ZASADY PROJEKTOWANIA INTERFEJSÓW

Jacek Tomaszczyk
Uniwersytet Śląski
Katowice

System informacyjny, interfejs, zasady projektowania interfejsów, przetwarzanie informacji, użytkownik informacji

Szybki rozwój technologii informacyjnej sprawił, że zautomatyzowane systemy informacyjno-wyszukiwawcze stały się dostępne dla szerszego grona użytkowników. Wcześniej korzystanie z tych systemów było na tyle skomplikowane, że tylko przeszkoleni specjaliści mogli wyszukiwać w nich informacje. Pojawienie się baz danych na dyskach optycznych CD-ROM miało również znaczący wpływ na użytkowników. Powstała nowa kategoria użytkowników, tzw. *użytkownicy końcowi*, czyli faktyczni odbiorcy informacji, którzy do korzystania z baz nie potrzebowali już *pośredników* (osób przeszkolonych w obsłudze systemu, najczęściej bibliotekarzy i pracowników ośrodków informacyjnych). Producenci i projektanci systemów informacyjnych zaczęli starać się, aby z ich systemów mogła korzystać samodzielnie jak największa liczba użytkowników bez jakiegokolwiek wcześniejszego przeszkolenia. To z kolei przyczyniło się do jeszcze większego wzrostu liczby użytkowników, ponieważ – jak wykazały liczne badania – użytkownik woli sam prowadzić wyszukiwania niż korzystać z pomocy pośredników, znacznie precyzyjniej wyrażając swoje potrzeby informacyjne i otrzymując bardziej satysfakcjonujące go wyniki[1].

Producenci systemów informacyjnych, chcąc udostępnić swoje produkty szerokiemu gronu użytkowników, starają się uprościć obsługę systemu, projektując takie interfejsy, których zadaniem jest m.in. ukrycie przed użytkownikiem wszelkich trudności związanych z procesem wyszukiwania informacji i obsługą komputera. W taki sposób powinni postępować nie tylko zawodowi projektanci dużych systemów komercyjnych, lecz również osoby tworzące niewielkie, darmowe systemy baz danych, np. na użytek własny firmy. Szkoły wyższe często same tworzą różnego rodzaju bazy danych i udostępniają je przez Internet. Warto wówczas pomyśleć o tym, aby zaprojektować te systemy w taki sposób, by jak najwięcej użytkowników mogło bez problemów z nich korzystać.

Systemy, w których interfejs użytkownika dobrze spełnia swoje zadania, czyli umożliwia pracę z komputerem osobom bez uprzedniego przeszkolenia i bez wiedzy z zakresu wyszukiwania informacji, są nazwane *systemami przyjaznymi dla użytkownika*[2]. W literaturze przedmiotu pojęcia *interfejs przyjazny dla użytkownika* i *system przyjazny dla użytkownika* są często stosowane

[1] E. Chmielewska-Gorczyca: *O tak zwanych systemach przyjaznych użytkownikowi*. „Zagadnienia Informacji Naukowej" 1990, nr 2, s. 45-82.

[2] Inne, często spotykane określenia takiego systemu to: *zorientowany na użytkownika* (*user-oriented*) i *łatwy w użyciu* (*easy-to-use*).

zamiennie, gdyż na ogół *przyjazność* systemu oceniana jest właśnie na podstawie jego interfejsu[3]. Pojęcie przyjazności systemu nierozerwalnie łączy się z zasadami projektowania interfejsów, bo nie ulega wątpliwości, że interfejsy należy projektować w taki sposób, aby były przyjazne dla użytkownika. Aby zaprojektować dobry interfejs, należy więc najpierw odpowiedzieć na pytanie, jakie cechy powinien posiadać interfejs, aby można go było nazwać przyjaznym. Na to pytanie nie ma jednoznacznej odpowiedzi, a samo słowo „przyjazny" może prowadzić do różnych interpretacji. E. P. Geyser próbuje podać podstawowe cechy przyjazności systemu, cytując i omawiając dziewięć definicji różnych autorów[4]. Zaznacza przy tym, że są to cechy subiektywne i trudne do sformalizowania podczas projektowania systemu oraz zależą od umiejętności i oczekiwań użytkowników. Z powodu licznych definicji i interpretacji terminu „przyjazność" można jedynie podjąć próbę wymienienia najważniejszych cech przyjaznego interfejsu użytkownika. Wymienione zostaną tylko te cechy, które odnoszą się wyłącznie do interfejsów, a więc dotyczące ich wyglądu i sposobu komunikacji z użytkownikiem[5]. Można je podzielić na trzy grupy[6].

1. Fizyczne właściwości sprzętu komputerowego, którym posługuje się użytkownik podczas korzystania z systemu.

2. Inherentne cechy systemu, relatywnie niezależne od subiektywnych ocen użytkownika; są to: niezawodność, sprzężenie zwrotne i obsługa błędów[7].

3. Pozostałe cechy, które czynią system przyjaznym (w sensie funkcjonalnym) dla użytkownika; należą do nich: konsekwencja, dialog, przeźroczystość, łatwość nauczenia się i korzystania z systemu, wsparcie dla użytkownika, system pomocy, prezentacja na ekranie, dane wyjściowe.

Projektowanie interfejsu obejmuje jego wygląd i sposób komunikacji z użytkownikiem, chociaż czasem trudno oddzielić je od możliwości programu. Trzeba zdawać sobie sprawę również z tego, że chociaż w projektowaniu interfejsów nie ma jedynie prawidłowych rozwiązań, znajdujących zastosowanie we wszystkich rodzajach interfejsów, to zasady ich projektowania są dość uniwersalne i pojawiają się najczęściej w literaturze poświęconej temu tematowi. W artykule zostały one omówione w taki sposób, aby korespondowały

---

[3] Należy jednak pamiętać, że system o doskonałym interfejsie nie zawsze można uznać za przyjazny dla użytkownika. Za przykład mogą posłużyć systemy baz danych, w których ogromną rolę odgrywa przede wszystkim ich struktura i zawartość, czyli relewancja, aktualność i kompletność zgromadzonych informacji, a więc cechy niemające nic wspólnego z interfejsem. Nie bez znaczenia jest również dostępność oraz opłata za korzystanie z systemu.

[4] E. P. Geyser: *Indiscriminate use of the term „user friendly" and its shortcomings in the evaluation of information retrieval systems.* „South African Journal of Library and Information Science" 1992 vol. 60 nr 2, s. 80-88.

[5] Stanowią one podzbiór cech przyjaznego systemu, którego pozostałe cechy, takie jak: dostępność, aktualność, kompletność, selektywność i relewancja nie mają wpływu na przyjazność samego interfejsu. Zatem system jako całość, aby był przyjazny dla użytkownika, powinien posiadać wyżej wymienione cechy oraz przyjazny interfejs użytkownika.

[6] M. Próchnicka: *Systemy przyjazne użytkownikowi: rzeczywistość czy utopia?* W: *Użytkownicy informacji elektronicznej.* Red. M. Kocójowa. Kraków 2000, s.119-126. Materiały Edukacyjne Bibliotekoznawstwa i Informacji Naukowej Uniwersytetu Jagiellońskiego, nr 11.

[7] Do tej grupy M. Próchnicka zaliczyła jeszcze przeźroczystość, natomiast nie uwzględniła obsługi błędów. Pojęcie przeźroczystości jest ściśle związane z modelem mentalnym użytkownika, a więc nie można go uznać za niezależne od jego subiektywnej oceny. Oceniając obsługę błędów, badamy, jak system radzi sobie z błędami programowymi oraz błędami popełnianymi przez użytkowników, a więc ta cecha może być w miarę obiektywnie oceniona.

z wyżej wymienionymi cechami przyjaznego interfejsu[8]. Uwzględniono w nich przede wszystkim użytkownika, jego cele, potrzeby, wymagania, umiejętności, predyspozycje i uwarunkowania kognitywne[9].

NIEZAWODNOŚĆ DZIAŁANIA

Niezawodność jest niezmiernie ważną cechą i należy do typowo programistycznej strony projektu, dlatego wielu programistów uważa, że nie wchodzi w proces projektowania interfejsu[10].

Klucz do osiągnięcia niezawodności działania systemu stanowi dokładna analiza techniczna projektu, w której trzeba wziąć pod uwagę następujące czynniki[11]:

- architekturę systemu,
- liczbę dostępów (ile komputerów będzie korzystało z systemu w tym samym czasie),
- liczbę transakcji,
- typ przetwarzania (obliczenia, dane, grafikę, sterowanie),
- charakter dialogu (sposób interakcji),
- wymagania niezawodnościowe, takie jak praca ciągła, praca okazjonalna,
- bezpieczeństwo.

Niezawodność działania należy potwierdzić testami, zanim system zostanie oddany użytkownikom. Wyróżniamy trzy podstawowe rodzaje testów:

1/ testy poprawności – za każdym razem użytkownik musi otrzymywać pożądane i poprawne rezultaty; w teście należy prześledzić działanie programu na przykładzie często wykonywanych operacji, a wyniki sprawdzić – ręcznie lub za pomocą innego systemu, co do którego mamy pewność, że działa poprawnie;

2/ testy wytrzymałości – powinny polegać na używaniu systemu przez dłuższy czas w warunkach, które go maksymalnie obciążają; w tym celu zaleca się[12]:

- zlecenie systemowi wykonywania jakiejś typowej operacji przez cały dzień lub nawet dłużej;
- sprawdzenie, jak program radzi sobie z błędnymi danymi;
- sprawdzenie, czy w przypadku awarii zasilania (sieci, dysku, serwera bazy danych, itp.) system potrafi powrócić do pracy po usunięciu usterki, a dane nie zostaną uszkodzone;
- kontrolę zajmowanej pamięci operacyjnej przy kolejnych uruchomieniach systemu;

---

[8] Prezentowane zasady projektowania skupiają się przede wszystkim na graficznych interfejsach użytkownika, stanowiących obecnie dominującą formę komunikacji użytkownika z komputerem.

[9] Zagadnienia związane z urządzeniami wejścia i wyjścia nie będą omawiane, ponieważ nie wiążą się bezpośrednio z zasadami projektowania interfejsów użytkownika. Warto jednak zaznaczyć, że mają one wpływ na ogólną ocenę przyjazności systemu.

[10] To zagadnienie jest omawiane szczegółowo w obszernej literaturze poświęconej inżynierii oprogramowania, np. A. Jaszkiewicz: *Inżynieria oprogramowania.* Gliwice 1997. M. Flasiński: *Wstęp do analitycznych metod projektowania systemów informatycznych.* Warszawa 1997, P. Coad, E. Yourdon: *Analiza obiektowa.* Warszawa 1994, G. Booch: *Object-Oriented Analysis and Design with Applications.* Redwood City, 1994.

[11] J. Chabik: *Praktyka skutecznego programowania.* Poznań 1999, s. 41.

[12] Tamże, s. 100-101.

3/ testy skalowalności – zaprojektowany system ma spełniać przyszłe wymogi bez dokonywania większych poprawek, dlatego np. system projektowany z myślą o 1000 jednocześnie korzystających z niego użytkowników powinien zostać poddany próbie, w której będzie musiał obsłużyć co najmniej 100.000; testy skalowalności należy przeprowadzać nie tylko dla większej liczby użytkowników, ale także dla większej liczby obiektów i danych niż ta, którą system w danej chwili przetwarza[13].

SPRZĘŻENIE ZWROTNE

Wysyłanie do użytkownika informacji na temat: sposobu interpretacji przez system poleceń wydanych przez użytkownika, wykonywanych w danym momencie czynności, oraz rezultatu zakończenia zadania jest dobrze znane w technologii informacyjnej i teorii kontroli. Trudno sobie wyobrazić pracę z systemem bez sprzężenia zwrotnego. Byłoby to podobne do malowania obrazu pędzlami, które nie pozostawiają na płótnie żadnych śladów[14].

System powinien informować na bieżąco o wszystkich ważnych operacjach, nie naruszając przy tym niepotrzebnie płynności pracy[15]. Informowanie za pomocą okien dialogowych, które wymagają od użytkownika ich zamknięcia, powinno mieć miejsce tylko w szczególnych sytuacjach, kiedy jest to rzeczywiście niezbędne. Mniej ważne informacje mogą być wyświetlane w sposób ciągły w dolnej części ekranu, na tzw. pasku stanu.

Sprzężenie zwrotne generuje komunikaty, które – w zależności od rodzaju – mogą być wyświetlane na ekranie przez określony czas.

- Informacja pozostaje na ekranie do chwili, gdy nie będzie już potrzebna. Na przykład komunikat pojawiający się w chwili próby zapisu na dyskietce, informujący, że w stacji dysków nie ma dyskietki, powinien zniknąć, gdy użytkownik umieści dyskietkę w stacji.
- Komunikat jest widoczny dla użytkownika przez kilka sekund, po czym samoczynnie znika. Na przykład może to być informacja o pomyślnym zakończeniu jakiejś czynności.
- System wyświetla ważny komunikat i czeka, aby go użytkownik potwierdził, naciskając klawisz *Enter* lub klikając myszką odpowiedni przycisk. Na przykład, gdy system nie znalazł wpisanych przez użytkownika słów kluczowych w zaznaczonym zbiorze rekordów, może wyświetlić komunikat następującej treści: *Przeszukiwanie zaznaczonego zbioru rekordów zostało zakończone. Nie odnaleziono poszukiwanych elementów. Czy chcesz przeszukać resztę rekordów?* Taki komunikat zniknie dopiero wtedy, gdy użytkownik naciśnie przycisk *Tak* lub *Nie*.
- Niektóre informacje mogą być tak ważne, że będą wyświetlane na ekranie przez cały czas, stając się stałą częścią interfejsu. Na przykład, gdy za korzystanie z systemu pobierane są opłaty, których wysokość zależy od czasu użytkowania, to interfejs powinien na bieżąco informować użytkownika o czasie i kosztach

[13] Po pierwsze, chodzi o to, czy system w ogóle będzie w stanie wytrzymać takie obciążenie, a po drugie – czy nie wydłuży to czasu oczekiwania na odpowiedź do takiego stopnia, że zostanie poważnie naruszona płynność korzystania z systemu.

[14] D. A. Norman: *The Design of Everyday Things*. New York 1990, s. 27.

[15] Ogólnie chodzi o to, aby system informował, ale nie przeszkadzał. Powinien zachowywać się podobnie do zegarka, który nieustannie odmierza czas, nie wymagając od właściciela potwierdzania każdej upływającej minuty czy godziny.

korzystania z systemu[16]. Należy jednak zwrócić uwagę na to, by interfejs informował użytkownika o czynnościach, które wykonuje system, bez podawania zbędnych szczegółów. Informacje te nie powinny pojawiać się w osobnych oknach, gdyż zmuszałoby to użytkownika do ciągłego ich zamykania.

Sprzężenie zwrotne staje się szczególnie ważne, gdy system pracuje wolno, co zniechęca użytkowników do pracy. W jednej z ankiet w odpowiedziach na pytanie, jakie cechy interfejsów baz danych na CD-ROM najmniej się podobają użytkownikom, najczęściej wymieniano długi czas oczekiwania na reakcję systemu[17]. Czas oczekiwania na odpowiedź (*response time*) jest określany liczbą sekund, jaka upływa od momentu, gdy użytkownik zleci wykonanie czynności (czas mierzony jest od chwili jej zatwierdzenia, zwykle po naciśnięciu klawisza *Enter* lub przycisku myszy) do chwili, gdy system wyniki wykonanego zadania zacznie prezentować na ekranie lub w postaci wydruku[18].

Czas odpowiedzi systemu na akcję użytkownika powinien przyjmować następujące wartości[19]:

- do 0,1 sekundy – system reaguje natychmiast na polecenia, nie potrzebne jest żadne sprzężenie zwrotne;
- między 0,1 a 1 sekundą – użytkownik dostrzega opóźnienie, ale nie wpływa ono na płynność wykonywanego zadania i nie wymaga komunikatów wyjaśniających;
- do 10 sekund – uwaga użytkownika jest wciąż skupiona na dialogu; gdy czas reakcji systemu przekracza tę wartość, niezbędne staje się informowanie użytkownika o trwaniu czynności i o przybliżonym czasie jej zakończenia; dobrym rozwiązaniem jest wówczas wyświetlenie wskaźnika (*progress indicator*) pokazującego postęp systemu w pracy nad wykonywanym zadaniem; w przypadku, gdy nie jest możliwe określenie czasu zakończenia operacji, zamiast wskaźnika postępu interfejs powinien przynajmniej zmienić wygląd kursora myszki lub w inny sposób[20] poinformować użytkownika o pracy systemu nad zakończeniem operacji;

Użytkownikom bardzo zależy na szybkości systemu, a krótki czas odpowiedzi wpływa korzystnie na zwiększenie wydajności i komfortu ich pracy. Nikt nie uzna za przyjazny systemu informacyjnego, choćby z najlepszym interfejsem, gdy czas jego reakcji na działania użytkownika będzie wynosił kilka minut[21]. Czas ten, mimo że jest obiektywnie mierzony w sekundach, może być różnie odbierany przez użytkowników. Czynnikami wpływającymi na jego interpretację są:

---

[16] Informacja ta może być umieszczona na pasku stanu lub byc wkomponowana (np. w postaci licznika) w główne okno aplikacji.

[17] W. H. Perry: *Some lessons for disc and software developers.* „CD-ROM EndUser" vol. 2 nr 1, s. 24-25.

[18] B. Shneiderman: *Designing the User Interface. Strategies for Effective Human-Computer Interaction.* Menlo Park 1992, s. 279.

[19] J. Nielsen: *Usability Engineering.* Boston 1993, s. 135. Czas reakcji systemu jest również ważny podczas przewijania danych, wyświetlanych na monitorze (tzw. *przewijanie ekranu* – ang. *scrolling*).

[20] Na przykład wyświetlając komunikat *„Proszę czekać. Trwa wykonywanie operacji."* lub jakąś animację (np. obracający się wielokąt), która będzie w ten sposób sygnalizować, że system cały czas pracuje. Jest to niezmiernie ważne, gdyż w przeciwnym razie użytkownik mógłby pomyśleć, że system się zawiesił. Dodatkową zaletą takiego rozwiązania jest to, że patrzenie na animację lub wspomniany wskaźnik postępu może zająć uwagę użytkownika i skrócić w ten sposób czas oczekiwania, który – jak wiadomo – jest dla wszystkich irytujący i mało przyjemny.

[21] Na czas reakcji systemu wpływa nie tylko oprogramowanie, ale również sprzęt komputerowy (procesor, pamięć, obciążenie sieci komputerowej).

1/ oczekiwania użytkowników oparte na doświadczeniu w pracy z innym systemem, którego czas reakcji był krótszy; dlatego nawet akceptowalny czas oczekiwania na odpowiedź może nie satysfakcjonować w pełni użytkowników, którzy byli przyzwyczajeni do poprzedniego, szybszego systemu;

2/ typ zadania i predyspozycje indywidualnych użytkowników – początkującym użytkownikom mniej przeszkadza nieznacznie dłuższy czas oczekiwania na odpowiedź niż użytkownikom zaawansowanym; ponadto na satysfakcję poszczególnych użytkowników z szybkości systemu może wpływać wiele czynników: znajomość dziedziny systemu, rodzaj zadania, osobowość użytkownika, jego wiek, nastrój, nastawienie do pracy, porę dnia, czynniki zewnętrzne (np. hałas), koszty oraz presję, związaną z pomyślnym wykonaniem zadania[22];

3/ zmienny sposób pracy użytkowników w zależności od szybkości systemu – w przypadku długiego czasu oczekiwania na odpowiedź użytkownik może dłużej zastanawiać się nad skuteczniejszą strategią wyszukiwania i w ten sposób, mimo wolniejszej pracy systemu, wykonać szybciej zadanie, co wpłynie niewątpliwie korzystnie na jego zadowolenie z interfejsu.

W razie awarii systemu interfejs powinien zapewnić również sprzężenie zwrotne w formie zrozumiałego komunikatu, informującego o jego niesprawności. Niestety, wiele systemów w przypadku awarii po prostu przestaje reagować na polecenia użytkownika. Całkowity brak reakcji systemu, jest najgorszym przypadkiem sprzężenia zwrotnego[23]. Interfejsy powinny radzić sobie z tymi niepożądanymi sytuacjami, reagując właściwie na wszelkie awarie systemu.

OBSŁUGA BŁĘDÓW

Sytuacje, w których pojawiają się błędy, są ważne dla przyjazności systemu z dwóch powodów. Po pierwsze – sygnalizują, że użytkownik ma problemy i nie będzie prawdopodobnie mógł kontynuować pracy z systemem, co uniemożliwi mu dokończenie zadania, a niekiedy może nawet zniszczyć wyniki dotychczasowej pracy. Po drugie – pomagają użytkownikowi zrozumieć lepiej działanie systemu, ponieważ wyświetlane w takich sytuacjach komunikaty przyciągają jego uwagę i zawierają zazwyczaj informacje na temat błędu, przyczyny jego powstania i sposobu usunięcia.

Znane przysłowie mówi: „lepiej zapobiegać niż leczyć", w projektowaniu interfejsów lepiej więc nie pozwalać na popełnienie błędu niż sygnalizować jego wystąpienie. Systemy powinny mieć tak skonstruowany dialog z użytkownikiem, żeby umożliwiał wykonanie wyłącznie poprawnych w danej chwili operacji, które mogą zakończyć się niepowodzeniem tylko z przyczyn obiektywnych. Przestrzeganie tej zasady można dostrzec w dobrze zaprojektowanych interfejsach graficznych, gdzie elementy wizualne (przyciski, ikony, opcje w menu) znikają lub stają się szare w sytuacji, kiedy dane polecenie nie ma sensu.

Nie jest możliwe uniknięcie wszystkich błędów i dlatego konieczne staje się dostarczenie użytkownikowi prostego i zrozumiałego mechanizmu ich usuwania w wydawanych przez niego poleceniach. Najważniejszą częścią tego mechanizmu jest tzw. *komunikat błędu* (*error message*) informujący użytkow-

---

[22] B. Shneiderman: *Designing the User Interface....* op. cit., s. 285.

[23] Niedopuszczalne są sytuacje, w których interfejs wyświetla informacje o rzekomej pracy systemu, a system w rzeczywistości nie reaguje w ogóle na polecenia użytkownika.

nika o nieprawidłowościach w działaniu systemu. Komunikat błędu powinien zawierać następujące elementy:

- numer (kod) błędu[24],
- przyczynę powstania błędu,
- miejsce wystąpienia błędu,
- sposób rozwiązania problemu,
- ewentualne odesłanie do systemu pomocy[25].

Poprawne komunikaty błędu zwiększają wydajność pracy i zadowolenie użytkowników z interfejsu, dlatego powinny:

- przyciągać uwagę użytkownika, nie dopuszczając do sytuacji, w której nie zauważy on ważnego komunikatu;
- być dokładne i zrozumiałe[26];
- opisywać problem, używając słownictwa, którym posługuje się użytkownik;
- używać łagodnego tonu i unikać krytykowania użytkownika[27];
- być konstruktywne i podpowiadać użytkownikowi, co ma w danej sytuacji zrobić[28];
- udostępniać różne stopnie szczegółowości opisu błędu i w razie konieczności odsyłać dodatkowo do systemu pomocy;
- zachowywać konsekwencję w formacie i miejscu wyświetlania na ekranie.

O poprawność obsługi błędów należy zadbać w fazie projektowania, stosując się do poniższych wytycznych[29]:

- trzeba ustanowić *grupę kontroli jakości* składającą się z programistów, użytkowników i specjalistów od interakcji człowieka z komputerem (*human-computer interaction specialists*), której zadaniem będzie zapewnienie skutecznej obsługi błędów;
- wszystkie komunikaty powinny być umieszczone w jednym pliku, który ma powstać już w fazie projektowania;
- w trakcie rozwijania projektu należy systematycznie kontrolować wszystkie komunikaty;
- należy starać się wyeliminować błędy – komunikat o błędzie ma być ostatecznością;
- powinno się regularnie przeprowadzać testy używalności;
- należy zbierać informacje dotyczące sytuacji, w których użytkownicy popełniają najczęściej błędy; te dane mogą być później wykorzystane do poprawy jakości komunikatów, do zmian w systemie pomocy lub też do zmodyfikowania modułu systemu, w którym te błędy się pojawiają.

---

[24] Gdy jest podany numer błędu można sprawdzić w dokumentacji systemu, co jest tego przyczyną i jak należy postąpić w takim przypadku.

[25] J. Chabik: *Praktyka skutecznego programowania*. Poznań 1999, s. 126.

[26] W systemie Windows, gdy próbujemy odczytać dane ze stacji dysków, w której nie ma dyskietki, pojawia się komunikat, mało zrozumiały dla początkujących użytkowników: `A:\ nie jest dostępny. Urządzenie nie jest gotowe.`

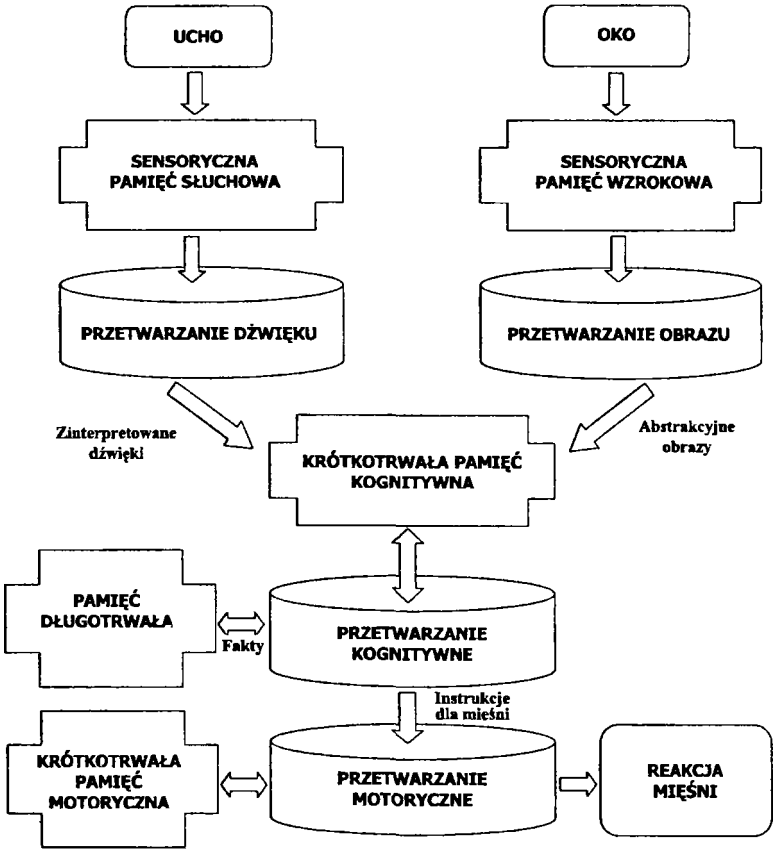
[27] Już samo pojawienie się komunikatu o błędzie stresuje użytkownika, a dodatkowa krytyka ze strony systemu mogłaby sprawić, że w przyszłości bałby się z niego korzystać. Powszechnie uważa się, że informacje pisane wielkimi literami odbierane są przez użytkowników jako krzyk i dlatego należy ich unikać. Zamiast więc komunikatu typu „BRAK DYSKIETKI W NAPĘDZIE A:" lepiej napisać: „Proszę włożyć dyskietkę."

[28] Interfejs może również zaproponować użytkownikowi inne, poprawne operacje, możliwe do wykonania zamiast tej nieprawidłowej, którą wybrał użytkownik.

[29] B. Shneiderman: *Designing the User Interface* ....op. cit., s. 310.

Poprawa jakości komunikatów błędu przynosi ogromne korzyści zarówno początkującym, jak i zaawansowanym użytkownikom. Projektanci, mając na uwadze zadowolenie użytkowników[30], zaczęli zdawać sobie z tego sprawę i dlatego nowe systemy budowane są z uwzględnieniem wyżej opisanych zasad, których stosowanie wpływa na jakość (przyjazność) całego systemu.

Omówione zasady projektowania odnosiły się do właściwych (inherentnych) cech systemu, które można uznać za niezależne od subiektywnych ocen użytkownika. Pozostałe cechy koncentrują się na użytkowniku, a zasady projektowania wykorzystują wiedzę na temat percepcji człowieka (postrzegania kolorów, kształtów i animacji) oraz odwołują się do psychologii kognitywnej. Nie jest możliwe stworzenie interfejsu przyjaznego dla użytkownika bez uwzględnienia takich aspektów jak percepcja, procesy kognitywne, pamięć sensoryczna, pamięć krótkotrwała i długotrwała. Punktem wyjścia jest więc analiza modelu odbioru i interpretacji informacji, który można przedstawić w postaci schematu[31].



Rys. 1. Model przetwarzania informacji

[30] Baza danych traktowana jest jak towar, a jej użytkownik jak klient, dlatego w inżynierii systemów informacyjnych obowiązuje również zasada „klient nasz pan". Widać więc, że żelazne prawa ekonomii docierają wszędzie, nawet do informatyki.

[31] Schemat sporządzono na podstawie modelu A. Sutcliffe'a: *Human-computer interface design*. Hampshire 1988, s. 25.

Kluczową rolę w tym modelu odgrywają różne rodzaje pamięci. Pamięć jest zdolnością organizmu do kodowania, przechowywania i odtwarzania informacji. Procesy pamięciowe nie są zjawiskami jednorodnymi – podlegają zróżnicowaniu w zależności od czasu utrzymywania się śladu pamięciowego oraz od stopnia świadomego zaangażowania się w zapamiętywanie i odtwarzanie informacji. Opierając się na zależnościach czasowych, można wyodrębnić trzy rodzaje pamięci[32]:

1. **Pamięć sensoryczną**, która rozumiana jest jako wstępny etap kodowania napływającej informacji, związany bezpośrednio z percepcją i trwający od kilku milisekund do kilku sekund. W zależności od rodzaju przechowywanego śladu pamięciowego wyróżnia się pamięć wzrokową (*ikoniczną*) i słuchową (*echoiczną*).

2. **Pamięć krótkotrwałą**, określaną również w literaturze jako operacyjną. Charakteryzuje się ona niewielką pojemnością i stosunkowo krótkim okresem przechowywania informacji, wynoszącym od kilku sekund do kilku minut, i wymaga odświeżania co 200 milisekund. W obsłudze systemów informacyjnych odgrywa ogromną rolę. Jej zadaniem jest interpretacja faktów. Podczas kognitywnego przetwarzania napływająca informacja z krótkotrwałej pamięci kognitywnej jest łączona z faktami przechowywanymi w pamięci długotrwałej. W tym procesie następuje również przypisanie znaczenia semantycznego obrazom i dźwiękom, dlatego pamięć krótkotrwała odpowiedzialna jest za identyfikację obiektów, które w przypadku graficznych interfejsów użytkownika są podstawą interakcji. Pojemność pamięci krótkotrwałej jest zwykle ograniczona do 7 (±2) obiektów. Oznacza to, że jednorazowo zapamiętuje się najlepiej informacje, które składają się maksymalnie z siedmiu elementów (np. numery telefonów nie powinny zawierać więcej niż siedem cyfr). Dlatego w interfejsach należy zwrócić uwagę, aby nie przeciążyć użytkownika ilością informacji prezentowanych w tym samym czasie na ekranie. Przeciążenie pamięci krótkotrwałej sprawia, że niedoświadczony użytkownik gubi się w systemie[33]. System pamięci krótkotrwałej człowieka pod względem ważności odpowiada pamięci operacyjnej RAM, bez której – jak wiadomo – komputer nie może funkcjonować, dlatego w projektowaniu interfejsu powinno się mieć na uwadze, że[34]:

- nawet drobne czynniki rozpraszające uwagę użytkownika mogą przyczyniać się do zapominania ostatnio nabytej przez niego wiedzy;
- nierelewantne i wyrwane z kontekstu informacje utrudniają zapamiętywanie;
- podobieństwo zapamiętywanych zdarzeń i obiektów pogarsza późniejsze ich odtworzenie z pamięci;
- zapamiętywanie szczegółów w skomplikowanych obrazach jest bardzo trudne;
- aby ułatwić proces zapamiętywania należy podczas prezentacji łączyć grafikę z tekstem;

---

[32] D. M. Kowalska: *Anatomiczne podstawy pamięci.* W: *Mózg a zachowanie.* Praca zbiorowa pod red. T. Górskiej, A. Grabowskiej, J. Zagrodzkiej. Warszawa 1997, s. 298.

[33] Nie wie, jak wrócić do poprzedniego etapu wykonywanego zadania, zapomina, jaką wybrał wcześniej opcję i nie ma pojęcia, jak należy dalej postępować.

[34] E. P. Geyser: *Human factors in interaction process between man and the user friendly information retrieval system.* „South African Journal of Library and Information Science" 1992 vol. 60 nr 3, s. 167-173.

- najlepiej pamięta się informacje prezentowane na początku i na końcu wydzielonej sekcji (np. rozdziału w podręczniku czy sesji wyszukiwawczej), a najsłabiej te ze środka.

3. **Pamięć długotrwałą**, charakteryzującą się dużą[35] pojemnością i teoretycznie nieograniczonym czasem trwania. Rozumienie i wiązanie (kojarzenie) ze sobą nowych informacji z wcześniej przyswojonymi faktami ułatwia efektywne[36] gromadzenie informacji w pamięci długotrwałej, dlatego użytkownicy muszą zrozumieć istotę działania systemu, aby zdobyć informacje, które ułatwią im korzystanie z systemu po krótkim zapoznaniu się z jego działaniem. Projektując interfejs należy mieć na uwadze dwie główne zasady dotyczące pamięci długotrwałej. Pierwsza z nich zaleca, aby interfejs grupował, porządkował i strukturalizował prezentowane informacje, ponieważ ułatwi to użytkownikowi ich przyswajanie. Proces przetwarzania złożonych informacji polega na rozkładaniu ich na prostsze i łatwiejsze do zapamiętania części, dlatego jest konieczne, aby interfejs pomagał w tym użytkownikowi. Druga zasada podkreśla, iż w procesie uczenia się i zapamiętywania ogromną rolę spełnia logiczność i spójność, która tworzy lepszy kontekst dla nowych informacji. Umożliwia to użytkownikowi kojarzenie faktów z nabytą wcześniej wiedzą, co zwiększa wydatnie szybkość nauki obsługi systemu.

Zanim jednak informacje dotrą do poszczególnych rodzajów pamięci, muszą najpierw zostać odebrane z otoczenia za pomocą zmysłów. Proces świadomej reakcji narządu zmysłowego na bodźce zewnętrzne nosi nazwę percepcji. W graficznych interfejsach użytkownika percepcja jest niezmiernie ważna, gdyż interakcja z systemem odbywa się za pomocą obiektów graficznych, które muszą być dobrze widoczne dla użytkownika i układać się w logiczną całość[37]. Formami percepcji, z których użytkownik korzysta podczas pracy z interfejsami graficznymi, są:

- **percepcja ruchu** – wykrywanie ruchu w otaczającym świecie; informacje o ruchu mogą być dwojakiego rodzaju: gdy obrazy przedmiotów przesuwają się po siatkówce oka lub gdy oczy poruszają się, śledząc ruchome obiekty; percepcja ruchu zależy od poprzednich doświadczeń, np. jeśli nieruchomy punkt zostanie umieszczony na poruszającym się ekranie, użytkownikowi będzie się wydawać, że to właśnie punkt się porusza, a ekran pozostaje nieruchomy; wrażenie to wynika najprawdopodobniej z tego, że w codziennym życiu małe obiekty poruszają się na ogół na większym tle, a nie odwrotnie; mózg, dokonując oceny docierającej do niego informacji, za prawdziwą przyjmuje tę bardziej prawdopodobną interpretację zdarzeń;
- **percepcja kształtu** – jeden z podstawowych elementów orientacji człowieka w świecie; ten skomplikowany proces polega na organizacji różnorodnych informacji w pewne całości, identyfikowane następnie jako przedmioty (obiekty), mające określone znaczenie oraz nazwę[38]; ma to zasadniczy wpływ na percepcję i decyduje w znacznym stopniu o tym, co jest napraw-

[35] Nie wiadomo dokładnie, jaka jest pojemność pamięci. Szacuje się, że mózg człowieka składa się z biliona komórek, a dodatkowo istnieją połączenia międzykomórkowe, mogące tworzyć gigantyczną liczbę konfiguracji i permutacji.

[36] Skuteczne gromadzenie (kodowanie) informacji w pamięci stanowi istotę szybkiego odtwarzania (przypominania sobie) zapamiętanych danych.

[37] W interfejsach, których zasada interakcji oparta jest na języku poleceń, percepcja nie odgrywa aż tak wielkiej roli jak w przypadku interfejsów graficznych.

[38] Dotyczy to również wyodrębniania kształtów z tła.

dę postrzegane; proces ten kończy się najczęściej identyfikacją obiektu, czyli przypisaniem go do określonej klasy obiektów znanych użytkownikowi z uprzednich doświadczeń, mających określone cechy i spełniających podobne funkcje;

- **percepcja odległości i głębi** – pomimo że obraz powstający w oku, a dokładnie na siatkówce, jest płaski, to otaczająca rzeczywistość postrzegana jest jako trójwymiarowa dzięki tzw. zjawisku stereoskopowemu[39]; ponieważ oczy (z powodu ich rozsunięcia w płaszczyźnie poziomej) patrzą na przedmioty pod nieco innym kątem, na siatkówkach powstają obrazy, które są przesunięte względem siebie, co pozwala na uzyskanie efektu trójwymiarowości;
- **widzenie barw** nie tylko wzbogaca percepcję, dostarczając wielu wrażeń estetycznych, ale ma często decydujące znaczenie dla rozpoznawania obiektów; w interfejsach graficznych kolor odgrywa ogromną rolę; dobór odpowiednich kolorów pozwala uzyskać efekt głębi, mimo że ekran monitora jest płaski; kolorystyka wpływa również na komfort pracy – źle dobrane barwy męczą nadmiernie oczy i mogą drażnić użytkowników[40]; w zależności od koloru komórki nerwowe siatkówki są pobudzane albo hamowane – dzięki temu można uzyskać efekt kontrastu, aby przyciągnąć uwagę użytkowników na ważne obszary interfejsu (kontrastujące ze sobą barwy to np. czerwona i zielona oraz niebieska i żółta)[41];
- **percepcja słuchowa** – jak na razie jest najmniej wykorzystywana w interfejsach graficznych[42], ale być może w przyszłości będzie stanowić podstawę interakcji człowieka z komputerem, gdy zostaną zaimplementowane interfejsy języka naturalnego mówionego.

Powszechny pogląd głoszący, że doznania percepcyjne stanowią dokładne odzwierciedlenie świata zewnętrznego i powstają niemal automatycznie, nie jest do końca prawdziwy, ponieważ percepcja jedynie częściowo wynika z pobudzenia układów sensorycznych. Złożone doznania ruchu, barwy, dźwięku czy dotyku nie są wyłącznie prostym zapisem zjawisk zewnętrznych. Percepcja jest więc bardziej procesem tworzenia niż odtwarzania, gdyż postrzegany obraz rzeczywistości nie stanowi jej bezpośredniego odzwierciedlenia, lecz raczej jej interpretację, która powstaje w wyniku ciągłego formułowania hipotez, weryfikowanych następnie na podstawie nowych informacji[43].

Z powyższego wynika, że zdolność użytkownika do wykorzystywania w procesie percepcji (podczas pracy z interfejsem) wcześniejszych doświadczeń[44] umożliwia mu prawidłową interpretację zdarzeń oraz podejmowanie

---

[39] Zjawisko stereoskopii poznano już w XIX wieku.

[40] Można sobie nie zdawać z tego sprawy, że stan rozdrażnienia spowodowany jest właśnie złym zestawieniem kolorów.

[41] Znaczenie poszczególnych kolorow, a także sposoby uzyskiwania efektu trójwymiarowego zostaną opisane w dalszej części artykułu.

[42] Interfejsy generują jedynie krótkie dźwięki, informujące np. o wystąpieniu błędu lub o zakończeniu operacji.

[43] A. Grabowska: *Percepcja wzrokowa i jej analogie do innych form percepcji*. W: *Mózg a zachowanie*. Praca zbiorowa pod red. T. Górskiej, A. Grabowskiej, J. Zagrodzkiej. Warszawa 1997, s. 147 i n.

[44] Nie chodzi tutaj wyłącznie o wcześniejsze doświadczenia w posługiwaniu się danym interfejsem, lecz o wiedzę ogólną, przeżycia i doświadczenia. Stąd też bierze się m.in. fakt, że nie można wszystkich użytkowników traktować jednakowo i niezbędne staje się uczynienie interfejsów adaptowalnymi do indywidualnych potrzeb użytkownika.

decyzji i działań adekwatnych do sytuacji, w jakiej się znajduje, czyli wpływa istotnie na efektywne korzystanie z systemu.

Percepcja i pamięć wykorzystywane są w procesach kognitywnych, które obejmują: uczenie się, zapamiętywanie, rozumienie, formułowanie i rozwiązywanie problemów, podejmowanie decyzji, koncentrację i uwagę. W nauce o informacji badania kognitywne prowadzone są niemal we wszystkich jej obszarach, ale głównym polem odniesień pozostają użytkownicy. Badania użytkowników koncentrują się na problemach powstawania potrzeb informacyjnych, przetwarzania pytań przez ludzi, wzorach negocjacji, interfejsach i wykorzystywaniu informacji[45].

Istotną zasadą wynikającą z badań kognitywnych, którą należy stosować w projektowania interfejsów, jest konieczność uwzględniania różnych poziomów wiedzy i umiejętności użytkowników w zakresie posługiwania się techniką informacyjną. Można to osiągnąć projektując interfejs tak, aby dostosowywał się on do użytkownika, pozwalając mu zdobyć w łatwy sposób wiedzę o systemie, która jest wystarczająca do realizacji zadań zaspokajających jego potrzeby informacyjne.

W projektowaniu interfejsów wykorzystuje się znane z psychologii kognitywnej pojęcie modelu mentalnego i modelu konceptualnego.

Model mentalny, tworzony w umyśle użytkownika, umożliwia mu przewidywanie rezultatów podejmowanych przez niego akcji i upraszcza ogromnie proces pojmowania, klasyfikując i grupując obiekty oraz zdarzenia[46]. W rezultacie, działając podświadomie, użytkownik koncentruje się tylko na ważnych i nieznanych mu wcześniej sytuacjach, a czynnościom rutynowym poświęca niewiele uwagi. W ten sposób może całkowicie skupić się na istocie zadania.

Bez dobrego modelu mentalnego użytkownik nie jest w stanie wykorzystać w pełni możliwości systemu, ponieważ postępując jak gdyby „na pamięć", nie będzie wiedział, co zrobić w przypadku pojawienia się błędu lub jakie podjąć czynności w sytuacji, z którą nie miał wcześniej do czynienia. Aby sobie z tym poradzić, niezbędne jest lepsze zrozumienie systemu, czyli poprawny model mentalny, dzięki któremu użytkownik potrafi wyjaśnić działanie systemu. Model mentalny użytkownika wcale nie musi dokładnie odzwierciedlać rzeczywistego działania systemu, ale powinien łączyć obiekty i czynności w spójną całość tak, aby użytkownik czuł, że kontroluje system, rozumiał zastosowanie poszczególnych opcji i funkcji, przewidując rezultaty ich zastosowania, oraz potrafił radzić sobie z błędami i nowymi sytuacjami[47].

Model konceptualny, tworzony w umyśle projektanta interfejsu, odzwierciedla rzeczywistą strukturę systemu i zasady jego funkcjonowania. Model konceptualny powinien być zrozumiały, spójny i funkcjonalny, aby się pokrywał z modelem mentalnym użytkownika[48]. Odpowiedzialni za to są projektan-

---

[45] J. Woźniak: *Kategoryzacja*. Warszawa 2000, s. 48

[46] Na model mentalny mają wpływ oczekiwania użytkowników, ich dotychczasowe doświadczenia w korzystaniu z systemów informacyjno-wyszukiwawczych, szkolenia oraz wyobraźnia i wiedza ogólna.

[47] Jako przykład można podać model mentalny związany z kierowaniem samochodem. Gdy obraca się kierownicą w lewo, samochód skręca w lewo. Nie jest ważna przy tym budowa i dokładne działanie całego układu kierowniczego. Kierowca skręcając nawet nie myśli o tym, że tak naprawdę, to skręcają tylko przednie koła pojazdu. Najważniejsze jest natomiast to, że zachowanie samochodu jest przewidywalne i zgodne z wyobrażeniami (modelem mentalnym) osoby kierującej.

[48] W sytuacji, w której model mentalny i konceptualny pokrywają się idealnie, użytkownik nie potrzebuje żadnej pomocy ani wyjaśnień podczas pracy z systemem.

ci, którzy powinni dążyć do zgodności tych dwóch modeli, aby w ten sposób skrócić użytkownikom czas nauki obsługi interfejsu i ułatwić efektywne korzystanie z systemu.

Z modelem mentalnym łączą się dwa pojęcia: luka w realizacji (*gulf of execution*) i luka w ocenie (*gulf of evaluation*), które wyrażają rozbieżność między modelem mentalnym użytkownika a rzeczywistym funkcjonowaniem systemu[49]. Stanowią one jeden z głównych problemów, jakie napotykają użytkownicy, którzy po raz pierwszy korzystają z danego interfejsu.

Luka w realizacji jest to różnica między tym, co zamierza osiągnąć użytkownik, a możliwościami interfejsu. Badając ten aspekt konstrukcji systemu należy odpowiedzieć sobie na pytanie, czy użytkownik może w bezpośredni i niezbyt skomplikowany sposób wykonać zadanie, wykorzystując dostępne w interfejsie opcje i funkcje.

Luka w ocenie określa wysiłek, jaki musi włożyć użytkownik, aby zinterpretować reakcje systemu oraz ocenić, w jakim stopniu zostały zrealizowane jego zamierzenia i oczekiwania. Problem ten zmniejsza się, gdy system informuje użytkownika o postępie w wykonywanych czynnościach i o wyniku ich zakończenia w sposób łatwy do zauważenia i zinterpretowania, oraz gdy interfejs działa zgodnie z przewidywaniami użytkownika.

Wyżej omówione zagadnienia, związane z pamięcią, percepcją oraz procesami kognitywnymi, są ogromnie ważne w projektowaniu systemów i będą wielokrotnie uwzględniane podczas rozważania kolejnej grupy zasad projektowania przyjaznych interfejsów, które w centrum zainteresowania stawiają użytkownika.

KONSEKWENCJA

Konsekwencja nie jest wyłącznie kwestią układu graficznego interfejsu, lecz dotyczy także spójności i logiczności wykonywanego zadania oraz struktury funkcjonalnej systemu. Zachowanie konsekwencji interfejsu przyczynia się do łatwiejszej obsługi systemu, skraca czas szkolenia użytkowników[50] i zwiększa wydajność ich pracy. Należy zwrócić uwagę na fakt, że konsekwentna interpretacja przez system działań użytkowników powinna być ważniejsza niż konsekwencja działania samego systemu.

Najlepszym sposobem na zaprojektowanie spójnego i logicznego interfejsu jest stosowanie się do obowiązujących standardów. Specjalna grupa, zajmująca się technologią CD-ROM (*Special Interest Group on CDROM Applications and Technologies – SIGCAT*), powołała komitet (*the CDROM Consistent Interface Committee – CDCINC*), którego zadaniem było opracowanie standardu dla aplikacji na CD-ROM. Prace komitetu zakończyły się opublikowaniem raportu zawierającego 13 postulatów, które powinni uwzględnić projektanci systemów. Te postulaty to propozycje podstawowych funkcji każdego systemu informacyjno-wyszukiwawczego[51].

---

[49] D. A. Norman: *The Design of....* op. cit., s. 49-51.

[50] Jak wykazały badania, czas szkolenia użytkowników skraca się wówczas o 25-50% – P.G. Polson: *The consequences of consistent and inconsistent user interfaces.* W: *Cognitive Science and its Applications for Human-Computer Interaction.* Red. R. Guindon. Hillsdale 1988, s. 59-108.

[51] CD-ROM Consistent Interface Committee: *CD-ROM Consistent Interface Guidelines: A Final Report.* „CD-ROM Librarian" 1992 nr 7, s. 18-29.

1. **Pomoc** (*Help*)[52] – powinna występować zarówno w formie elektronicznej (wyświetlana na ekranie), jak i drukowanej (dokumentacja, podręcznik). Musi być dostępna w każdej chwili, niezależnie od rodzaju i etapu wykonywanego zadania. W żadnym wypadku nie może zniszczyć danych wprowadzonych wcześniej przez użytkownika. Pomoc może spełniać następujące funkcje:

- podpowiadać użytkownikowi, jakie czynności może wykonać w danym momencie (tzw. pomoc kontekstowa);
- doradzać rozwiązanie problemów, na jakie natknął się użytkownik;
- wyświetlać indeks lub spis treści wszystkich rodzajów pomocy, z których użytkownik może wybrać to, co jest mu potrzebne.

2. **Indeks** (*Browse Index*) – powinien umożliwiać przeglądanie słów kluczowych z całej bazy danych (nazwiska autorów, tytuły czasopism lub słowa pojawiające się w treści dokumentów w bazach pełnotekstowych). Indeks może być wyświetlany w osobnym oknie lub zajmować główny ekran. Użytkownik powinien mieć możliwość wyboru słowa bezpośrednio z indeksu lub przez wpisanie go z klawiatury.

3. **Wyszukiwanie** (*Search*) – nie zostało ujęte w raporcie w konkretne zasady. Standard nie określa typów indeksów, metod wyszukiwania[53] ani sposobów formułowania zapytań informacyjnych (np. operatory algebry Boole'a). Zwraca się tylko uwagę, że termin *Search* jest najbardziej odpowiednim określeniem tej funkcji, gdyż inne terminy, jak *Find*, *Select*, *Get*, *Retrieve*, *Fetch*, *Lookup* i *Query*, mogą być mylące dla użytkowników.

4. **Prezentacja** (*Display*) – rozumiana jako wyświetlanie na ekranie informacji dla użytkownika, np. menu, spis treści, całość lub część rekordu bibliograficznego, dokument pełnotekstowy, historia wyszukiwania, grafika, inne informacje związane z systemem bazy danych. Ponieważ są to problemy dotyczące głównie projektowania graficznego, nie zaproponowano żadnych wytycznych ani metod implementacji tej funkcji.

5. **Drukowanie** (*Print*) – podstawowa funkcja zachowywania wyników sesji wyszukiwawczej.

6. **Zapisywanie** (*Download*) – funkcja pozwalająca zapisać wyniki wyszukiwania na nośniku elektronicznym, np. dyskietce, twardym dysku lub innym komputerze. Termin *Download* uważa się za najbardziej poprawny. Inne, często stosowane określenia, mogą być mniej zrozumiałe ze względu na niejednoznaczność, np. *Save* stosuje się również do zapisu historii wyszukiwawczej, *Export* – wprowadza w błąd sugerując, że chodzi o zmianę formatu zapisywanych danych, *Print-to-disk* – mylące określenie, kojarzące się z drukowaniem.

7. **Zacznij ponownie** (*Restart*[54]) – funkcja, która powinna umożliwiać użytkownikowi rozpoczęcie nowej sesji wyszukiwawczej (wyświetlając główne okno systemu) bez potrzeby zakończenia aplikacji i ponownego jej uruchamiania. Zwalnia to również użytkownika od wielokrotnego naciskania klawisza *Escape*, anulującego jedną akcję lub wracającego do poprzedniego menu.

---

[52] W nawiasach podaję angielskie nazwy funkcji.

[53] Mogą to być np. listy proste, listy inwersyjne, pliki z funkcją mieszającą (tzw. *haszującą*), drzewa poszukiwań binarnych.(*Binary Search Trees*). Standard nie narzuca żadnej z wymienionych metod, a wybór zależy wyłącznie od implementacji.

[54] Nie ma to nic wspólnego z restartem komputera, polegającym na jego ponownym uruchomieniu.

8. **Zmień** (*Change*) – ta funkcja powinna obejmować trzy akcje:

- zmianę dysku w obrębie tej samej bazy danych[55] bez spowodowania strat w instrukcjach wyszukiwawczych wpisanych przez użytkownika, ani w otrzymanych wynikach poprzedniego wyszukiwania;
- zmianę bazy danych znajdującej się na tym samym dysku[56] przy zachowaniu wszystkich wcześniejszych kwerend;
- zmianę bazy danych znajdującej się na innym dysku[57] bez potrzeby ponownego uruchamiania aplikacji.

9. **Wyjście** (*Quit*) – funkcja, której zadaniem jest zakończenie działania aplikacji. Powinna być dostępna z każdego miejsca w systemie. Inne nazwy tej funkcji to: *Exit*, *Stop*, *End*, *Bye*, *Logoff*, *Logout*, *Off*, *Disconnect*.

10. **Zacznij** (*Execute*) – funkcja wydająca systemowi polecenie rozpoczęcia wyszukiwania. Można ją uruchomić naciskając klawisz *Enter*, przycisk myszy lub literę będącą skrótem do opcji w menu. Oprócz nazwy *Execute* występują dość często *Begin* i *Start*, a czasami *Initiate* i *Transmit*.

11. **Przerwij** (*Break*) – funkcja, która powinna przerwać proces wyszukiwania. Umożliwia to użytkownikowi rezygnację z operacji, która trwa zbyt długo. Przerwanie wykonywanej akcji nie może być przyczyną utraty danych, a system musi powrócić do poprzedniego stanu. Standardowe klawiatury umożliwiają zatrzymanie procesu kombinacją klawiszy CTRL+Break.

12. **Wróć** (*Escape*) – użytkownik musi zawsze mieć możliwość powrotu do jednego (dowolnego) z poprzednich etapów procesu wyszukiwania. Zazwyczaj umożliwia to przycisk Esc na klawiaturze[58]. Innymi określeniami tej funkcji są *Cancel*, *Back*, *Backup*.

13. **Nawigacja** (*Navigation*) – poruszanie się po bazie danych lub zbiorze rekordów. Raport CD-CINC proponuje 10 podstawowych sposobów poruszania się po systemie, które mają ułatwić pracę użytkownikowi[59]:

- przesuwanie się w dół po tekście linijka po linijce;
- przesuwanie się w górę po tekście linijka po linijce;
- przejście o jeden ekran w dół;
- przejście o jeden ekran w górę;
- przejście do następnego rekordu w zbiorze[60];
- przejście do poprzedniego rekordu w zbiorze;
- przejście do następnego obiektu (np. menu lub rekordu);
- przejście do poprzedniego obiektu;
- przejście do danego miejsca w aplikacji;
- poziome poruszanie się po obiektach interfejsu (polach, oknach, menu).

---

[55] Duże bazy danych mogą zajmować kilka dysków CD-ROM.

[56] Na jednym dysku może znajdować się kilka baz.

[57] Taka sytuacja ma miejsce, gdy na różnych dyskach znajduje się wiele baz, które są obsługiwane przez jeden system zarządzania bazą danych.

[58] Każde kolejne naciśnięcie klawisza *Esc* cofa o jeden etap.

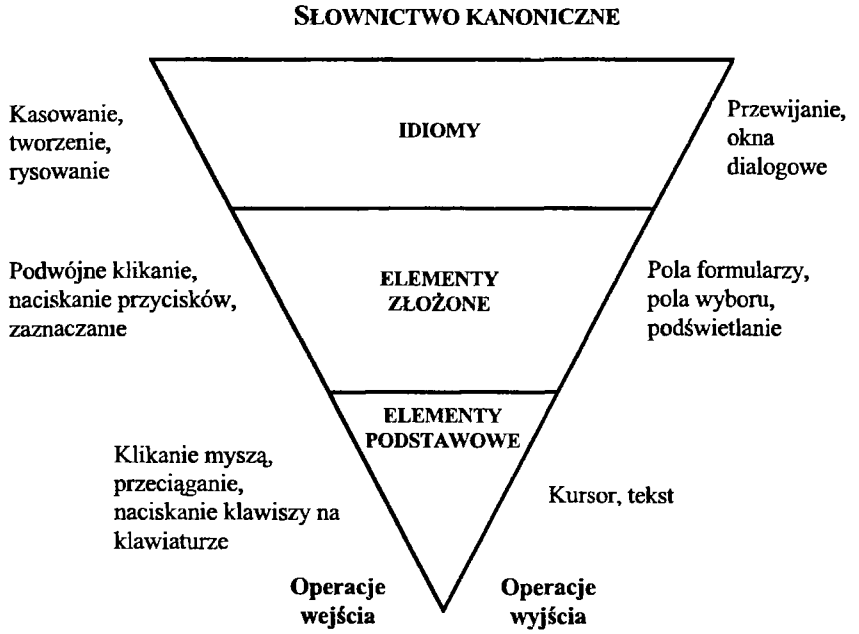
[59] Lista nie zawiera wszystkich możliwych sposobów przemieszczania się, lecz uwzględnia tylko najbardziej potrzebne i niezbędne do swobodnego korzystania z bazy.

[60] Chodzi o zbiór rekordów, wyświetlanych jako odpowiedź na zadane przez użytkownika pytanie (rekordów, które spełniają kryteria zawarte w instrukcji wyszukiwawczej). Zbiór ten nazywany jest również listą trafień (*hitlist*).

DIALOG Z UŻYTKOWNIKIEM

Jest to najważniejsza cecha interfejsu, wpływająca na ocenę systemu przez użytkownika[61]. Dobry dialog to podstawa efektywnej i przyjemnej pracy użytkowników z systemem informacyjno-wyszukiwawczym. Uważa się, że interfejsy graficzne odniosły natychmiastowy sukces, ponieważ umożliwiły użytkownikom prowadzenie dialogu z systemem za pomocą bardzo ograniczonego słownictwa oraz wizualnej interakcji, zwalniając ich od uczenia się nazw wielu komend i parametrów. Dialog w tego typu interfejsach opiera się jedynie na trzech podstawowych akcjach: kliknięciu przyciskiem myszy, podwójnym kliknięciu oraz kliknięciu i równoczesnym przemieszczaniu myszy (przeciąganie obiektów przy wciśniętym przycisku myszy). Klawiaturę wykorzystuje się głównie do wprowadzania instrukcji wyszukiwawczych i skrótów komend. Alan Cooper, projektant popularnego języka programowania Visual Basic twierdzi, że sukces interfejsów graficznych leży właśnie w ograniczonym do niezbędnego minimum słownictwie wykorzystywanym w dialogu z systemem (wspomniane trzy akcje), które nazywa słownictwem kanonicznym (*canonical vocabulary*). Według niego powinno ono przyjmować kształt odwróconej piramidy, a projektanci, którzy dążą do uproszczenia obsługi systemu, muszą przestrzegać tej struktury[62].

SŁOWNICTWO KANONICZNE



Rys. 2. Struktura słownictwa kanonicznego

[61] Z dialogiem na ogół ściśle łączy się przeźroczystość systemu, dlatego zostanie tutaj uwzględniona, mimo że podczas prezentowania cech przyjaznego interfejsu była oddzielnie opisana.

[62] A. Cooper: About Face. *The Essentials of User Interface Design*. Foster City 1995, s.47-49.

Dolna część piramidy zawiera podstawowe elementy atomowe, które nie podlegają dekompozycji. Z tych elementów zbudowany jest cały język, za pomocą którego użytkownik prowadzi dialog z systemem. Liczba elementów podstawowych języka nie powinna przekraczać czterech.

Elementy złożone, zajmujące środkową część piramidy, powstają wyłącznie w wyniku połączenia jednego lub dwóch elementów podstawowych.

Najwyższa część piramidy zawiera najbardziej zaawansowane struktury języka. A. Cooper nazywa je idiomami i definiuje jako połączenie elementów złożonych z wiedzą na temat rozwiązywanego problemu, znaną jako domena (dziedzina) systemu. W interfejsie graficznym mogą to być np. paski tytułu, okna dialogowe i ikony.

Aby dobrze zaprojektować dialog, należy uważnie przeanalizować jego etapy. Donald Norman przedstawia model interakcji człowieka z komputerem w 7 etapach[63]:

- Sformułowanie celu.
- Zamiar osiągnięcia celu.
- Wybór akcji (funkcji).
- Wykonanie zadania.
- Obserwacja stanu systemu.
- Interpretacja stanu systemu.
- Ocena rezultatów.

Na podstawie powyższych etapów D. Norman zaproponował zasady dobrego projektowania, które obejmują zagadnienia związane z widocznością, modelami, mappingiem i sprzężeniem zwrotnym[64].

Korelacja modeli

Projektanci interfejsów powinni dążyć do tego, aby tworzone przez nich modele konceptualne korespondowały ściśle z modelami mentalnymi użytkowników. Użytkownik i projektant najczęściej porozumiewają się ze sobą tylko przez tzw. obraz systemu (*system image*), czyli wygląd interfejsu, jego działanie, interakcję oraz podręczniki i instrukcje do niego dołączone. Projektant musi więc tak zaprojektować interfejs, aby użytkownik wyłącznie na podstawie obrazu systemu mógł stworzyć model mentalny, który będzie zbliżony do modelu konceptualnego. Innymi słowy, zadaniem projektanta jest zaproponowanie funkcjonalnego i spójnego modelu konceptualnego tworzącego właściwy obraz systemu, na podstawie którego powstanie model mentalny użytkownika umożliwiający mu sprawne posługiwanie się systemem. Należy podkreślić, że użytkownik całą wiedzę o systemie czerpie wyłącznie z obrazu systemu.

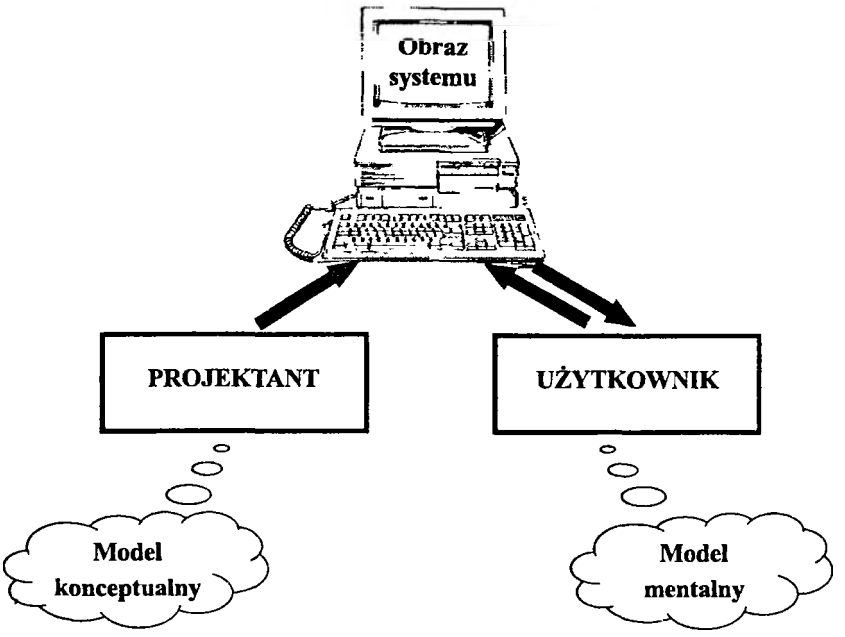
Rysunek 3. ilustruje zależność między modelami a obrazem systemu[65].

---

[63] D. A. Norman: *The Design* ....op. cit., s. 48.

[64] W dialogu znajdują głównie zastosowanie zasady dotyczące modeli i mappingu. Sprzężenie zwrotne zostało już opisane, a o widoczności będzie mowa przy omawianiu prezentacji informacji.

[65] Tamże, s. 190.



Rys. 3. Związek między modelem konceptualnym, modelem mentalnym i obrazem systemu

Mapping

Termin ten oznacza związek między dwiema rzeczami. W przypadku interfejsów rozumie się go jako związek między: działaniami a rezultatem tych działań, funkcjami a efektem ich zastosowania, rzeczywistym stanem systemu a tym, co widzi użytkownik. Poprawny mapping pomaga stworzyć właściwy model mentalny, stanowi podstawę *przeźroczystości* systemu, czyniąc go przewidywalnym, oraz zmniejsza luki w realizacji i luki w ocenie, przyczyniając się do lepszego zrozumienia jego działania.

Mapping jest również odpowiedzialny za to, aby każdy komponent interfejsu mógł zaprezentować się użytkownikowi w sposób widoczny i zrozumiały, tzn. pozwalał przewidzieć jego zastosowanie[66]. Można to osiągnąć stosując **metafory i idiomy**.

Metafora, rozumiana jako figura stylistyczna, *jest to wyrażenie, w którym przynajmniej jeden wyraz uzyskuje inne, obrazowe, ale pokrewne znaczenie*[67]. Przykładami mogą być: „*głęboka* myśl", „*ostra* wymiana zdań", „sportowiec *szlifuje* formę". W projektowaniu interfejsów stosuje się metafory wizualne. Są to rysunki, które reprezentują opcje i funkcje interfejsu, wykorzystując analogię między rzeczą, którą przedstawia rysunek, a jej wykorzystaniem w rzeczywistym świecie. Metaforą jest np. ikona z symbolem nożyczek ✂ , służąca do wycinania zaznaczonego tekstu. W realnym świecie nożyczki służą właśnie do wycinania, więc w interfejsie funkcja zobrazowana za pomocą ikony nożyczek będzie łatwo skojarzona z czynnością wycinania.

---

[66] D. Collins: *Designing Object-Oriented User Interfaces*. Redwood City 1995, s. 105.
[67] W. Kopaliński: *Słownik wyrazów obcych i zwrotów obcojęzycznych*. Warszawa 1989, s. 329.

Użytkownicy rozumieją metafory intuicyjnie. Pojmują ich znaczenie, ponieważ łączą je mentalnie z obrazami i czynnościami, których się już kiedyś nauczyli. Metafory mogą jednak prowadzić do nieporozumień, ponieważ zależą od skojarzeń projektanta systemu i użytkownika. Ponadto bardzo trudno jest znaleźć w środowisku elektronicznym dobrą metaforę, która oddaje w pełni znaczenie obiektu, a przede wszystkim czynności ze świata rzeczywistego[68]. Niepełne metafory ułatwiają nieznacznie pracę użytkownikom, którzy po raz pierwszy używają interfejsu, i sprawiają im wiele problemów w trakcie dłuższego korzystania z systemu. Metafory, opierając się na obiektach ze świata rzeczywistego, dziedziczą wszystkie ich wady i fizyczne ograniczenia, a przez to nie wykorzystują ogromnych możliwości współczesnego programowania[69].

Idiom to *związek wyrazowy właściwy tylko danemu językowi, nieprzetłumaczalny dosłownie na inny język*[70], np. „powiedzieć prosto z mostu". Idiomy znalazły również zastosowanie w projektowaniu interfejsów. Większość komponentów interfejsu graficznego to właśnie idiomy. Okna, listy rozwijane, ikony sterujące oknami są elementami, z których nauczyliśmy się korzystać w sposób idiomatyczny[71], a nie na podstawie ich znaczenia metaforycznego. Okna w naszych domach wyglądają przecież zupełnie inaczej niż te w komputerze i wcale nie zamyka się ich w prawym górnym rogu, naciskając klamkę w kształcie kwadratu z krzyżykiem w środku.

Każdego idiomu trzeba się nauczyć[72], ale umysł człowieka jest dobrze przysposobiony do uczenia się tego typu rzeczy. Interfejsy poprzedniej generacji były trudne do opanowania, ponieważ wymagały od użytkownika zrozumienia ich wewnętrznego działania. Człowiek zdobywa większą część swojej wiedzy w sposób naturalny, bez potrzeby rozumienia zasad funkcjonowania rzeczy, których się uczy, np. rozpoznawania ludzkich twarzy, otwierania drzwi czy rozmieszczenia mebli w pokoju. Nie „rozumiemy", dlaczego czyjaś twarz jest akurat tak zbudowana i właśnie tak wygląda, ale „znamy" tę twarz. Rozpoznajemy ją, ponieważ patrzyliśmy na nią i automatycznie (i z łatwością) ją zapamiętaliśmy[73].

Idiomy odgrywają obecnie ogromną rolę w interfejsach graficznych. Prawdopodobnie w przyszłości zalety idiomów będą również doceniane ze względu na specyfikę uczenia się ich, która odpowiada naturalnym predyspozycjom człowieka do zdobywania wiedzy. W przeciwieństwie do metafor idiomy w interfejsach wykorzystują w pełni osiągnięcia z dziedziny programowania, a ich liczba praktycznie jest nieograniczona.

Kończąc rozważania na temat dialogu warto jeszcze przypomnieć, że dobry dialog systemu z użytkownikiem musi być adaptowalny, pozwalając użytkownikowi wykonywać biegle zadania w sposób, który najbardziej mu odpowiada. Użytkownik powinien mieć możliwość sterowania dialogiem

---

[68] Trudno byłoby znaleźć dobrą metaforę np. na proces zawężania wyników wyszukiwania czy zmiany bazy danych.

[69] Gdyby wszystko miało opierać się na metaforach, to np. hipertekst nigdy by nie zaistniał, ponieważ zasada działania hiperłączy nie znajduje odzwierciedlenia w zachowaniu się obiektów świata rzeczywistego.

[70] W. Kopaliński: *Słownik wyrazów...* op. cit., s. 221.

[71] Być może nawet nie zdajemy sobie z tego sprawy, bo wielu producentów oprogramowania zachwala swoje produkty i łatwość ich obsługi, powołując się na zastosowane metafory, które tak naprawdę są idiomami.

[72] Jeden z aksjomatów głoszonych przez A. Coopera brzmi: *Wszystkich idiomów trzeba się nauczyć. Dobrych idiomów uczy się tylko raz.* A. Cooper: *About Face...* op. cit., s. 59.

[73] A. Cooper: *About Face....* op. cit., s. 58.

i dostosowywania go do własnych potrzeb i umiejętności. Dlatego należy zwrócić szczególną uwagę na:

- elastyczność struktury dialogu pozwalającą użytkownikowi na natychmiastowe zrezygnowanie z wybranej opcji, pominięcie kilku kroków lub powrót do wybranego etapu wykonywanej procedury; umożliwi mu to bardziej efektywne uczenie się, pozwalając na eksperymentowanie;
- komunikaty skierowane do użytkownika, które powinny być poprawne językowo, zwięzłe, łatwe do zrozumienia i jednoznaczne; nie należy zadawać pytań z przeczeniami, aby nie narażać użytkownika na ich błędną interpretację[74]; powinno się używać języka, jakim posługuje się użytkownik i unikać żargonu informatycznego[75]; komunikaty nie mogą krytykować ani stresować użytkownika nawet wtedy, gdy popełniony przez niego błąd stworzył potencjalnie niebezpieczną sytuację, grożącą np. utratą danych – w takich sytuacjach pojawiający się komunikat powinien być rzeczowy i konstruktywny, informując użytkownika, jak ma w takiej sytuacji postąpić[76];
- konsekwentne stosowanie różnych rodzajów okien dialogowych w zależności od typu komunikatu[77].

ŁATWOŚĆ NAUCZENIA SIĘ I ŁATWOŚĆ W POSŁUGIWANIU SIĘ SYSTEMEM

Odpowiedź na pytanie, jaki powinien być system, aby można go nazwać „łatwym", wydaje się trudna, ponieważ zależy to od wielu czynników, a zwłaszcza od indywidualnych cech użytkowników. Jednak istnieją pewne znamiona, które mogą o tym decydować. Użytkownikom system nie sprawia trudności wtedy, gdy czują, że go kontrolują, znają jego możliwości i ograniczenia, wiedzą, jak postępować w przypadku pojawienia się błędu i jakich rezultatów oczekiwać po wykonaniu danej operacji, inaczej mówiąc – kiedy rozumieją system. Mowa jest tutaj o rozumieniu wyłącznie funkcjonalnym, a nie technicznym[78]. Użytkownicy rozumieją funkcjonalne działanie systemu wtedy, gdy dysponują jego poprawnym modelem mentalnym.

Największą trudność sprawiają użytkownikom arbitralnie zaprojektowane opcje i funkcje oraz sytuacje, w których nie mogą oni określić operacji, którą

---

[74] Na przykład pytanie „Czy nie chcesz zapisać sesji wyszukiwawczej?" z dwoma przyciskami do wyboru *TAK* i *NIE* może budzić wątpliwości, czy po naciśnięciu przycisku TAK system zapisze sesję, interpretując „*TAK*" w znaczeniu „tak, chcę zapisać sesję" czy jej nie zapisze, bo przy takim postawieniu pytania może to zostać zrozumiane jako „tak, nie chcę zapisać sesji". Właściwym pytaniem byłoby „Czy zapisać sesję wyszukiwawczą?"

[75] Nie oznacza to, że trzeba zrezygnować z terminów specjalistycznych. Wręcz przeciwnie – należy poznać terminologię dziedziny, którą obejmuje system i stosować charakterystyczne dla niej określenia, które jednak mogą być rozumiane inaczej w innych dyscyplinach. Dlatego właśnie jest tak ogromnie ważne, aby już podczas projektowania systemu poznać jak najdokładniej przyszłych użytkowników.

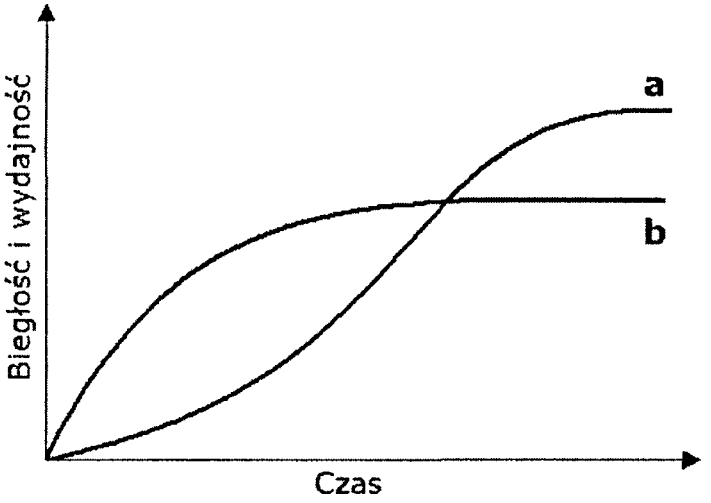
[76] Zamiast komunikatu „UWAGA!!! Zniszczysz dane zawarte w tym pliku!!!" lepiej byłoby napisać: „Plik „Rozdział 1" już istnieje. Zapisanie pliku pod tą samą nazwą spowoduje utratę poprzednich danych."

[77] Na przykład w systemie operacyjnym Windows wszystkie komunikaty ostrzegające pojawiają się w oknie dialogowym z ikoną ⚠; pytające – ?; informujące – i; krytyczne – ⊗.

[78] D. Norman: *The Invisible Computer.* Londyn 1999, s. 173.

system wykonuje w danej chwili, ani nie wiedzą, jak z niej zrezygnować; a także, gdy nie widzą efektów zastosowania wybranych funkcji.

Projektowanie interfejsu byłoby znacznie łatwiejsze, gdyby wszyscy użytkownicy z niego korzystający prezentowali cały czas taki sam poziom wiedzy. Problemy pojawiają się dlatego, że użytkownicy, w zależności od stopnia zaawansowania w obsłudze systemu, korzystają z niego na różne sposoby. Często zdarza się, że system, ze względu na początkujących użytkowników, jest łatwy do nauczenia się, ale zaawansowani użytkownicy nie mogą z niego wydajnie korzystać[79].



Rys. 4. Krzywe nauki dla hipotetycznych systemów, z których jeden jest łatwy do nauczenia się, ale mniej wydajny (*krzywa b*), a drugi trudny do nauczenia się, ale za to bardziej wydajny (*krzywa a*)[80].

Interfejsy ukierunkowane na łatwość uczenia się umożliwiają początkującym użytkownikom osiągnięcie w dość krótkim czasie wystarczającej wydajności i biegłości w obsłudze systemu. Natomiast w interfejsach, w których kładzie się nacisk na wydajność[81], zdobywanie pierwszych umiejętności posługiwania się systemem zabiera znacznie więcej czasu, ale pozwala w przyszłości na bardziej efektywną pracę.

Mogłoby się więc wydawać, że można korzystać albo z systemu łatwego do nauczenia się, albo wydajnego. Na szczęście istnieją interfejsy, które zaspokajają potrzeby zarówno użytkowników początkujących, jak i zaawansowanych. Co więcej, dobre interfejsy umożliwiają uczenie się i posługiwanie się nimi w taki sposób, że użytkownicy zdobywają wiedzę najpierw w sposób przedstawiony za pomocą krzywej b, a później (w odpowiednim czasie) – krzywej a.

---

[79] Ci sami użytkownicy, którzy jako początkujący chwalili system, po nabraniu wprawy w jego obsłudze, mogą być później niezadowoleni, skarżąc się na nieefektywne metody wykonywania zadań.

[80] J. Nielsen: *Usability Engineering*. Boston 1993, s. 28.

[81] Wydajność wynika przede wszystkim z łatwości w posługiwaniu się systemem.

Jednym ze sposobów osiągnięcia takiej możliwości jest stosowanie w interfejsach tzw. akceleratorów (*accelerators*), które pozwalają użytkownikowi na szybkie wykonywanie tych samych zadań, pomimo że identyczny rezultat można otrzymać stosując metody podstawowe[82], które są jednak wolniejsze. Typowe akceleratory to:

- **klawisze funkcyjne** znajdujące się na klawiaturze, które uruchamiają operację przypisaną do danego klawisza przez projektanta interfejsu, np. klawisz F1 w większości aplikacji uruchamia pomoc;
- **klawisze skrótów**, np. kombinacja klawiszy Ctrl+S, która w programach firmy Microsoft zapisuje dane na dysk;
- **menu kontekstowe** pojawiające się po naciśnięciu prawego przycisku myszy, z którego można wybrać i uruchomić dostępną w danym momencie funkcję, np. wysłania wyników wyszukiwania pocztą elektroniczną;
- **historia** (np. wyszukiwania) będąca listą akcji, które użytkownik wykonał podczas pracy z programem – zwalnia od ponownego wykonywania tych samych czynności, ponieważ można je wielokrotnie aktywować, dokonując wyboru z listy[83];
- **skróty nazw komend** znajdujące zastosowanie w przypadku, gdy interfejs graficzny umożliwia również używanie języka poleceń – pozwalają przyspieszyć wpisywanie komend, ograniczając przy tym liczbę popełnianych przez użytkownika błędów literowych, np. można wpisać *au = Dickens* zamiast *autor = Dickens*;
- **makroinstrukcje** (tzw. makra), czyli zestawy akcji i poleceń zapamiętane pod skrótem klawiszowym lub nazwą – po naciśnięciu właściwego klawisza skrótu lub wpisaniu bądź wybraniu nazwy makroinstrukcji system wykonuje automatycznie cały zapisany ciąg poleceń, co oszczędza użytkownikowi sporo czasu, zwalniając go od wielokrotnego wykonywania tych samych czynności.

Na rynku istnieje kilkadziesiąt tysięcy baz danych i z każdym rokiem ta liczba się zwiększa. Z tego powodu praca użytkowników korzystających wciąż z nowych baz mogłaby być niezmiernie trudna, gdyż wymagałaby od nich nieustannego poświęcania czasu na naukę obsługi nowych systemów. Problem ten został częściowo rozwiązany, ponieważ większość systemów wykorzystuje konsekwentnie umowne standardy[84], chociaż nie powstał jeszcze żaden ogólnoświatowy wzór interfejsu. Dzięki tym standardom proces uczenia się obsługi systemu skraca się znacznie[85], co ilustruje Rys. 5.

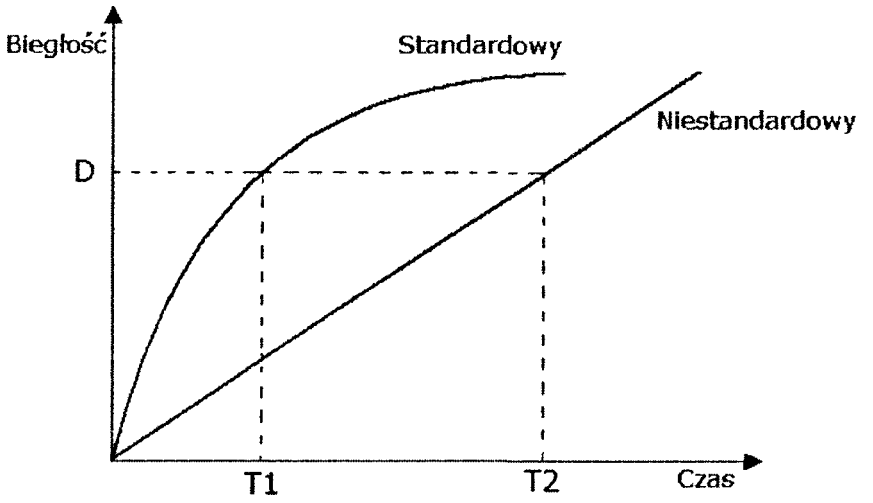
---

[82] Stanowią one pierwszą część krzywej a. Nie są wydajne, ale łatwe do nauczenia się i dlatego korzystają z nich najczęściej początkujący użytkownicy.

[83] Z historii można korzystać na ogół tylko podczas jednej sesji. Po zakończeniu programu i jego ponownym uruchomieniu lista akcji jest pusta. Nie jest to regułą, gdyż niektóre programy zachowują historię do czasu, aż użytkownik ją skasuje.

[84] Przyjęło się np., że obecne niemal w każdej aplikacji menu *Plik* zawiera standardowe funkcje, takie jak zapisywanie, otwieranie i zamykanie dokumentu, listę używanych ostatnio plików czy zakończenie działania programu. Innym nieformalnym standardem jest stosowanie klawisza F1 w celu przywołania systemu pomocy lub przycisku z ikoną drukarki służącego do drukowania. Powszechnie stosowane są również standardowe klawisze skrótów, np. Ctrl+C (kopiowanie do schowka), Ctrl+X (wycinanie), Ctrl+V (wklejanie ze schowka).

[85] J. Chabik: *Praktyka skutecznego programowania ....* op. cit., s. 107.



Rys. 5. Zależność biegłości w obsłudze systemu od czasu nauki dla interfejsu standardowego i niestandardowego

Rysunek przedstawia dwie linie reprezentujące obsługę tego samego systemu: jedną przy komunikacji standardowej, czyli takiej, jak w innych interfejsach, z których korzysta dany użytkownik, a drugą przy komunikacji niestandardowej, czyli specyficznej dla danego systemu. Na osi pionowej oznaczono punkt D, w którym biegłość w obsłudze systemu jest zadowalająca, to znaczy taka, przy której jego użytkownik może wydajnie pracować. Krzywa dla standardowego interfejsu jest początkowo bardzo stroma, gdyż użytkownik nie musi się uczyć najprostszych rzeczy, ponieważ zna je z innych systemów, z którymi dotychczas pracował. W przypadku interfejsu niestandardowego linia utrzymuje mniej więcej jednakowe nachylenie, bo użytkownik musi włożyć tyle samo wysiłku w poznanie podstawowych funkcji, co w poznanie bardzo zaawansowanych zastosowań systemu. Jak widać, punkt dostatecznej biegłości (D) w standardowym interfejsie użytkownika (T1) zostaje osiągnięty znacznie wcześniej niż w interfejsie niestandardowym (T2).

POMOC I WSPARCIE DLA UŻYTKOWNIKA

Im lepszy interfejs, tym mniejsza potrzeba użytkowników korzystania z pomocy zarówno np. ze strony bibliotekarzy, jak i w trybie online (pomocy systemowej). Niestety, rzadko się zdarza, by pomoc nie była w ogóle potrzebna w systemie. Wyjątek mogą stanowić bardzo proste w obsłudze systemy, które zaprojektowano w celu szybkiej obsługi użytkowników (tzw. systemy *walk-up-and-use*), np. bankomaty[86].

Interfejsy systemów informacyjno-wyszukiwawczych posiadają zbyt wiele funkcji i opcji, aby można było korzystać z nich bez dodatkowej pomocy w for-

[86] Niewiele bardziej skomplikowane systemy, jak np. automaty do sprzedaży biletów lub napojów, wymagają już krótkiej instrukcji obsługi, umieszczonej w formie pisemnej w miejscu łatwo widocznym dla użytkowników.

mie elektronicznej czy też drukowanej (podręcznik, dokumentacja). Problem tkwi w tym, że zdecydowana większość użytkowników nie czyta instrukcji obsługi. Świadomie bądź nieświadomie stosują oni jedno z praw Murphy'ego, które radzi: „Gdy wszystko inne zawiedzie, przeczytaj instrukcję obsługi". Rada ta może jest śmieszna i niedorzeczna[87], ale – niestety – według niej postępuje większość użytkowników, i to nie tylko systemów informacyjnych[88]. Użytkownicy nie chcą tracić czasu na czytanie instrukcji i dlatego rozpoczynają natychmiast pracę z systemem, aby jak najszybciej wyszukać potrzebne informacje. Z pomocy korzystają dopiero w ostateczności. Pomimo tego warto tworzyć rozbudowane i szczegółowe systemy pomocy, ponieważ w niektórych sytuacjach okazują się bardzo przydatne, zwłaszcza dla zaawansowanych użytkowników[89]. Poza tym, mogą one zastąpić instrukcje drukowane, przewyższając je na ogół szybkością użycia dzięki możliwościom stosowania zautomatyzowanych technik wyszukiwania informacji na dany temat. Ponadto, ograniczona liczba egzemplarzy podręczników do obsługi systemu nie pozwala zwykle na to, aby przy każdym stanowisku znajdował się jeden egzemplarz[90].

Nie bez powodu mówi się o systemie pomocy, ponieważ pomoc bywa tak obszerna i wieloaspektowa, że sama może stanowić odrębny system. Wynika z tego wiele problemów, gdyż użytkownicy muszą się dodatkowo nauczyć, jak korzystać z systemu pomocy[91]. Potwierdzają to obserwacje, na podstawie których stwierdzono, że w 52.576 sesjach wyszukiwawczych 23% wszystkich prób skorzystania z systemu pomocy okazało się nieudanych, to znaczy że użytkownicy nie otrzymali od systemu żadnej pomocy. W przypadkach, w których użytkownikom udało się uzyskać pomoc, jej przydatność ocenili oni zaledwie na 35%[92].

Tworząc system pomocy projektanci powinni kierować się również pewnymi zasadami, wytycznymi i standardami, aby użytkownicy mogli łatwo z niego korzystać. Przede wszystkim należy zadbać o zrozumiałość, zwięzłość i dobrą strukturę tekstu pomocy, gdyż – jak twierdzi N. Borenstein – jakość tekstów pomocy jest dużo ważniejsza niż mechanizmy, za pomocą których z nich się korzysta[93]. Użytkownicy nie mają ochoty czytać długich wywodów, lecz szukają wzrokiem jedynie najważniejszych dla nich zagadnień, a jednolite bloki tekstu, wypełniające całkowicie ekran monitora, natychmiast zniechęcają ich do czytania. Podstawową sprawą jest język i układ graficzny prezentowanych

---

87 Zresztą jak wszystkie pozostałe rady Murphy'ego, których zadaniem jest pocieszanie osób, którym często coś się nie udaje.

88 Potwierdza to M. Retting: *Nobody reads documentation.* „Communications of the ACM" 1991 vol. 34 nr 7, s. 19-24.

89 Chociaż mogłoby wydawać się, że pomoc powinna służyć przede wszystkim użytkownikom początkującym, nie jest to jednak prawda. Użytkownicy początkujący w ogóle z niej nie korzystają, ponieważ jest to dla nich za trudne – niemal tak trudne jak obsługa głównego systemu. Poza tym, jeśli istnieje potrzeba odwoływania się do pomocy w celu wyjaśnienia działania podstaw systemu, oznacza to, że interfejs jest bardzo źle zaprojektowany. System pomocy powinien koncentrować się na użytkownikach zaawansowanych, którzy potrafią już wykorzystywać podstawowe funkcje, ale chcą poszerzyć swoją wiedzę i zakres umiejętności.

90 Jako inną zaletę wymienia się również fakt, że pomoc on-line nie może zginąć – jak to dzieje się czasem z drukowanymi instrukcjami.

91 Mimo że wydaje się to niedorzeczne, ale potrzebna jest wówczas „pomoc do pomocy", którą można zazwyczaj znaleźć pod hasłem: „Jak korzystać z pomocy".

92 Cyt. za. J. Nielsen: *Usability Engineering....* op. cit., s. 150.

93 Tamże, s. 150.

informacji. Godnym polecenia pomysłem jest powierzenie redagowania tekstów pomocy osobom z wykształceniem filologicznym[94]. Żargon informatyczny i wielokrotnie złożone zdania z licznymi wtrąceniami skutecznie uniemożliwiają zrozumienie czegokolwiek, dlatego należy starać się podawać instrukcje w punktach, krok po kroku. Będzie to zarówno przejrzyste, jak i zrozumiałe, a poparte dodatkowo ilustracjami przyczyni się do stworzenia lepszego modelu mentalnego, który – jak już wiadomo – pomaga zrozumieć zasady działania systemu. Cenne są też przykłady łatwiejsze do zrozumienia niż abstrakcyjne opisy[95]. Powinna istnieć możliwość kopiowania ich do schowka i późniejszego wykonania podczas właściwej sesji wyszukiwawczej.

Użytkownicy mają tendencję do czytania wyjaśnień związanych wyłącznie z interesującym ich zagadnieniem, pomijają natomiast poprzednie sekcje, które jednak mogą okazać się ważne. Dlatego należy dążyć do tego, aby opisy poszczególnych zagadnień stanowiły w miarę zamkniętą całość, a tam, gdzie jest to niezbędne, odsyłały użytkownika do relewantnego fragmentu w innej sekcji, najlepiej za pomocą hiperłączy.

Projektanci muszą także zadbać o to, aby system pomocy i zawarte w nim przykłady dotyczyły wersji systemu, z której właśnie korzystają użytkownicy, jeżeli bowiem nie można aktualnie zastosować wyjaśnień i wskazówek użytkownikom wydaje się, że przyczyną takiego stanu jest ich nieudolność.

Korzystanie z systemu pomocy ułatwiają trzy podstawowe narzędzia:

- indeks – powinien zawierać nie tylko terminologię systemową, lecz także słownictwo związane z technikami rozwiązywania różnych zadań (słownictwo ukierunkowane na zadanie); indeks musi zawierać jak najwięcej synonimów oraz terminy używane przez konkurencyjne systemy do określania tych samych pojęć, ponieważ użytkownicy być może wcześniej z nich korzystali; dużym udogodnieniem jest również systemowe dopisywanie (dopasowywanie) reszty słowa, znajdującego się w indeksie, do wpisanych przez użytkownika liter; może to skrócić czas wpisywania, a ponadto szybko można sprawdzić, czy dane słowo występuje w indeksie;
- wyszukiwanie pełnotekstowe – użytkownik wpisuje słowa kluczowe, które jednak nie znajdują się w indeksie, lecz w tekście opisu wyjaśniającego dane zagadnienie – jest to bardzo pomocne, ale może działać wolno w przypadku rozbudowanego systemu pomocy;
- mapa struktury (*overview map*) – implementowana na ogół w postaci spisu treści lub diagramu; bardzo przydatny w spisie treści może okazać się dodatkowy punkt poświęcony rozwiązywaniu najczęściej pojawiających się problemów.

Pomoc udzielana użytkownikom może przyjmować też inne formy. Dobrze sprawdzają się etykiety (krótkie napisy) wyjaśniające działanie funkcji, przypisanej do danej ikony (*tooltips*). Etykiety pojawiają się na ekranie lub na pasku stanu po umieszczeniu kursora myszki na ikonie lub w innym wyznaczonym miejscu. Skuteczna jest również pomoc kontekstowa – użytkownik, wskazując

---

[94] Po wielu błędach językowych i nieporadnych konstrukcjach gramatycznych w polskiej wersji systemu operacyjnego Windows 95 jego następca, Windows 98, został spolszczony przez profesjonalistów – polskich językoznawców. Poprawiło to ogromnie zrozumiałość plików pomocy, a także wszelkich komunikatów systemowych.

[95] J. LeFevre, P. Dixon: *Do written instructions need examples?* „Cognition and Instruction" 1986 vol. 3 nr 1, s. 1-30.

myszką dowolny obiekt w interfejsie, może otrzymać wyjaśnienia na temat zastosowania tego obiektu.

Uruchomienie pomocy w żadnym wypadku nie może doprowadzić do utraty wcześniej wpisanych danych. Wyjaśnienia powinny pojawiać się w osobnym oknie, które użytkownik będzie mógł umieścić w dowolnym miejscu na ekranie, aby mógł jednocześnie kontynuować wykonywanie zadania i korzystać z pomocy[96].

Cały wysiłek projektowania powinien skupiać się jednak nie na systemie pomocy, a na interfejsie, którego zadaniem jest informowanie[97] o dostępnych opcjach, stosowanie takich terminów, które nie wymagają dodatkowych wyjaśnień i – ogólnie mówiąc – wspieranie użytkownika. Nie wolno kierować się przekonaniem, że wszystko jest opisane w dokumentacji i kto chce się nauczyć obsługiwać system, ten niech uważnie ją czyta. Nic bardziej błędnego! To właśnie interfejs ma za zadanie pokazać możliwości systemu.

Wspieranie użytkownika nie ogranicza się tylko do udzielania pomocy. System powinien dbać o to, aby użytkownik mógł cały swój czas poświęcić wyłącznie na rozwiązywanie zadań, a nie na zmaganie się z interfejsem. Aby to osiągnąć, należy stosować się do następujących trzech wskazówek:

1. System powinien wspierać wykonanie całego zadania, a nie pojedynczą operację[98] (*task-centered development*), dzięki czemu czynności mogą układać się w logiczną całość, znaną użytkownikowi ze świata rzeczywistego.

2. W centrum zainteresowania ma być użytkownik, a nie system (*human-centered development*). Nie wolno tworzyć interfejsu utrudniającego pracę użytkownika tylko dlatego, że taki system łatwiej było zaprogramować.

3. Dobry model konceptualny to podstawa szybkiego nauczenia się obsługi systemu (*conceptual mode-guided development*). Interfejs powinien komunikować, jakie czynności można w danej chwili wykonać. Należy wykorzystywać ograniczenia (*constraints*)[99], które ułatwiają tworzenie poprawnego modelu mentalnego, odciążając pamięć użytkownika i ograniczając możliwości popełnienia przez niego błędu.

PREZENTACJA NA EKRANIE

Patrząc na interfejs użytkownik powinien łatwo dostrzegać dostępne funkcje i rozpoznawać, jaką operację wykonuje system w danym momencie. Prezentacja ekranowa musi być ściśle powiązana z dialogiem i sprzężeniem zwrotnym, aby umożliwić użytkownikowi obserwowanie rezultatów podjętej przez niego akcji. W tym celu interfejs powinien wykorzystywać strukturę Model – Widok – Kontroler (**MVC**, ***Model-View-Controller***), która zapewnia poprawną interakcję użytkownika z systemem[100].

---

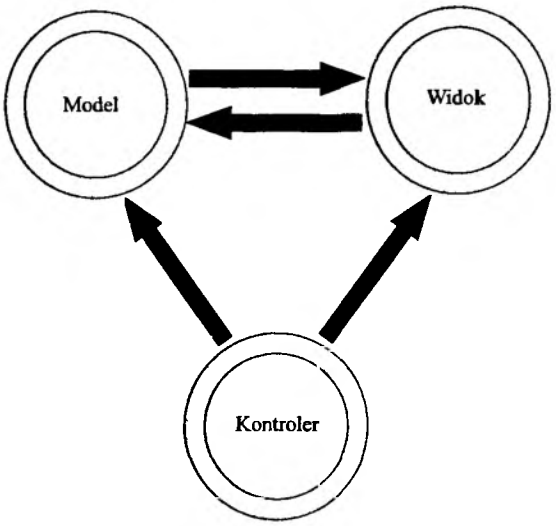
[96] Dobrze byłoby również, gdyby pojawiające się okno z tekstem pomocy nigdy nie zasłaniało miejsca, gdzie znajduje się kursor do wprowadzania danych.

[97] Oczywiście nie w sensie wyświetlania komunikatów. Każdy komponent interfejsu powinien prezentować się tak, aby było jasne, do czego służy.

[98] Obowiązuje tu również zasada „najpierw obiekt, potem czynność", która jest odzwierciedleniem posługiwania się przedmiotami w otaczającym nas świecie.

[99] W tym przypadku wspieranie użytkownika polega na tym, że interfejs ogranicza chwilowo liczbę opcji i funkcji do tych, które można wykonać w danym momencie. Na przykład, funkcja drukowania nie powinna być aktywna, gdy użytkownik nie otrzymał jeszcze żadnych rezultatów wyszukiwania. Takie ograniczenia mogą dotyczyć wielu aspektów interfejsu i powinny zawsze pełnić rolę pozytywną, gdyż celowe ograniczenia działają na korzyść użytkowników.

[100] Paradygmat MVC pochodzi z obiektowego języka programowania Smalltak-80.



Rys. 6. Struktura *Model – Widok – Kontroler*

Model, precyzyjniej nazywany modelem informacyjnym, tworzą obiekty wewnątrz systemu odzwierciedlające obiekty ze świata rzeczywistego. Jak każdy obiekt w systemie, mają przypisane atrybuty i metody, z których korzysta kontroler i widok wyłącznie za pomocą wysyłanych komunikatów[101].

Widok prezentuje na ekranie aktualny stan obiektu wchodzącego w skład modelu. Widokiem może być np. pole wyświetlające liczbę znalezionych dokumentów, które spełniły kryteria użytkownika. Widok i model są ze sobą połączone, a każda zmiana w modelu pociąga za sobą zmianę w widoku. Symbolizuje to strzałka od modelu do widoku.

Kontroler to ogólna nazwa obiektów, które użytkownik wykorzystuje do interakcji z modelem (systemem). Kontroler implementuje działania fizyczne, takie jak naciśnięcie przycisku myszy lub klawisza na klawiaturze. Za ich pomocą do modelu wysyłane są komunikaty uruchamiające odpowiednie metody, które mogą zmienić lub sprawdzić jego stan[102]. Kontroler zajmuje się interakcją między użytkownikiem i modelem lub widokiem. Użytkownik może wprowadzić zmiany do modelu bądź zmodyfikować dane wyjściowe (widok). Te dwie akcje reprezentowane są przez strzałki z kontrolera do modelu i z kontrolera do widoku.

Wygląd graficzny interfejsu nie pełni wyłącznie roli dekoracyjnej, ale służy do efektywnej komunikacji, która może kierować działaniami użytkownika, motywować go do poszerzania wiadomości na temat zaawansowanych funkcji systemu, ale i zniechęcać go, męczyć i rozpraszać. Nie jest więc bez zna-

[101] Mowa jest tutaj o pojęciach wykorzystywanych w programowaniu obiektowym. **Atrybuty** – cechy obiektu, dane opisujące obiekt, które są istotne z punktu widzenia dziedziny systemu. W atrybutach przechowuje się informacje zmieniane i udostępniane przez metody. **Metody** – funkcje lub procesy, które umożliwiają manipulowanie obiektami, czyli odczytywanie i zmianę wartości ich atrybutów. **Komunikaty** – wyrażenia językowe skierowane do obiektu, wywołujące (uruchamiające) jedną z metod skojarzoną z tym obiektem.

[102] Każdy obiekt posiada **stan**, będący kombinacją wartości wszystkich składowych obiektu, przede wszystkim wartości wszystkich jego atrybutów oraz powiązań z innymi obiektami.

czenia organizacja i prezentacja informacji na ekranie, która obejmuje takie elementy jak:

- komponenty interfejsu,
- kolory,
- układ graficzny,
- dźwięki[103].

KOMPONENTY INTERFEJSU[104]

Ikony

Elementami, od których prawdopodobnie najbardziej zależy ocena wyglądu interfejsu, są ikony. Mogą one reprezentować złożone obiekty i funkcje na niewielkiej powierzchni, która w interfejsie, ze względu na ograniczenia ekranu monitora, jest bardzo cenna. Użytkownicy rozpoznają szybciej rzeczy prezentowane za pomocą ikon niż samego tekstu.

Można wyróżnić trzy rodzaje ikon[105]:

1/ arbitralne – niemające nic wspólnego z normalnym wyglądem czynności, którą reprezentują; zależą wyłącznie od ich autora;

Rys. 7. Przykłady ikon arbitralnych

2/ abstrakcyjne – budzące naturalne skojarzenia za pomocą elementu albo cechy charakterystycznej obiektu lub czynności; nie są całkowicie umowne jak arbitralne, ale nie przypominają tak ściśle rzeczywistości jak ikony dosłowne;

Rys. 8. Przykłady ikon abstrakcyjnych

3/ dosłowne – odwzorowujace przedmiot lub czynność, która zostanie wykonana w wyniku ich naciśnięcia.

Rys. 9. Przykłady ikon dosłownych

---

[103] Mimo że nie należą sensu stricte do „prezentacji na ekranie", mogą wpływać na ogólną ocenę interfejsu i dlatego zostaną tutaj omówione.

[104] Zostaną omówione tylko te najważniejsze. Szczegółowe informacje na temat dokładnego projektowania kontrolek, suwaków, ramek, pasków narzędzi i innych elementów można znaleźć w *The Windows Interface Guidelines for Software Design.* Redmond 1995.

[105] J. Chabik: *Praktyka skutecznego programowania.....* op. cit. s, 121-122.

Ikony powinny spełniać następujące warunki:

- reprezentować w zrozumiały sposób obiekt albo czynność;
- pozostawać w zgodności z metaforami mentalnego modelu użytkownika, bez korzystania z negatywnych czy specyficznych dla danej kultury skojarzeń;
- charakteryzować się łatwością nauczenia i zapamiętania;
- odznaczać się wyraźnie na tle;
- reprezentować konsekwentny styl pod względem kolorystyki, rozmiaru, kształtu;
- zapewniać wizualne podobieństwo z ikonami o podobnych funkcjach;
- imitować trójwymiarowość[106];
- na wybranej przez użytkownika ikonie powinna pojawiać się etykieta tekstowa, wyjaśniająca funkcję reprezentowaną przez tę ikonę;
- zaznaczona ikona musi odróżniać się od pozostałych, niezaznaczonych ikon.

Ikony powinny być projektowane przez projektantów graficznych. Projektanci systemu mają jednak do dyspozycji setki gotowych ikon, które są często stosowane w różnych aplikacjach i zaczynają powoli stanowić standard. Dlatego właśnie należy wykorzystywać istniejące[107] już ikony, co nie tylko oszczędza czas, ale również ułatwia obsługę systemu, ponieważ użytkownicy mogą znać niektóre ikony z innych interfejsów[108].

Okna

Komunikacja użytkownika z systemem odbywa się najczęściej w oknach. Każde okno powinno mieć tytuł i być otoczone wyraźnymi ramkami[109]. W przypadku, gdy informacje nie mieszczą się w oknie, potrzebne są suwaki do przewijania jego zawartości. Istnieją suwaki poziome i pionowe, ale z obserwacji użytkowników wynika, że suwaki poziome sprawiają im zwykle problemy, gdyż po pierwsze – czytanie jest utrudnione ze względu na to, że trzeba przesuwać suwak do końca linii i z powrotem na jej początek, a dodatkowo korzystać z pionowego suwaka, aby móc czytać kolejne linijki tekstu; po drugie – użytkownicy po prostu zapominają o poziomym przesuwaniu, ponieważ korzystają z niego rzadko (w porównaniu z przewijaniem pionowym). W ten sposób mogą pominąć cenne informacje, niewidoczne w danym momencie na ekranie, dlatego projektanci powinni stosować wyłącznie suwaki pionowe.

Użytkownik musi mieć swobodę w rozmieszczaniu okien na ekranie i zmienianiu ich rozmiarów. Wskazane jest, aby po zamknięciu okno ponownie pojawiało się dokładnie w tym samym miejscu i miało taki sam rozmiar. Użytkownik, korzystając z kilku okien równocześnie, poświęca sporo czasu na ułożenie ich tak, aby mógł łatwo z nich korzystać, dlatego systemowe zapamiętywanie

---

[106] Dzięki zastosowaniu odpowiednich kolorów można uzyskać efekt trójwymiarowości, który przyciąga wzrok użytkownika, ale może również działać rozpraszająco.

[107] Można skorzystać z bibliotek ikon, czyli zbiorów ikon, pogrupowanych zazwyczaj w kategorie. Z niektórych zbiorów można korzystać bezpłatnie, natomiast inne mogą być chronione prawem autorskim i niezbędna jest zgoda na ich używanie.

[108] Należy się jednak upewnić, czy w konkurencyjnych systemach, z których korzystał dotychczas użytkownik, ta sama ikona reprezentuje tę samą funkcję. W przeciwnym razie użytkownik będzie popełniał szereg pomyłek, a to z kolei wpłynie negatywnie na ocenę nowego systemu.

[109] Ma to szczególne znaczenie, gdy okna stykają się ze sobą krawędziami, ponieważ wówczas widać wyraźnie ich zakres.

rozmieszczenia i wielkości okien stanowi dla niego istotną pomoc. Przydaje się również automatyczne układanie okien, np. kaskadowo czy sąsiadująco w pionie lub poziomie.

Menu

Trudno byłoby znaleźć system informacyjny, który nie posiadałby żadnego menu. Menu, w którym użytkownicy mogliby intuicyjnie znajdować interesujące ich opcje, wydaje się wręcz niemożliwe do zaprojektowania, ponieważ pozycje i układ menu zależą od charakteru systemu. Jedna z ogólnych wytycznych zaleca, aby każda aplikacja posiadała trzy podstawowe menu[110]:

1. Plik (*File*), m.in. z takimi funkcjami, jak zapisywanie, drukowanie, otwieranie i zamykanie plików; powinno zajmować pierwsze miejsce od lewej;
2. Edycja (*Edit*), drugie w kolejności menu, w którym powinno się znaleźć kopiowanie, wklejanie i kasowanie danych;
3. Pomoc (*Help*) – menu uruchamiające system pomocy, a także udostępniające ogólne informacje o programie (datę wypuszczenia na rynek, numer wersji); powinno być umieszczone jako ostatnie, czyli pierwsze z prawej strony.

Pozostałe menu zależą od funkcji, które spełnia system. Dąży się to tego, aby grupowały one funkcje, układając je w kolejności „od ogółu do szczegółu", zaczynając od lewej strony. Pierwsze, najbardziej ogólne menu, dotyczyłoby ustawień związanych z samym programem, drugie – z konkretnym zadaniem (np. ustawianie marginesów w redagowanym dokumencie, drukowanie wyników wyszukiwania), trzecie – z właściwościami i funkcjami obiektów występujących w tym zadaniu (np. wybór stylu dla zaznaczonego akapitu, wybór języka z listy w kryteriach wyszukiwawczych dla danego pola), a czwarte – z pomocą[111]. Ważne jest, żeby każde menu miało właściwą nazwę, określającą jednoznacznie zastosowanie występujących w nim opcji. Należy również pamiętać o tym, aby:

- stosować szerokie i płytkie menu;
- pokazywać pozycje za pomocą grafiki, liczb lub tytułów;
- stosować nazwy opcji jako nazwy dla drzew;
- właściwie (logicznie) grupować opcje;
- nadawać opcjom krótkie i znaczące nazwy;
- stosować konsekwentny układ i terminologię;
- wykorzystywać różne rodzaje akceleratorów;
- umożliwić elastyczne sposoby poruszania się po menu (pomijanie poszczególnych ekranów, powrót do menu głównego)[112].

Ciekawą propozycją jest również zastąpienie dotychczasowego menu kontekstowego w postaci tradycyjnego spisu opcji tzw. menu tortowym. Poszczególne opcje są w nim rozmieszczone w sektorach okręgu, a nie w segmentach poziomego paska. Z takiego menu dokonuje się wyboru o 30% szybciej i robi o połowę mniej błędów w stosunku do menu tradycyjnego. Wynika to z faktu, że kąty zapamiętuje się o wiele lepiej niż odległości, a dodatkowo pomaga przy tym pamięć mięśni[113].

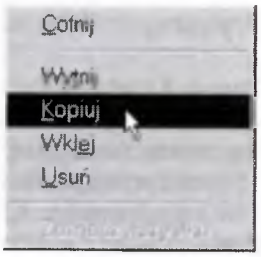
---

[110] Niestety, w przypadku systemów baz danych ta zasada nie jest przestrzegana. Bazy posiadają różne menu, a jedynym wspólnym jest zazwyczaj menu Pomoc.

[111] A. Cooper: *About Face....* op. cit., s. 285-286.

[112] B. Shneiderman: *Designing the User Interface....* op. cit., s. 121.

[113] *Cztery pomocne (przypuszczalnie) innowacje.* URL: http://www.proszynski.pl/-swiatnauki/1997/wrzesien/trendy.htm.



Rys. 10. Menu standardowe

Rys. 11. Menu tortowe

KOLORY

Kolory zwiększają atrakcyjność interfejsów, ale nieumiejętnie dobrane mogą wpływać negatywnie na użytkowników. Nie można podać dokładnych zasad stosowania kolorów, ponieważ są one w dużej mierze sprawą indywidualnych upodobań i kultur[114] oraz zależą od położenia, rozmiaru i kształtu obszaru, który wypełniają, a także od otaczających go kolorów[115]. W projektowaniu niezbędne są jednak ogólne wiadomości na temat widzenia barw przez oko ludzkie, gdyż pomaga to zrozumieć istotę kolorów i może pomóc znacznie projektantowi w doborze odpowiedniej kolorystyki dla interfejsu.

Człowiek rozpoznaje łatwo i bezbłędnie zaledwie sześć kolorów: trzy podstawowe (czerwony, niebieski, żółty) plus ich drugorzędne połączenia (biały, czarny, zielony) oraz radzi sobie dobrze ze skalą szarości[116]. Przeciętnie jest jednak w stanie odróżnić około 200 różnych barw o identycznym oświetleniu. Ponieważ każda z tych barw może się charakteryzować różnym nasyceniem (kolory mogą być intensywne lub blade) oraz mieć różną jasność, człowiek może odróżnić około dwóch milionów różnych jakości barwnych. Percepcja barw jest bardzo skomplikowanym procesem, w którym biorą udział zarówno receptory siatkówkowe, jak i liczne struktury układu nerwowego.

Kształt jest lepszym narzędziem identyfikacji obiektów niż kolor – nie tylko dlatego, że ujawnia liczniejsze różnice jakościowe, ale i dlatego, że wyraziste cechy kształtu silniej opierają się zmianom zachodzącym w ich otoczeniu. Dlatego kolory należy stosować jako uzupełnienie do kształtu, wzoru i położenia prezentowanych informacji. Kolor nie może stanowić w interfejsie jedynego środka wyrażania wartości czy funkcji.

Niektóre kolory mają umownie przypisane znaczenie, budzące podświadome skojarzenia. Należy o tym pamiętać i wykorzystywać je zgodnie z poniższymi zaleceniami[117].

- **Biel** kojarzona jest z czystością i wolnym miejscem, dlatego pusty obszar przeznaczony dla użytkownika powinien być biały.
- **Czerń** to kolor obramowania i treści. Czarne powinny być litery, znaki rysunki. Czarna ramka powinna otaczać to wszystko, co ma być zaprezentowane jako całość.

---

[114] Szacuje się, że aż 8% mężczyzn i 0,5% kobiet ma problemy z poprawnym rozpoznawaniem kolorów.

[115] Na przykład szare obiekty na zielonym tle mogą wydawać się lekko zabarwione na czerwono.

[116] R. Arnheim: *Sztuka i percepcja wzrokowa. Psychologia twórczego oka.* Warszawa 1978, s. 335.

[117] J. Chabik: *Praktyka skutecznego programowania*....op. cit., s. 116.

- **Czerwień** oznacza zagrożenie, zakaz, wystąpienie błędu. Należy ją stosować konsekwentnie do komunikowania o krytycznych sytuacjach. Nie wolno jej używać równocześnie w innym celu.
- **Zieleń** to zgoda, bezpieczeństwo, neutralność. Kolor zielony należy stosować do sygnalizowania bezpiecznych opcji i neutralnych informacji.
- **Żółć** oznacza ostrzeżenie. Kolor żółty może zwracać uwagę użytkownika na potencjalne zagrożenie, sygnalizować możliwość wystąpienia błędu, itp.
- **Błękit** symbolizuje stałość, trwałość, chłód i neutralność. Kolor niebieski mają najczęściej obiekty specyficzne dla danej aplikacji, w które użytkownik nie może zwykle ingerować.

Kolorystyka interfejsu zależy od przeznaczenia systemu[118]. Systemy, z których korzysta się po kilka godzin dziennie, muszą używać stonowanych kolorów. Zaleca się wykorzystanie najpierw odcieni szarości, a dopiero potem dodawanie kolorów, których liczba nie powinna przekraczać czterech. Należy również pamiętać, że im rzadziej używany jest kolor, tym skuteczniej przyciąga uwagę użytkowników. Efekt będzie spotęgowany, gdy zostaną w tym celu wykorzystane kontrasty. Oto przykładowe pary kontrastujących ze sobą kolorów: czarny – biały, czerwony – zielony, niebieski – żółty, purpurowy – zielony, zielono – niebieski – pomarańczowy, żółtozielony – fioletowy.

Wynika z tego, że aby otrzymać dobrą widoczność tekstu na ekranie, można zastosować, oczywiście oprócz tradycyjnej bieli i czerni, kolor niebieski dla tła i żółty dla liter. Należy unikać niekontrastowych zestawień, jak niebieski – zielony, żółty – czerwony. W zasadzie nie powinno stosować się czerni bez tekstury jako tła, ponieważ może to spowodować efekt „pływania" kolorowych liter. W przypadku, gdy zastosowanie czerni jako tła jest nieuniknione, litery muszą być w kolorze białym lub nienasyconym czerwonym, zielonym, żółtym lub niebieskim. Dobrym kolorem na tło jest neutralny jasnoszary, teksturowany[119].

Niektórzy projektanci, uważając czarne litery za mało oryginalne i zbyt często stosowane przez innych, używają w swoich programach niebieskiej czcionki. Zdarza się to często, a jest to duży błąd, bo człowiek postrzega zawsze tekst napisany nasyconym (czystym) kolorem niebieskim jako rozmyty i zamazany, co jest uciążliwe w trakcie dłuższego czytania. Nie należy więc używać koloru niebieskiego do małych obiektów, jak np. tekstu. W zależności od tła zaleca się tekst w kolorze czarnym, białym lub szarym, ponieważ na tych kolorach oko skupia się najlepiej[120].

Kolejny błąd, który projektanci[121] popełniają dość często, to nadużywanie nasyconych kolorów. Takie kolory powodują szybkie zmęczenie wzroku. Nasycone kolory mogą też wywoływać błędne postrzeganie głębi. Na przykład patrząc na czerwony obiekt wydaje się, że jest on położony bliżej niż niebieski. Może się również wydawać, że nasycone kolory „pływają" przed lub za ekranem monitora, co z pewnością nie ułatwia pracy użytkownikom. Dlatego najlepiej stosować kolory wymieszane, lekko stonowane, o widmie rozmytym, jak błękity, turkusy, fiolety i odcienie szarości, które nie męczą oczu i nie rozpraszają użytkowników, spędzających niejednokrotnie wiele godzin przed ekranem monitora.

---

[118] Aplikacje dla dzieci, gry i inne programy mające na celu przyciągnięcie użytkowników muszą stosować bardzo krzykliwe i kontrastowe kolory, które jednak nie nadają się do interfejsów systemów informacyjno-wyszukiwawczych.

[119] L. J. Najjar: *Using color effectively (or peacocks can't fly).* IBM TR52.00188. Atlanta, 1990.

[120] L. J. Najjar: *Using color effectively...* op. cit.

[121] Bogate i renomowane firmy informatyczne do projektowania interfejsów zatrudniają grafików i plastyków, którzy nie popełniają takich błędów.

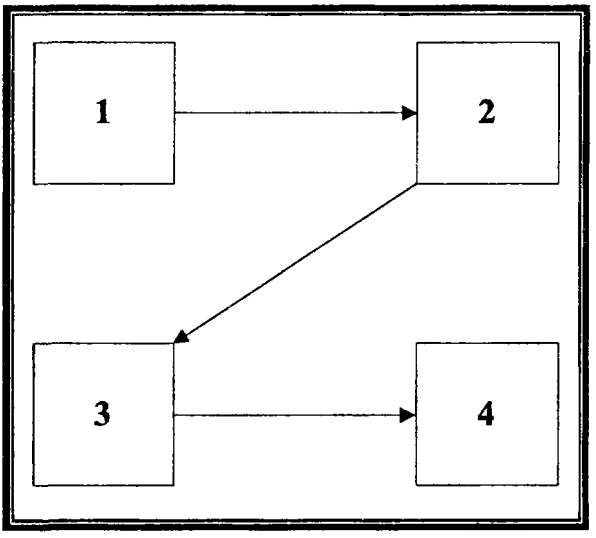
Mimo że ekran jest płaski, można na nim wywołać wrażenie trójwymiarowości, nie tworząc skomplikowanych perspektyw, a jedynie umiejętnie dobierając kolory. Człowiek postrzega bowiem to, co ciemniejsze jako dalsze, a to, co jaśniejsze jako bliższe. Tak więc, jeżeli potrzeba ukazać jakąś informację jako „bliższą" w stosunku do obecnych już na ekranie, należy lekko rozjaśnić jej tło. W celu „oddalenia" informacji wystarczy nieco ściemnić jej tło.

Postrzeganie barw bywa kwestią gustu. Każdy może mieć swój ulubiony kolor, a nawet „osobistą" paletę kolorów. Z tego powodu wszystkie interfejsy powinny posiadać opcję umożliwiającą dostosowywanie kolorów do indywidualnych upodobań użytkowników. Powinna istnieć możliwość wyboru koloru poszczególnych obiektów interfejsu (np. pasków, menu), a także gotowe tzw. schematy, ustawiające odpowiednio dobrane kolory dla wszystkich komponentów. Dotyczy to również dostosowywania menu oraz ikon. Użytkownicy muszą mieć możliwość dodawania i usuwania opcji w menu lub zmieniania ikon na paskach narzędzi[122]. Zwiększy to adaptowalność interfejsu i umocni dodatkowo użytkownika w przekonaniu, że to on kontroluje system.

UKŁAD GRAFICZNY

Dobry projekt graficzny to ważny element warunkujący prosty i naturalny dialog systemu z użytkownikiem. Czytając z ekranu, staramy się organizować informacje, grupując je przestrzennie. Elementy postrzegane są jako grupa, gdy znajdują się blisko siebie, otoczone są ramką, razem poruszają się lub zmieniają, są do siebie podobne pod względem koloru, kształtu lub wielkości.

Planując rozkład graficzny elementów w interfejsie należy brać pod uwagę kolejność, według której człowiek postrzega informacje.



Rys. 12. Kolejność postrzegania elementów w zamkniętej powierzchni

[122] Mogą to być zmiany liczby, wyglądu i funkcji ikon. Użytkownik powinien sam decydować, jakie ikony (czyli funkcje) chce mieć na danym pasku. W ten sposób może ograniczyć liczbę ikon, pozostawiając tylko te, których używa najczęściej. Pozwoli mu to na szybsze dokonywanie wyboru.

Patrząc na zamkniętą powierzchnię, np. ekran monitora lub kartkę papieru, większość osób[123] będzie postrzegać znajdujące się na nim elementy, kreśląc wzrokiem kształt litery **Z** i skupiając największą uwagę na miejscach oznaczonych kwadratami, a najlepiej zapamiętane zostaną obszary reprezentowane przez kwadraty nr 1 i 4 (Rys. 12)[124]. Warto o tym pamiętać i umieszczać najważniejsze informacje w lewym górnym rogu ramki lub okna, trochę mniej ważne – w prawym dolnym, a najmniej ważne – w prawym górnym i lewym dolnym.

Jeśli chcemy dodatkowo zwrócić uwagę użytkownika na jakąś ważną informację, możemy wykorzystać jeden z poniższych sposobów:

- zastosować pogrubienie, pochylenie, podkreślenie lub wypunktowanie albo numerowanie;
- zmienić wielkość elementów (maksymalnie 4 rozmiary)[125];
- wykorzystać różne kroje czcionek (nie więcej niż 3 rodzaje);
- wyróżnić kolorem (liczba stosowanych kolorów, które będą pojawiać się standardowo w interfejsie, nie powinna przekraczać czterech, ale można użyć również dodatkowych kolorów, zarezerwowanych do wykorzystania w wyjątkowych sytuacjach);
- stosować migające elementy (częstotliwość migania: 2-4 herców);
- zmieniać kolory migających obiektów (tekstu, grafiki);
- wykorzystać dźwięki[126].

Mając do dyspozycji wiele technik przyciągania uwagi użytkownika, można jednak popaść w przesadę, stosując je bez umiaru i osiągnąć wręcz przeciwny rezultat – nieczytelny ekran, przypominający migoczącą światłami choinkę lub witraż.

Bardzo dobrą techniką pomagającą użytkownikom organizować i przyswajać informacje jest umieszczanie obiektów w relacji wyżej-niżej. Dane znajdujące się na dole użytkownik będzie uważał za podsumowujące to, co jest powyżej. Jeżeli więc jakieś dane powstają w wyniku działania na innych danych, to powinny zostać umieszczone pod spodem. Jeżeli natomiast jakieś elementy są wspólne dla kilku danych, to powinny być umieszczone powyżej. Dodatkowo, dystans graficzny powinien odzwierciedlać dystans informacyjny[127].

DŹWIĘKI

W programach edukacyjnych i prezentacyjnych oraz grach dźwięk stanowi nieodłączną[128] część aplikacji, podnosząc jej jakość i zrozumiałość. W systemach informacyjnych dźwięki są rzadko stosowane i mają na celu jedynie uzupełnianie sprzężenia zwrotnego tam, gdzie metody graficzne mogłyby okazać się mało skuteczne (niewidoczne). Podobnie jak w przypadku kolorów,

---

123 Nie dotyczy to osób, które piszą od prawej strony do lewej, jak w języku arabskim i hebrajskim.

124 Ma to związek z funkcjonowaniem mózgu, który najlepiej zapamiętuje informacje z początku i z końca prezentowanego materiału, a najsłabiej te ze środka.

125 Jeśli chcemy przyciągnąć uwagę użytkownika, stosując w tekście duże litery, należy wziąć pod uwagę to, że wyrazy pisane wyłącznie dużymi literami czyta się o 10% wolniej.

126 B. Shneiderman: *Designing the User Interface*.... op. cit., s. 80.

127 J. Chabik: *Praktyka skutecznego programowania*.... op. cit., s. 110.

128 Oczywiście użytkownik może wyłączyć dźwięk, wybierając odpowiednią opcję.

dźwięk nie może być jedynym sposobem przekazywania informacji, ale powinien spełniać wyłącznie funkcję uzupełniającą.

Dźwięki mogą być pożyteczne, ale mogą również drażnić użytkowników. Krótkich piknięć używa się, aby zasygnalizować zakończenie zadania, pojawienie się komunikatu ostrzegawczego lub wystąpienie błędu. Może to okazać się przydatne w sytuacjach, gdy użytkownik jest zaabsorbowany czymś innym i nie patrzy na ekran. Jednak równie dobrze takie dźwięki mogą okazać się niepożądane. Użytkownik korzysta najczęściej z baz danych w czytelniach lub ośrodkach informacji naukowej, gdzie oprócz niego w pomieszczeniu znajdują się również inne osoby. Wielokrotnie powtarzające się dźwięki, sygnalizujące każdorazowo pojawienie się błędu, mogą wprawiać go w zakłopotanie. Żaden użytkownik nie chce, aby wszyscy wiedzieli o jego braku wprawy w posługiwaniu się systemem, co zdradzają dźwięki towarzyszące popełnianym błędom. Komfort pracy użytkownika i innych osób znajdujących się w tym samym pomieszczeniu, natychmiast się zmniejsza, a może to doprowadzić nawet do tego, że użytkownik nie będzie miał odwagi kontynuować pracy.

Dźwięki bywają więc zarówno potrzebne, jak i niepożądane, dlatego musi zawsze istnieć możliwość wyłączenia lub dostosowania natężenia dźwięków.

DANE WYJŚCIOWE

Po zakończeniu sesji wyszukiwawczej praca użytkownika z systemem jeszcze się nie kończy. Dane otrzymane w wyniku działania kwerendy skierowanej do systemu muszą zostać w jakiś sposób przeanalizowane, przetworzone i zapisane[129]. Ponieważ są one wyświetlane na monitorze, powinny się stosować do omówionych już reguł, dotyczących prezentacji informacji na ekranie.

Po otrzymaniu wyników użytkownik powinien móc przeszukiwać je wielokrotnie, formułując za każdym razem zmodyfikowaną kwerendę. Powinna istnieć możliwość dowolnego rozmieszczania wyników na ekranie, układania w hierarchię, grupowania, dodawania etykiet tekstowych (opisywania), a także zmiany ich wyglądu (koloru, czcionki, wielkości). W ten sposób dane wyjściowe zostaną odpowiednio przygotowane do wydruku lub prezentacji, bez potrzeby eksportowania ich do innego programu zajmującego się obróbką danych[130].

Zapisując wyniki wyszukiwania na dysku, system powinien udostępniać kilka formatów zapisu, np. w formie pliku tekstowego (rozszerzenie txt), dokumentu HTML (rozszerzenie htm lub html), dokumentu MS Word (rozszerzenie doc) lub Adobe Acrobat (rozszerzenie pdf). Pożytecznym rozwiązaniem byłaby wbudowana funkcja kompresowania danych i podziału pliku na części. Dzięki temu użytkownik, w przypadku gdy dane przeznaczone do zapisu nie mieściłyby się na jednej dyskietce, mógłby je skompresować, a jeśli skompresowany plik byłby wciąż większy niż pojemność dyskietki, to zostałby poprawnie podzielony na części, które zmieściłyby się na pojedynczych dyskietkach. Ponadto powinna również istnieć możliwość wysłania wyników wyszukiwania pocztą elektroniczną.

---

[129] Zapisane zostaną tylko relewantne informacje, które użytkownik uzna za potrzebne.

[130] Nie muszą to być zaawansowane funkcje formatowania, wystarczą podstawowe: zmiana czcionki i kolorów, otaczanie ramkami, zmiana położenia pól.

Najczęstszą operacją dokonywaną na zbiorze wynikowym jest drukowanie. Interfejs musi zapewniać użytkownikowi swobodę w wybieraniu rekordów i pól, które chce wydrukować. Powinien być także dostępny podgląd wydruku[131] oraz informacje dotyczące liczby stron zajmowanych przez rekordy zaznaczone przez użytkownika. Ważne jest również, aby można było łatwo przerwać drukowanie, gdyż zdarza się, że użytkownicy przez pomyłkę wydają polecenie drukowania tego, co nie jest im potrzebne, i tracą przez to czas oraz ponoszą dodatkowe koszty druku.

W omówionych zasadach projektowania wielokrotnie poruszaną kwestią było wyposażenie interfejsu w mechanizm pozwalający użytkownikom dostosowywać swobodnie do swoich potrzeb i umiejętności niemal wszystkie jego opcje i ustawienia, takie jak sposób komunikacji z systemem, zawartość pasków narzędzi, pozycje w menu, kolorystykę i dźwięki. Dzięki temu interfejs staje się częściowo adaptowalny, bo chociaż sam nie potrafi dostosowywać się automatycznie do preferencji użytkownika, to i tak możliwość wyboru ustawień przynosi wymierne korzyści.

Należy jednak być świadomym tego, że początkujący użytkownicy nigdy nie dokonują samodzielnych zmian w opcjach interfejsu. Nawet użytkownicy zaawansowani, którzy niezbyt często korzystają z systemu, nie chcą tracić czasu na dokonywanie ustawień. Rezygnują z tego, mimo że mogłoby to podnieść komfort ich pracy. Dlatego trzeba tak dobrać ustawienia, aby odpowiadały one jak najszerszej grupie użytkowników. Można to osiągnąć jedynie przez ciągłe testowanie i ulepszanie projektu.

Zasady i wskazówki dotyczące projektowania, zaprezentowane w tym artykule, powinny zwrócić uwagę projektantów na szereg aspektów, które muszą oni uwzględnić, aby tworzone przez nich interfejsy cieszyły się dobrą opinią wśród użytkowników. Ta opinia będzie zależeć od tego, w jakim stopniu interfejs spełni kryteria przyjazności dla użytkownika. Oczywiście, im bardziej skomplikowane zadanie, tym więcej czasu i umiejętności będzie potrzebował użytkownik, żeby je rozwiązać. Rzecz w tym, aby użytkownik nie tracił czasu na uczenie się obsługi interfejsu, ale skupił się wyłącznie na istocie problemu, co pozwoli mu szybciej i skuteczniej rozwiązywać zadania.

Mimo wielu wskazówek i zasad projektowania podawanych w obszernych publikacjach, interfejsy nadal są dalekie od doskonałości. Gdy graficzne interfejsy użytkownika pojawiły się po raz pierwszy w programach komputerowych sądzono, że ten typ interfejsu rozwiąże wszystkie problemy, jakimi były obarczone szeroko rozpowszechnione języki poleceń. Okazało się jednak, że wprawdzie zastosowanie obiektów graficznych poprawiło znacznie jakość interfejsów, ale nie w takim stopniu, aby obsługa komputera stała się dla wszystkich „łatwa, prosta i przyjemna". Nadzieje i oczekiwania wielu użytkowników, a także i samych projektantów, nie zostały do końca spełnione. Obecnie ogromne nadzieje wiąże się z zastosowaniem interfejsów w postaci języka naturalnego. Czy rzeczywiście język naturalny uczyni obsługę komputera bezproblemową? Czy w ogóle okaże się lepszy od interfejsu graficznego? Intensywne prace nad ulepszaniem interfejsów trwają nieustannie i wydaje się, że niezbyt odległa przyszłość przyniesie odpowiedzi na te pytania.

---

[131] Na podglądzie wydruku widać, jak zostaną wydrukowane dane, tzn. użytkownik może zobaczyć ich przestrzenne rozmieszczenie i odpowiednio ustawić marginesy.

Summary

The article deals with the principles of designing user interface, focusing mainly on the features of graphical user interface as a dominant form of contemporary human-computer interaction. As non-professionals, computer users come to rely on computer systems to perform more of their basic tasks, it is crucial to provide them with software that makes their work easier and more effective. To achieve this aim the interface should be designed in such a way that meets the requirements of a user friendly system: reliability, feedback, error handling, consequence, dialog, transparency, ease of use, user support, help, screen display, and output.