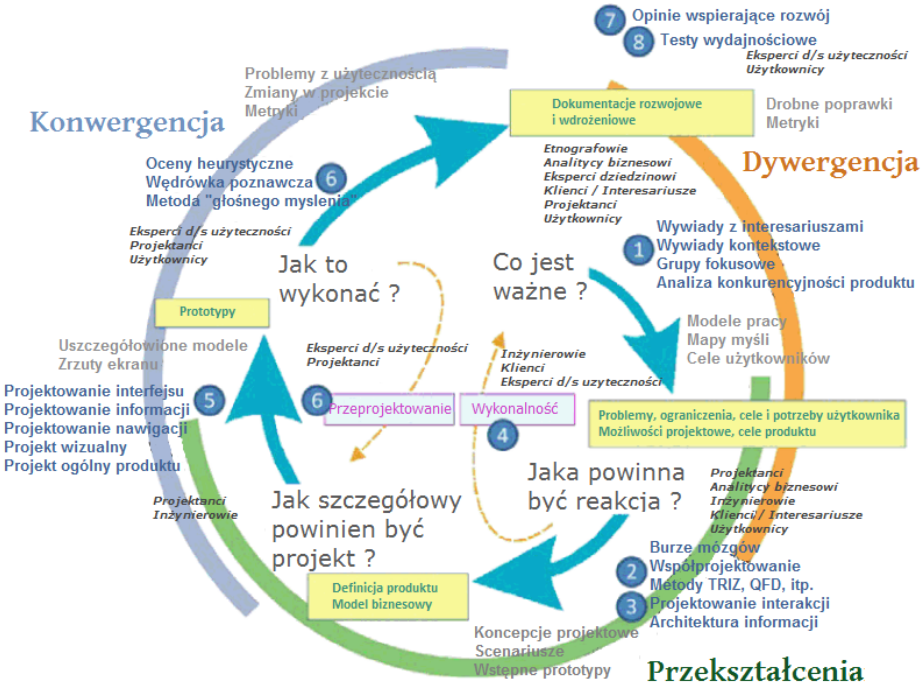
9. Testy empiryczne.
10. Projektowanie iteracyjne
11. Zbieranie informacji zwrotnych z pola użycia [Niel93, s. 71-110].

Należy przy tym zaznaczyć, że Nielsen opracował zestaw kroków i ograniczył się do stwierdzenia, że przejście przez nie wszystkie nie jest wymagane. Pozostali specjaliści dziedziny, rozwijając temat i opierając się na swoim doświadczeniu zasugerowali, które z tych kroków raczej powinny być wykonane, a które można pominąć.

Kolejnym znanym podejściem jest projektowanie kontekstowe zaproponowane przez Beyera i Holtzblatt (dość powszechnie stosowane przez zespoły projektowe stosujące metody zwinne). Podejście to integruje kilka technik projektowania zorientowanego na użytkownika w jeden proces projektowy, którego kluczowym problemem jest określenie tego jak użytkownik będzie pracować w przyszłości. To jak system ma działać, jaka ma być jego struktura i jakie zostaną użyte do jego stworzenia technologie jest zdeterminowane zebranymi danymi od użytkowników [BeHo98, s. 3].

W większości przypadków wymienieni wcześniej autorzy tworzyli swe podejścia projektowe kilkanaście lat temu. Nie oznacza to jednak, że są one nieaktualne lub też zamykają możliwości tworzenia rozwiązań opartych na komunikacji człowiek-komputer. Aktualność podejścia zorientowanego na użytkownika potwierdzają swoimi zaleceniami wiodące firmy informatyczne, takie jak np. IBM [WWW12]. Na podstawie tego podejścia został również opracowany w 1999 roku międzynarodowy standard ISO 13407, opisujący proces projektowania komputerowych systemów interaktywnych (*Human-centred design for interactive systems*) zorientowany na człowieka (szerszy kontekst niż projektowanie zorientowanie na użytkownika czy konsumenta). Dostarcza on wymagania i rekomendacje zasad oraz czynności poprzez cały cykl życia systemu interaktywnego. Standard ten został zastąpiony w 2010 roku przez ISO 9241--210. Na temat problematyki projektowania systemów interaktywnych powstaje nadal duża ilość publikacji naukowych, a o wadze problemu mogą świadczyć liczne konferencje naukowe poświęcone tej tematyce, m.in. cykl konferencji Human-Computer Interaction, czy Human-Centred System Engineering. Nadal trwają poszukiwania jak najlepszego modelu projektowania oraz najlepszych wzorców postępowania dla zadań wchodzących w skład tych modeli. Przykładem może być publikacja z 2010 roku, w której autorzy próbują po raz kolejny stworzyć model oparty na najlepszych do tej pory praktykach projektowych [JoSa10, s. 166-181]. Zaproponowany model składa się nie tylko z opisu

8 konkretnych działań, ale zawiera również inne użyteczne informacje projektowe, takie jak: przynależność do określonych faz, uczestnicy biorący udział w kolejnych czynnościach projektowych, metody jakie powinny być stosowane podczas tych czynności i efekty działań, którymi poszczególne działania powinny się zakończyć.



Rys. 2. Proces projektowania systemów interaktywnych

Źródło: [JoSa10, s. 169].

Duża różnorodność dostępnych modeli projektowania systemów interaktywnych nie zmienia faktu, że istnieje kilka podstawowych zasad, które w przypadku tych systemów powinny być spełnione. Udane systemy interaktywne to takie, które są:

– użyteczne: robią to do czego zostały zaprojektowane,
– możliwe do użycia: pozwalają na naturalną interakcję, bez zagrożeń lub niespodziewanych błędów,
– używalne: sprawiają, że ludzie chcą ich używać.

Istnieją prawne regulacje określające jak powinna wyglądać interakcja z użytkownikiem, zabezpieczające potencjalnych użytkowników przed oprogramowaniem niespełniającym powyższych założeń. Przykładem może tu być dyrektywa Unii Europejskiej dotycząca minimalnego bezpieczeństwa i zdrowia przy pracy z ekranami monitorów (90/270/EEC), nieograniczająca się jedynie do określania norm i standardów zdrowotnych, a wymuszająca także projektowanie systemów zgodnych z wytycznymi:

a) oprogramowanie musi odpowiadać zdefiniowanym zadaniom,
b) oprogramowanie musi być łatwe w użyciu, a tam gdzie to możliwe, dostosowane do poziomu wiedzy lub doświadczenia operatora; nie można używać bez wiedzy pracownika żadnych urządzeń do kontroli jakościowej lub ilościowej,
c) systemy muszą zapewniać przekazywanie pracownikom informacji zwrotnej o ich działaniu,
d) systemy muszą wyświetlać informacje w formacie i tempie dostosowanym do operatorów,
e) muszą być stosowane zasady ergonomii oprogramowania, w szczególności przy przetwarzaniu danych dotyczących ludzi [WWW9].

Wymienione powyżej zasady są jednymi z wielu dostępnych zasad projektowania przeznaczonych dla projektantów systemów interaktywnych. Niektóre z tych zasad zostały opisane w dalszej części tekstu.

2. Projektowanie interfejsu

Projektowanie interfejsu użytkownika jest jednym z wielu zadań składających się na proces projektowania systemów interaktywnych (rys. 2). Przyjmując jednak podejście zorientowane na użytkownika jest to ta część projektowanego systemu, która jest najbliższa użytkownikowi. Metoda prototypowania systemów interaktywnych w głównej mierze opiera się na generowaniu kolejnych modyfikacji interfejsu użytkownika. Przyglądając się cyklowi życia systemu interaktywnego, można zauważyć, że już w początkowej fazie projektowania powstają wstępne prototypy interfejsu w postaci szkiców formatek oraz scenariuszy kilku lub kilkunastu formatek, ułożonych w sekwencje mające imitować nawigację użytkownika po systemie. W tej wczesnej fazie opracowywania interfejsu niezwykle ważny jest aktywny udział użytkowników, którzy na tym interfejsie mają pracować. Wspólne projektowanie interfejsu realizuje koncepcję zorientowania na użytkownika, niesie jednak ze sobą zagrożenie wydłużenia procesu projektowego lub skupienia się na rzeczach mało istotnych w tej fazie

projektowej (kolory, kształty itp.). W związku z tym warto wykorzystać praktyczne metody wspólnej pracy z użytkownikiem nad projektem interfejsu. Jedną z takich metod jest przeprowadzenie sesji JAD (Joint Application Development) z udziałem użytkowników, analityków, projektantów i programistów. W przypadku gdy sesja JAD jest poświęcona działaniu całego systemu, wśród uczestników powinni również znaleźć się menedżerowie. Kluczem do powodzenia sesji jest wyodrębnienie lidera z dużymi umiejętnościami organizacyjnymi i interpersonalnymi [ShCR09, s. 141]. Jego zadaniem jest sterowanie procesem projektowania oraz ucinanie rozważań nad kwestiami nieistotnymi. Jeżeli projektowanie systemu odbywa się według jednego z wcześniej wymienionych modeli, na tym etapie powinny istnieć już zebrane wymagania użytkowników dotyczące systemu oraz być wyodrębnieni aktorzy i ich przypadki użycia. W takiej sytuacji jedną z metod jest wychodzenie od wyodrębnionych przypadków użycia do konkretnych projektów interfejsów realizujących wymagania użytkownika związanego z omawianym problemem. W celu silniejszego aktywowania użytkownika w działaniach projektowych, wstępne projekty są opracowywane wspólnie na papierze albo w postaci rysunków, albo w postaci dużego arkusza papieru, gdzie są naklejane karteczki z oznaczeniami komponentów, które powinny zostać użyte do realizacji konkretnego interfejsu. Prototypy papierowe pozwalają na opisanie ogólnych cech interfejsu, na ich podstawie projektanci mogą opracować bardziej złożone projekty interfejsu już z użyciem narzędzi komputerowych pozwalających na tworzenie projektów interfejsu, takich jak: Microsoft Visio, czy Enterprise Architect. Dopuszczalne jest także użycie środowisk programistycznych pozwalających na projektowanie interfejsów (takich jak Microsoft Visual Studio, czy Eclipse). Na tym etapie tworzenia systemu interaktywnego nie zawsze jest to jednak wskazane, ponieważ efektem działania takich programów są działające prototypy, które w przypadku niedoświadczonych użytkowników sugerują finalny wygląd i działanie interfejsu, co prowadzi do niepotrzebnych nieporozumień na linii użytkownik-projektant. Dodatkowym atutem użycia narzędzi „czysto" projektowych jest możliwość tworzenia scenariuszy, czyli sekwencji projektów interfejsów wizualizujących realizację określonego przypadku użycia. Scenariusze w przeciwieństwie do projektów pojedynczych ekranów interfejsu pozwalają zaprojektować wstępną nawigację po systemie. Tworzenie scenariuszy i wstępnych prototypów odbywa się na w miarę ogólnym poziomie, dlatego projektanci mogą zadbać o całą strukturę interfejsu systemu. Jest to dość ważne, by na każdym etapie projektowania pamiętać również o całym rozwiązaniu, a nie skupiać się tylko i wyłącznie na pojedynczych ekranach systemu. Takie podejście zapewnia spójną nawigację, sposób działania i wygląd w obrębie całego systemu.

Kolejnym etapem jest poddanie szkicowych projektów analizie i ocenie, a następnie cały zespół projektowy wraz z użytkownikami musi podjąć decyzję o wstępnej akceptacji opracowanych interfejsów lub ponownym ich przeprojektowaniu. Zaakceptowane prototypy powinny zostać przekształcone w prototypy funkcjonalne, które już w sposób bezpośredni imitują docelowy wygląd interfejsu. W tym celu należy wykorzystać wymienione wcześniej środowiska programistyczne, w przypadku projektowania aplikacji internetowych mogą to być wstępne, opracowane strony w języku HTML. Na tym etapie tworzeniem prototypów zajmują się specjaliści, których zadaniem jest zaproponowanie takich interfejsów, które po pierwsze są rozwinięciem prototypów powstałych we wcześniejszych krokach, a po drugie są zrealizowane zgodnie z obowiązującymi regułami i wskazówkami projektowymi.

Działające prototypy powinny zostać poddane operacji testowania i oceniania. Testowanie interfejsu może odbywać się za pomocą różnych metod i technik, których wspólnym celem jest sprawdzenie i dalsze doskonalenie projektu interfejsu. Najczęściej spotykanymi podejściami do testowania i oceniania interfejsu są: oceny heurystyczne, metoda przejścia się po interfejsie, oceny interaktywne i formalne testowanie użyteczności interfejsu [DeWR08, s. 323]. Ocena heurystyczna jest zazwyczaj realizowana przez jedną z osób zespołu projektowego, który sprawdza czy zaprojektowany interfejs spełnia określone zasady i wzorce projektowe. Również w metodzie przejścia przez interfejs obecny jest członek zespołu, który prezentuje końcowemu użytkownikowi prototypy interfejsu, a ten zgłasza swoje uwagi. Metody oceny interaktywnej i testów użyteczności są realizowane już bezpośrednio przez użytkowników pracujących na systemie. W pierwszej z nich testowanie przez użytkownika odbywa się na zasadzie podoba/nie podoba się, w drugiej testy odbywają się w sposób formalny, najczęściej w specjalistycznych laboratoriach mierzących sprawność pracy użytkownika z danym interfejsem. Formalne testy użyteczności są uznawane za najdroższe i są stosowane zazwyczaj przy większych projektach. Do prawidłowego przeprowadzenia testów użyteczności wystarczy 5-10 użytkowników [WWW11].

Proces projektowania interfejsu jest procesem iteracyjnym, dlatego po wykonaniu fazy ocen i testów następuje powrót do fazy tworzenia prototypów, gdzie następuje poprawa lub opracowanie nowych prototypów interfejsu.

3. Zasady i wzorce projektowe

Podobnie jak w przypadku modeli całego cyklu życia systemów interaktywnych, istnieje także ogromna ilość proponowanych technik: wskazówek, wzorców, reguł, konwencji, czy standardów projektowania interfejsu. Techniki te dodatkowo można podzielić na zależne i niezależne od platformy, dla której

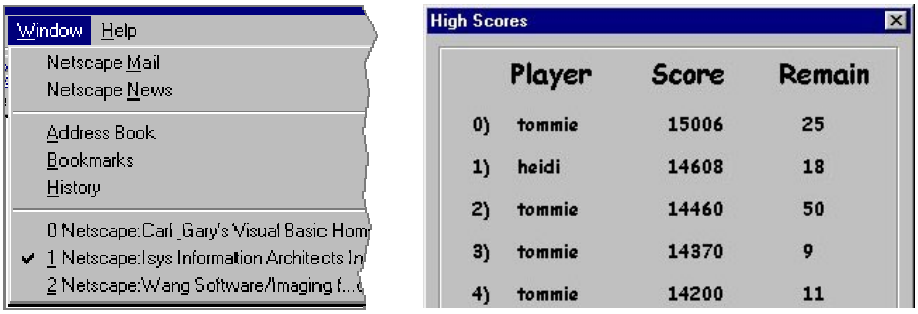
jest przeznaczony projektowany system. Przykładowo, inne wskazówki stosuje się do projektowania interfejsów aplikacji przeznaczonych dla systemu Mac OS X firmy Apple [WWW4], a inne dla aplikacji działających pod systemem Windows firmy Microsoft [Micr95].

Proces projektowania interfejsu jest zawsze procesem kreatywnym, każdy projekt jest inny i w końcowym efekcie otrzymuje się różnorodne interfejsy użytkownika. Dodatkowym problemem jest różnorodność użytkowników oraz ich nieograniczony potencjał używania interfejsu do celów zupełnie nieprzewidzianych przez projektanta. W związku z tym przyjmuje się, że nawet najlepszy zaprojektowany samodzielnie przez projektanta interfejs (z użyciem wszystkich dostępnych wskazówek i zasad projektowania), będzie nieodpowiedni dla użytkownika. Jest to jeden z 11 sloganów użyteczności zdefiniowanych przez Nielsena, które w zwięzłej formie oddają problematykę tworzenia użytecznych interfejsów:

1. Twoje najlepsze pomysły nie będą wystarczająco dobre.
2. Użytkownik zawsze ma rację.
3. Użytkownik nie zawsze ma rację.
4. Projektanci nie są użytkownikami.
5. Użytkownicy nie są projektantami.
6. Wiceprezesi nie są użytkownikami.
7. Mniej znaczy więcej.
8. Szczegóły mają znaczenie.
9. System pomocy nie zawsze jest konieczny.
10. Inżynieria użyteczności jest procesem [Niel93, s. 10-16].

Na pierwszy rzut oka widać, że część wymienionych sloganów jest wzajemnie sprzeczna. Użytkownik ma zawsze rację, bo to on będzie pracował na systemie, ale z drugiej strony często nie zna swoich potrzeb i nie zawsze jest w stanie wyobrazić sobie najlepszego rozwiązania. Podobnie jak projektanci, którzy przez to, że również są użytkownikami systemów komputerowych przenoszą swoje doświadczenia na projektowany interfejs, zapominając, że są zupełnie innym typem użytkownika niż docelowy odbiorca. Wymienione przez Nielsena slogany doczekały się rozwinięcia i można obecnie znaleźć w literaturze, jak i w Internecie ich znacznie większą ilość [WWW3].

Nie tylko projektanci próbują przenosić swoje doświadczenia, robią to również programiści (choć często w sposób nieświadomy). Czasami efekty takich działań wymykają się testerom i można je zaobserwować w finalnych wersjach oprogramowania. Poniżej znajdują się dwa przykłady, które nie są specjalnie uciążliwe dla użytkowników, ale pokazują inny sposób postrzegania świata przez programistów. Zwykli użytkownicy numerują elementy od wartości 1, natomiast programiści w swoich językach programowania odliczają zazwyczaj od 0, co czasami przenoszą na programowane interfejsy.



Rys. 3. Numerowanie od 0 – przenoszenie doświadczeń na interfejs

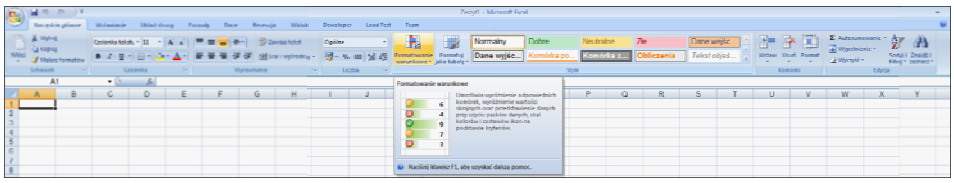
Źródło: Zrzuty ekranów z systemów Netscape Navigator i gry xBlock, za stroną [WWW5].

Większość ze sloganów ma swoje odzwierciedlenie we wskazówkach i zasadach projektowych. Przykładowo jedną ze wskazówek dotyczącą projektowania interfejsu jest zasada czystego i prostego interfejsu, niepozwalającego użytkownikowi na zbytnią swobodę, za to dopasowanego do niego i do jego stylu pracy. Wskazówka ta odzwierciedla slogan „mniej znaczy więcej", czyli jeżeli interfejs ma mniej elementów, za to bardziej dopasowanych do stylu pracy użytkownika, to użyteczność takiego interfejsu będzie większa, niż interfejsu z dużą ilością elementów, które mogą użytkownika wprowadzać w błąd.

Wskazówki projektowe są niezwykle cenne dla projektantów i są jednymi z najczęściej wykorzystywanych narzędzi w procesie projektowania interfejsu. Szczególnie przydatne są przy tworzeniu pierwszych funkcjonalnych prototypów interfejsu. Przykładowymi wskazówkami projektowymi może być lista podstawowych zasad i najlepszych praktyk z zakresu projektowania interfejsu proponowanych przez firmę IBM:

1. Łatwy dostęp do funkcji, które użytkownicy potrzebują przez większość czasu pracy, funkcje mniej potrzebne powinny być umieszczone w mniej widocznym miejscu.
2. Optymalizacja interfejsu pod kątem najczęstszych i najważniejszych z punktu widzenia użytkownika zadań.
3. Obiekty i informacje powinny być widoczne na interfejsie, a dostęp do nich powinien być prosty [WWW13].

Koncepcje zawarte w pierwszych 3 wskazówkach projektowych są realizowane przykładowo przez obiekt wstążki w systemach Office.

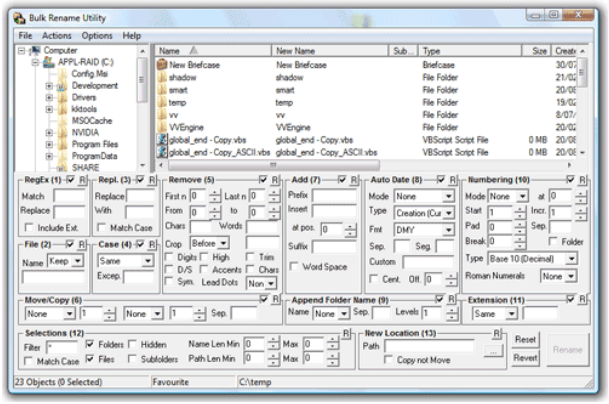


Rys. 4. Wstążka systemu Microsoft Office

Źródło: Microsoft Office 2007.

4. Użycie właściwych wartości domyślnych powoduje kończenie zadań przez użytkownika relatywnie prosto i szybko.
5. Interfejs powinien być elastyczny, tak by umożliwić użytkownikowi dostosowanie go do własnych potrzeb.
6. Użytkownicy powinni być informowani o tym, co w danej chwili robi system (paski postępu oraz stanu).
7. Elementy interfejsu, które wyglądają identycznie, powinny zachowywać się identycznie i dawać takie same rezultaty działania.
8. System powinien umożliwiać użytkownikowi swobodę działania (która umożliwia popełnianie błędów) bez wywoływania strachu, że może on doprowadzić do uszkodzenia danych (przykładowo zastosowanie opcji: cofnij – ponów operację).
9. Aplikacja powinna być przewidywalna dla użytkownika, dlatego powinna używać standardowych konwencji, gdzie to tylko możliwe (np. obsługa skrótów klawiszowych Ctrl-C, Ctrl-V, operacje drag & drop). Użycie powszechnie stosowanych wzorców projektowych sprawi, że użytkownik nie będzie musiał uczyć się interfejsu.
10. Podczas projektowania systemu zawsze należy mieć na uwadze użytkownika końcowego.
11. Należy unikać dodawania zbyt dużej ilości funkcji, ponieważ zwiększają one zakres możliwych działań, co może paraliżować użytkowników.

Poniżej znajduje się zrzut ekranu z systemu pozwalającego na definicję reguł i zmianę nazw wielu plików równocześnie. Hasło reklamowe tego oprogramowania brzmi: „Zmieniaj nazwy wielu plików jednym przyciskiem. Wsadowa zmiana nazw jest prosta.”, jednak interfejs systemu jest tak przeładowany funkcjami, że nie wydaje się być prosty.



Rys. 5. Ekran główny systemu Bulk Rename Utility

Źródło: [WWW2].

12. Interfejs powinien być tak zaprojektowany, by mógł być bez problemu lokalizowany dla innych krajów bez potrzeby jego ponownego projektowania.
13. Podczas projektowania interfejsu należy pamiętać o użytkownikach niepełnosprawnych, którzy być może nie mogą posługiwać się klawiaturą, myszą, czy joystickiem.

Rys. 6. Klawiatura wirtualna

Źródło: System Microsoft Windows.

14. Pomoc kontekstowa powinna być dostępna dla użytkownika wszędzie tam gdzie może jej potrzebować.
15. Interfejs wizualny powinien być intuicyjny i przyjazny.
16. Interfejs powinien być przejrzysty i wizualnie prosty.

    Poniżej została przedstawiona funkcjonalność wyboru par języków, dla których będzie dokonywane tłumaczenie zrealizowane w odmienny sposób. Pierwszy, pomimo że czytelny, zabiera mnóstwo miejsca, drugi jest wizualnie prostszy.

Rys. 7. Parametryzacja par języków, dla których ma być dokonywane tłumaczenie

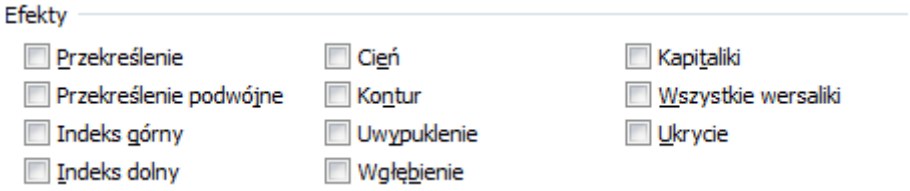
Źródło: Zrzut ekranu z systemu firmy LEC za stroną: [WWW6].

Rys. 8. Wybór języków do tłumaczenia w translatorze Google

Źródło: Zrzut ekranu ze strony.

17. Zasady projektowania interfejsu wizualnego:
    a) projektowanie subtraktywne – redukcja szumu informacyjnego poprzez eliminowanie elementów wizualnych niezwiązanych bezpośrednio z wizualną komunikacją z użytkownikiem;
    b) hierarchia wizualna – zrozumienie ważności zadań użytkownika i ustalenie wizualnej hierarchii tych zadań. Zadania ważniejsze powinny być bardziej widoczne, np. poprzez zmianę położenia, kontrastu, rozmiaru;
    c) afordancja – użytkownik powinien być w stanie w prosty sposób określić jaka czynność może być podjęta z danym obiektem wizualnym. Zasada ta dotyczy nie tylko niestandardowych elementów wizualnych, które mogą być niezrozumiałe dla użytkownika, ale również tradycyjnych elementów, które są już użytkownikowi znane.

Nawet największym firmom tworzącym oprogramowanie, takim jak Microsoft, zdarza się zaprojektować interfejs tak, by łamał zasadę afordancji. Przykładem jest Microsoft Word i okno dialogowe ustawień czcionki, gdzie w efektach użyto pól wyboru sugerujących na możliwość dowolnej kombinacji efektów. W praktyce jednak pola te działają jak pola opcjonalne uniemożliwiając wykonanie nieprawidłowych wyborów typu: kapitaliki i wersaliki równocześnie;



Rys. 9. Okno dialogowe ustawień czcionki

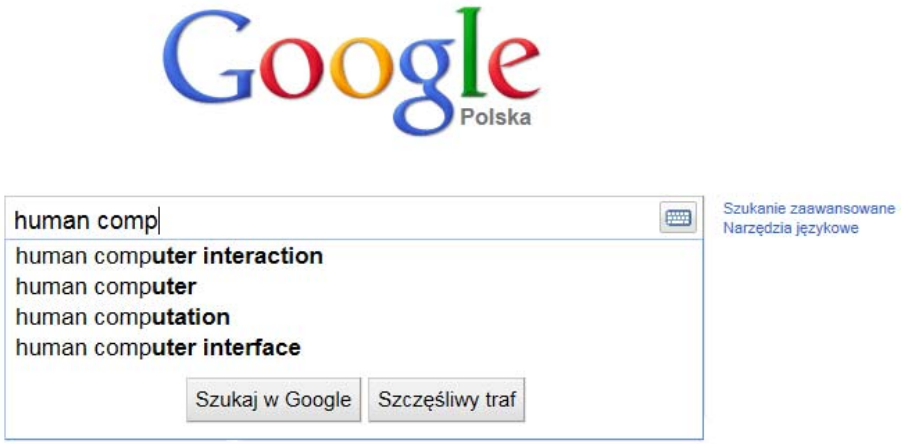
Źródło: System Microsoft Office Word.

d) przestrzeń – brak oszczędzania na miejscu poprzez eliminowanie dodatkowych przestrzeni. Używanie „białych znaków" dających dodatkowe pole manewru dla użytkownika.

Zasad i wskazówek projektowania interfejsu można w literaturze znaleźć o wiele więcej. Dwaj wymienieni wcześniej autorzy – Nielsen i Norman również proponują swoje wskazówki projektowe (jako firma doradcza Nielsen Norman Group) – [WWW1], które w większości pokrywają się z tymi wymienionymi wcześniej. Wśród ich wskazówek projektowych można odnaleźć zasadę mówiącą o nienadużywaniu ilości kolorów. Autorzy zwracają uwagę na zjawisko zwane „ślepotą kolorów". Pewna część populacji (ok. 10% mężczyzn) nie rozróżnia kolorów – różne odmiany daltonizmu, stąd też sugestia o nienadużywaniu kolorystyki. W Internecie można odnaleźć również artykuły obfitujące w praktyczne rady odpowiedniego doboru kolorów, tak by interfejs był czytelny dla większości użytkowników [WWW7].

Wskazówki projektowe są ciągle rozwijane i chętnie wykorzystywane przez projektantów systemów interaktywnych. Innym równie popularnym narzędziem są wzorce projektowe, które w przeciwieństwie do zasad projektowych nie operują na ogólnym poziomie, lecz oferują gotowe rozwiązania (zgodne z wskazówkami projektowymi). Wzorce projektowe powinno się zastosować w okreś-

lonych sytuacjach do realizacji z góry założonych celów projektowych. Przykładem dość powszechnie stosowanego wzorca jest autowypełnianie (*auto-complete*). Stosuje się go w sytuacji, gdy projektant interfejsu wie, że użytkownik będzie musiał wypełniać pewne pole swojego interfejsu wielokrotnie, często wpisując te same wartości, lub wartości te będzie musiał uzyskać w inny sposób. System automatycznego podpowiadania możliwych wartości przy zadanym kontekście w znaczący sposób ułatwia i przyspiesza pracę użytkownika.



Rys. 10. Przykład wykorzystania wzorca projektowego autowypełnianie

Źródło: Wyszukiwarka internetowa firmy Google.

Wzorce projektowe, oprócz samego rozwiązania, opisują również kontekst, w którym nie powinny być stosowane. W przypadku opisywanego tu autowypełniania, nie wolno go stosować w polach wrażliwych (służących np. do wpisywania haseł), w przypadkach gdy ilość dostępnych wartości w danym kontekście jest zbyt duża lub odwrotnie gdy lista wartości jest stała. W tym ostatnim przypadku należy po prostu stosować proste pole wyboru. Innymi znanymi i często spotykanymi wzorcami projektowymi są informacja na żądanie (*details on demand*), czy dynamiczne formularze, których wygląd zmienia się w trakcie wprowadzania informacji przez użytkownika, w zależności od kontekstu.

Podsumowanie

Komunikacja pomiędzy człowiekiem a komputerem od początku odbywała się w głównej mierze poprzez wizualny interfejs. Wraz z rozwojem technologii interfejs stawał się coraz bogatszy, a autorzy systemów zaczęli dysponować szerokim wachlarzem form multimedialnych. Wydajność dzisiejszych urządzeń umożliwia wykorzystywanie chociażby takich technik, jak wirtualna, czy poszerzona rzeczywistość, które stwarzają dodatkowe możliwości dla projektantów i twórców systemów interaktywnych. Również znaczący postęp dokonał się w zakresie sterowania komputerem, gdzie oprócz tradycyjnych kontrolerów, takich jak klawiatura czy mysz, w powszechnym użyciu znalazły się ekrany dotykowe, sterowanie głosem, gestami, czy nawet całym ciałem.

Wydawać by się mogło, że całe to bogactwo technik i narzędzi, którymi dysponują współcześni twórcy oprogramowania wymusza stosowanie równie nowoczesnych technik i zasad projektowania. Okazuje się jednak, że metody opracowane oraz stosowane kilka, czy kilkanaście lat temu są nadal aktualne i doskonale sprawdzają się również dzisiaj. Wynika to z podejścia zorientowanego na użytkownika, a nie na technologię. Wydaje się nawet, że współcześni projektanci jeszcze bardziej skupiają się na problemach użytkowników, dopasowując systemy do dedykowanych, mniejszych grup odbiorców. Przykładowo można spotkać rozwiązania nie tylko przeznaczone dla osób niepełnosprawnych, ale również specjalnie dedykowane dla dzieci czy osób starszych. W szerszym ujęciu podejście zorientowane na człowieka widać w projektach informatycznych stających przed wyzwaniami związanymi z powszechną globalizacją. Zasady projektowe dotyczące lokalizowania aplikacji na potrzeby różnych języków są rozbudowywane o kwestie typowo ludzkie, takie jak zasady kulturowe, religijne, historyczne, czy językowe, często pomijające fizyczne granice państw.

Zorientowanie na użytkownika, wspólne projektowanie, prototypowanie, testy i mierzenie wydajności interfejsu oraz podejście iteracyjne nadal pozostają najlepszymi metodami projektowania systemów interaktywnych.

Literatura

[BeHo98] Beyer H., Holtzblatt K.: *Contextual Design: Defining Customer-centered Systems*. Academic Press, San Diego 1998.

[Boot89] Booth P.A.: An *Introduction to Human-computer Interaction*. Psychology Press, Harlow 1989.

[DeWR08] Dennis A., Wixom B.H., Roth R.M.: *System Analysis and Design*. Wiley, Hoboken 2008.
[DFAB04] Dix A., Finlay J., Abowd G.D., Beale R.: *Human-computer Interaction*. Pearson Education, Harlow 2004.
[Easo76] Eason K.D.: *Understanding the Naive Computer User*. „The Computer Journal" 1976, Vol. 19(1).
[GoLe85] Gould J., Lewis C.: *Designing for Usability: Key Principles and What Designers Think*. „Communications of the ACM" 1985, No. 28(3).
[JoSa10] Joshi A., Sarda N.: *Evaluating Relative Contributions of Various HCI Activities to Usability*. W materiałach konferencyjnych Human-Centred Software Engineering: Third International Conference, HCSE 2010.
[Micr95] *Microsoft Corporation: The Windows Guidelines for Software Designers*. Microsoft Press, Redmond WA 1995.
[Niel93] Nielsen J.: *Usability Engineering*. Academic Press, Boston 1993.
[Norm88] Norman D.: *The Psychology of Everyday Things.* Basic Books, New York 1988.
[SaJB08] Satzinger J.W., Jackson R.B., Burd S.D.: *Systems Analysis and Design in a Changing World*. Cengage Learning EMEA, 2008.
[ShCR09] Shelly G.B., Cashman T.J., Rosenblatt H.S.: *System Analysis and Design*. Cengage Learning, Hampshire 2009.
[SmAt06] Smith-Atakan S.: *Human-computer Interaction*. Cengage Learning EMEA, Hampshire 2006.

Źródła internetowe

[WWW1] http://developer.apple.com/library/mac/#documentation/UserExperience/Conceptual/AppleHIGuidelines/XHIGIntro/XHIGIntro.html [dostęp: 22.11.2012].
[WWW1] http://www.asktog.com/basics/firstPrinciples.html#humanInterfaceObjects [dostęp: 22.11.2012].
[WWW2] http://www.bulkrenamentility.co.uk./screenshots.php [dostęp: 22.11.2012].
[WWW3] http://www.cs.cmu.edu/~bam/uicourse/special/ [dostęp: 22.11.2012].
[WWW5] http://www homepage.mac.com/bradster/iarchitect/clarity.htm [dostęp: 22.11.2012].
[WWW6] http://www.lec.com [dostęp: 22.11.2012].
[WWW7] http://www.lighthouse.org/accessibility/design/accessible-print-design/effective-color-contrast [dostęp: 22.11.2012].
[WWW8] http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2037509/ [dostęp: 22.11.2012].
[WWW9] http://osha.europa.eu/en/legislation/directives/provisions-on-workload-ergonomical-and-psychosocial-risks/osh-directives/5 [dostęp: 22.11.2012].
[WWW10] http://www.translate.google.com [dostęp: 22.11.2012].
[WWW11] http://www.useit.com/alertbox/20000319.html [dostęp: 22.11.2012].
[WWW12] http://www-01.ibm.com/software/ucd/ucd.html [dostęp: 22.11.2012].
[WWW13] http://www-01.ibm.com/software/ucd/designconcepts/designbasics.html [dostęp: 22.11.2012].

HUMAN-COMPUTER INTERACTION

Summary

The paper covers deliberations on methods of designing interactive systems. The first part of paper presents the issue of human-computer interaction. Then describes a user-oriented approach and design models of the interactive systems. The last two parts of the paper focuses on the design of user interface. Starting from the description of the design process, and ending with the principles and design patterns.