POLITECHNIKA LUBELSKA

PODRĘCZNIKI

Marek Miłosz

Ergonomia systemów informatycznych

Lublin 2014

Ergonomia systemów informatycznych

Podręczniki – Politechnika Lubelska

Człowiek – najlepsza inwestycja

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Publikacja współfinansowana ze środków Unii Europejskiej w ramach
Europejskiego Funduszu Społecznego

Marek Miłosz

Ergonomia systemów informatycznych

Politechnika Lubelska
Lublin 2014

Recenzenci:
dr hab. inż. Dariusz Czerwiński
dr hab. inż. Jerzy Montusiewicz, prof. PL

Redakcja i skład: Marek Miłosz

Książka przeznaczona dla studentów pierwszego i drugiego stopnia kierunku Informatyka

Publikacja dystrybuowana bezpłatnie.

Publikacja przygotowana i wydana w ramach projektu „Kwalifikacje dla rynku pracy - Politechnika Lubelska przyjazna dla pracodawcy" nr umowy POKL.04.03.00-00-035/12-00 z dnia 27 marca 2013 r. współfinansowanego ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

Publikacja wydana za zgodą Rektora Politechniki Lubelskiej

ISBN: 978-83-7947-106-5

Wydawca: Politechnika Lubelska
ul. Nadbystrzycka 38D, 20-618 Lublin
Realizacja: Biblioteka Politechniki Lubelskiej
Ośrodek ds. Wydawnictw i Biblioteki Cyfrowej
ul. Nadbystrzycka 36A, 20-618 Lublin
tel. (81) 538-46-59, email: wydawca@pollub.pl
www.biblioteka.pollub.pl
Druk: TOP Agencja Reklamowa Agnieszka Łuczak
www.agencjatop.pl

Elektroniczna wersja książki dostępna w Bibliotece Cyfrowej PL www.bc.pollub.pl
Nakład: 125 egz.

Spis treści

Wstęp

Systemy informatyczne wykorzystywane są przez użytkowników w codziennych działaniach, zarówno tych zawodowych, jak i prywatnych. Niestety, związane jest to zwykle z wielogodzinną pracą człowieka z oprogramowaniem i środkami technicznymi informatyki. Efektywność tej pracy ma wymierne znaczenie ekonomiczne. Od ergonomiczności środowiska i narzędzi komputerowych (zarówno softwarowych jak i sprzętowych) zależy produktywność pracowników, a także koszty ich działań.

System informatyczny w połączeniu z użytkownikiem staje sie złożonym systemem klasy maszyna-człowiek. Analiza jego funkcjonowania jest skomplikowanym i trudnym zadaniem, tym bardziej, że w jej toku należy uwzględnić także środowisko jego działania, czyli warunki zewnętrzne. Warunki te silnie wpływają na parametry funkcjonowania systemu maszyna-człowiek. Dostosowanie interfejsów oprogramowania do specyficznych mentalnych i fizycznych cech człowieka jest jednym z najważniejszych celów projektowania systemów informatycznych. Poprawny interfejs umożliwia bowiem efektywne wykorzystywanie oprogramowania, zmniejsza wysiłek człowieka związany z pracą, a w konsekwencji zmniejsza liczbę popełnianych błędów.

Książka prezentuje różne aspekty ergonomii systemów informatycznych, ze szczególnym uwypukleniem ergonomii interfejsów oprogramowania na tle ogólnego pojęcia ergonomii i modeli ją objaśniających. Przedstawiono w niej także najważniejsze elementy związane z projektowaniem interfejsów oprogramowania i oceną ich jakości. Problemy te, z uwagi na ich dużą złożoność i wieloaspektowość, zostały potraktowane dość przeglądowo. Takie podejścia nie zmniejsza, zdaniem autora, wagi przekazu o konieczności, przyczynach i metodach zadbania o wysoką jakość interfejsów oprogramowania. W książce przedstawiono także prawne przesłanki tworzenia poprawnych interfejsów. Zaprezentowano też cały szereg aktów normujących tą sferę produkcji oprogramowania.

Autor wyraża swoje podziękowanie recenzentom tej książki, profesorom dr hab. inż. Dariuszowi Czerwińskiemu i dr hab. inż. Jerzemu Montusiewiczowi, za wnikliwą analizę jej treści i bardzo szczegółowe uwagi. Ich uwzględnienie niewątpliwie przyczyniło się do znaczącej poprawy jej jakości. Podziękowania należą sie także koleżankom i kolegom z różnych instytucji, z którymi wspólna praca w wielu projektach, przyczyniła się do podniesienia poziomu wiedzy autora na temat treści prezentowanych w książce. W szczególności dziękuję Dr. Jean-Michel Adam, Dr. Sybille Caffiau oraz Brigitte Meillon z LIG Laboratory, Joseph Fourier University, Grenoble, Francja oraz Dr. Sergio Luján Mora i Dr. Cristina Cachero z University of Alicante, Alicante, Hiszpania, a także Łukaszowi Wójcikowi z Karo-Studio, Lublin.

Na kształt książki wpłynęły także wspólne prace, publikacje i dyskusje z koleżankami i kolegami z Laboratorium Analizy Ruchu i Ergonomii Interfejsów Instytutu Informatyki Politechniki Lubelskiej, w szczególności z Magdaleną Borys, dr inż. Małgorzatą Plechawską-Wójcik, i Maciejem Laskowskim. Ich ślady można znaleźć w wielu miejscach książki. Za co im serdecznie dziękuje.

Serdecznie dziękuje także wszystkim innym, powyżej niewymienionym, którzy w większym lub mniejszym stopniu przyczynili się do powstania tej książki.

Autor

1. Koncepcja ergonomii systemów

1.1. Ergonomia systemów – pojęcia i koncepcje podstawowe

Ergonomia (z języka greckiego jest połączeniem dwóch słów: praca (gr. εργον) i prawo, zasada, norma (gr. *νομος*) [25]) to dyscyplina naukowa (i praktyczna) zajmująca się organizacją pracy człowieka w układzie: człowiek-maszyna-warunki otoczenia. Ta interdyscyplinarna nauka ma na celu minimalizację kosztu biologicznego pracy (np. ryzyka wystąpienia wypadków lub chorób zawodowych) i zwiększenie jej efektywności [25]. Utylitarnym celem ergonomii jest polepszanie warunków pracy człowieka, poprzez dostosowanie ich do możliwości pracownika i na odwrót – poprzez właściwy dobór i przygotowanie (np. poprzez edukację) pracownika do danej pracy.

Zakres zainteresowań ergonomii zmienia się. Początkowo było to dostosowanie warunków pracy do możliwości biologicznych człowieka. Obecnie [26] jest to dostosowanie całej techniki do człowieka, zarówno w pracy jak i codziennym życiu (domu, szkole, transporcie, sklepie itd.). Z tych zmian wynika fakt, że ergonomia jest ważna dla wszystkich wytworzonych i używanych przez człowieka maszyn i urządzeń, niezależnie, czy jest to obrabiarka, traktor, komputer i jego oprogramowanie, czy też zwykły nóż lub czajnik. Ergonomiczny, czyli dostosowany do właściwości organizmu ludzkiego, może być (lub nie) młotek, klamka w samochodzie, długopis, klawiatura albo myszka.

Ergonomia jest nauką humanistyczną, czyli stawiającą w centrum zainteresowań i celów człowieka. Badania i ich zastosowanie ma na celu przede wszystkim ochronę człowieka i jego zdrowia. Ze względu jednak na to, że w układzie człowiek-maszyna-warunki otoczenia to człowiek jest najbardziej cenionym zasobem, w długim okresie jego ochrona zapewnia wzrost wydajności i jakości rezultatów jego pracy (produkcji) lub życia (np. straty czasu lub energii na codzienne czynności). Takie podejście przyczynia się do zapewnienia człowiekowi dominacji nad elementami materialnymi w jego życiu i pracy.

Ergonomia wchodzi w skład nauk o pracy człowieka i jej organizacji. Są to nauki takie jak:

- antropologia,
- higiena,
- prakseologia,
- organizacja i zarządzanie,
- socjologia pracy,
- psychologia,
- fizjologia człowieka,
- ekonomia.

Z pozycji teorii systemów, ergonomia bada zachowanie się złożonego systemu wyodrębnionego z otoczenia, ale zachowującego związki z nim. Badanie to ma charakter poznawczy i ma na celu rozpoznanie i analizę prawidłowości, które rządzą systemem wynikającym z jego elementów (oraz ich właściwości) i powiązań pomiędzy nimi. Systemy człowiek-maszyna są systemami skomplikowanymi – zawierają zarówno czynnik deterministyczny (środek sztuczny, techniczny – maszyna i jej przewidywalne właściwości), jak i losowy (człowiek z nie do końca poznanymi cechami, zachowaniami i zależnościami). System ten jest złożony, również z pozycji wzajemnego niedostosowania warunków, trybu oraz szybkości pracy człowieka i maszyny. Maszyny są zwykle optymalizowane (na etapie ich projektowania) pod kątem efektywności czasowo-kosztowej. Optymalizacja ta powoduje, że człowiek nie jest „kompatybilny" z warunkami pracy maszyn.

Ergonomia jako nauka ma na celu doprowadzenie do tej kompatybilności z zachowaniem priorytetu człowieka, ale też z optymalnym wykorzystaniem jego możliwości w danym systemie do realizacji konkretnej pracy.

Ergonomia bada także obciążenie człowieka wynikające z pracy. Obciążenie to może być fizyczne (praca mięśni) jak i psychiczne (praca mózgu). Wysiłek psychofizyczny związany jest z pokonywaniem uciążliwości, związanej z wykonywaniem pracy w określonych warunkach.

Złożony system człowiek-środowisko pracy (maszyny i środowisko ich użycia), zwany w uproszczeniu systemem człowiek-praca, charakteryzuje się następującymi właściwościami:

- realizuje zamierzone, celowe działanie;
- współpracuje (lub może współpracować) z innymi systemami;
- przeciwdziała pojawiającym się zakłóceniom (chroniąc cel działań);
- dostosowuje się do zmieniających się warunków (zmienia, doskonali, wykorzystuje wiedzę);
- zużywa się i wymaga konserwacji i odnowy.

System człowiek-praca funkcjonuje w wielu płaszczyznach, które można opisać poprzez:

- rodzaj pracy (fizyczna/psychiczna);
- warunki pracy (dynamika, pozycja ciała, środowisko, maszyny, ochrona itd.);
- czas pracy (długość, pora dnia, przerwy, cykliczność itd.).

Z praktycznego punktu widzenia ergonomia ma dwa kierunki działania związane z dostosowaniem środowiska pracy do możliwości człowieka:

- ergonomia korekcyjna,
- ergonomia koncepcyjna.

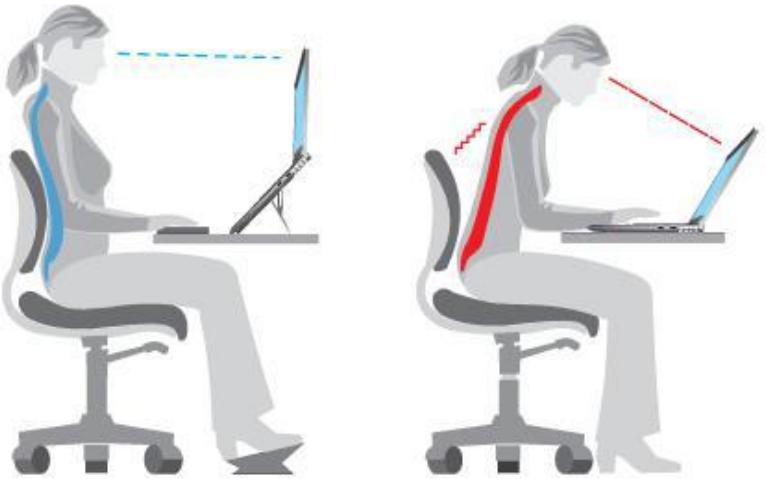
Ergonomia korekcyjna polega na analizie istniejących układów człowiek-praca i wprowadzaniu zmian usuwających wykryte usterki w eksploatacji maszyn. Ergonomia korekcyjna jest zatem dziedziną nauk stosowanych (praktycznych) i ma na celu [26]:

- poprawę warunków pracy człowieka (np. poprzez poprawę oświetlenia, zmniejszenia poziomu czynników szkodliwych, polepszenie mikroklimatu stanowiska pracy itd.);
- eliminację nadmiernych obciążeń fizycznych i psychicznych pracownika.

Przykład poprawy warunków pracy człowieka (poprawa takich parametrów jak: właściwa odległość od ekranu, właściwa pozycja kręgosłupa, wykorzystanie oparcia) poprzez zastosowanie niewielkich zmian (podpórka pod laptopa) w układzie człowiek-komputerowe stanowisko pracy ilustruje rys. 1.1.

Działania, realizowane w ramach ergonomii korekcyjnej są wymuszane przez różnorodne wady systemów człowiek-praca. Przyczyną tych wad jest [26]:

- wykorzystanie starych maszyn projektowanych bez uwzględnienia wymagań ergonomii pracy;
- przenoszenie błędów konstrukcyjnych (starych maszyn) na nowe maszyny, urządzenia i linie technologiczne – tzw. utrwalanie nieergonomicznych rozwiązań;
- oszczędnościowa polityka realizacji projektów inwestycyjnych (np. zmiany w specyfikacji urządzeń lub też ograniczenia w szkoleniach pracowników);
- nowe technologie eliminujące jedne problemy, ale wprowadzające inne, o których ich twórcy zapomnieli.



Rys. 1.1. Przykład korekcji warunków pracy człowieka w celu zwiększenia ergonomii
Źródło: htpp://drcrystaldraper.com

Ergonomia koncepcyjna wykorzystuje doświadczenia ergonomii korekcyjnej poprzez wymuszenie stosowania prawidłowych, z punktu widzenia ergonomii, rozwiązań już na etapie projektowania układów człowiek-maszyna. Ergonomia koncepcyjna wykorzystuje mechanizm norm i dobrych praktyk (które są odpowiednikiem norm w obszarach niesformalizowanych) na etapie projektowania systemów, w celu uzyskania ergonomicznych rozwiązań już na wstępnych etapach ich budowy. W trakcie projektowania systemów technicznych (przede wszystkim mieszanych: człowiek-maszyna) istotnym problemem jest występująca sprzeczność pomiędzy systemami w pełni technicznymi (np. linia automatyczna pakowania czegoś tam) a właściwościami psychomotorycznymi człowieka – pracownika wykonującego określoną pracę w tym systemie. Ergonomia koncepcyjna napotyka na szereg barier natury psychologicznej i organizacyjnej. Do najważniejszych z nich [26] należą:

- skłonność projektantów do powielania starych rozwiązań – jest to naturalna skłonność człowieka do użycia sprawdzonych w praktyce rozwiązań (nazywanych obecnie wzorcami projektowymi) w procesie tworzenia nowych rozwiązań; takie podejście jest w pełni uzasadnione chęcią ograniczenia ryzyka oraz niechęcią (obawą) do stosowania nowatorskich rozwiązań w inżynierii;
- brak chęci do tworzenia nowatorskich rozwiązań – przyczyny są podobne do wskazanych powyżej;
- trudności w utworzeniu (w tym i pozyskaniu) oraz zorganizowaniu pracy zespołu specjalistów o różnorodnych kompetencjach (tj. interdyscyplinarnego, przy dziedzinowym podziale kompetencji w większości organizacji biznesowych) i istniejących barierach komunikacyjnych pomiędzy członkami zespołu o różnym tle zawodowym (kompetencjach).

Część sygnalizowanych problemów wynika z odhumanizowaniu pracy często występującym w założeniach do różnorodnych przedsięwzięć, poprzez stosowanie na etapach projektowania nowych systemów technicznych wskaźników takich jak: maksymalny udźwig, wysokość podnoszenia maszyny, droga hamowania czy gabaryty. Te parametry są istotne, ale bardzo często zastosowanie ich związane jest z utratą przez projektanta postrzegania problemów ergonomii systemów.

Funkcjonowanie człowieka w otoczeniu (w tym też jego praca) charakteryzuje się całym szeregiem problemów. Wynikają one z biologicznej i społecznej natury człowieka. Do tych problemów należy zaliczyć:

- działania człowieka (niezależnie od tego czy jest to podjęcie pracy w przedsiębiorstwie, czy tylko np. pójście do kina) rodzą skutki prawne; we współczesnym społeczeństwie większość zasad funkcjonowania człowieka i jego otoczenia została mniej lub bardziej precyzyjnie określona w przepisach lub powszechnie uznawanych normach społecznych;

- człowiek ma ograniczenia biologiczne w działaniach, których przekroczenie wywołuje negatywne skutki (często nieodwracalne); obciążenie człowieka działaniami powoduje zmęczenie organizmu, co w istotny sposób ogranicza intensywność i długotrwałość działań; ograniczenia te są bardzo ważne i powinny być poznane oraz uwzględniane przy projektowaniu, a także przy wykorzystywaniu systemów człowiek-maszyna;
- ludzie (i ich właściwości) są zróżnicowani, niepowtarzalni i zmienni w czasie; częstokroć właściwości konkretnego człowieka nie są znane lub zbadane; problemy te z jednej strony utrudniają badania i standaryzację, a z drugiej wymuszają indywidualizację układu człowiek-maszyna; zróżnicowanie ma także dobrą stronę – umożliwia bowiem dobór ludzi do działań;
- efektywność (w tym np. wydajność) działań człowieka zależy od wielu czynników o naturze obiektywnej (np. wyposażenie techniczne) jak i subiektywnej (np. motywacja i stosunek do pracy); człowiek odpowiednio zmotywowany, wypoczęty fizycznie i psychicznie oraz wyposażony we właściwe środki techniczne, wykonuje działania bardziej efektywnie niżeli w innym przypadku; stan psychiczny człowieka, jego osobowość oraz chęć do pracy są równie ważne jak wyposażenie techniczne czy umiejętności;
- działania zespołowe wielu ludzi, ukierunkowane na osiągniecie jednego celu; współpraca i więzi pomiędzy ludźmi, adaptacja nowego pracownika, zapobieganie kryzysom i ich rozwiązywanie w takim układzie stają się elementem otoczenia systemu człowiek-maszyna.

Do każdej z grup problemów ergonomia powinna się odnieść. Wszystkie one bowiem wpływają na funkcjonowanie systemu człowiek-maszyna.

1.2. Modele zachowań człowieka

Ergonomia wykorzystuje rezultaty badań nauki o ludzkim poznaniu (ang. *Cognitive Sciences*), zwanej często nauką kognitywną lub kognitywistyką [25]. Obszarem badawczym tej dziedziny nauki jest poznawanie działania zmysłów, mózgu i umysłu człowieka. Poza działaniami typowo poznawczymi, celem kognitywistyki jest modelowanie i symulacja poznawczych działań ludzkich, a także takich obszarów jak język, uczenie się, percepcja, myślenie, świadomość, podejmowanie decyzji itd.

Kognitywistyka wraz z psychologią zaliczana jest do grupy nauk behawioralnych, czyli zajmujących się poznawaniem zachowań ludzkich.

Opracowano cały szereg modeli związanych z zachowaniami ludzkimi i sposobem ich realizacji. Do najważniejszych z nich należą następujące modele:

- hierarchii potrzeb Maslowa,
- ukierunkowanych działań Normana,

- Rasmussena,
- przetwarzania informacji MHP (ang. *Model Human Processor*)
- ICS (ang. *Interacting Cognitive Subsystems*).

Model hierarchii potrzeb Abrahama Maslowa

Model Maslowa usiłuje wyjaśnić zachowania i motywacje ludzi. Zakłada on, że potrzeby ludzi są zhierarchizowane, od tych najbardziej elementarnych, po te wyższego rzędu. Model ten, oprócz uporządkowania potrzeb ludzkich na pięciu poziomach, zakłada, że każdy człowiek dąży do zaspokajania potrzeb oraz, że potrzeby te są zaspokajane hierarchicznie. Oznacza to, że potrzeba wyższego rzędu jest zaspokajana (a w zasadzie nawet tylko pojawia się potrzeba jej zaspokojenia) dopiero wówczas kiedy potrzeba niższego rzędu jest przynajmniej częściowo zaspokojona. Hierarchia potrzeb tworzy tzw. piramidę Abrahama Maslowa – rys. 1.2.

Maslow dokonał podziału potrzeb według dwóch różnych kategorii. Są nimi potrzeby:

- niższego i wyższego rzędu,
- redukcji odczuwalnych braków i potrzeby rozwoju.

Rys. 1.2. Piramida potrzeb wg Maslowa
Źródło: htpp://motywowanie-pracownikow.eprace.edu.pl/530,Teorie_tresci.html

Maslow postawił także tezę (zwaną **zasadą rozwoju**), że zachowanie człowieka jest motywowane (lub też wynika) przez najniższą w hierarchii niezaspokojoną potrzebę. Teza ta wskazuje, że jeśli brak jest zaspokojenia np. potrzeb fizjologicznych, to jednostka (człowiek) nie myśli o bezpieczeństwie czy też potrzebie samorealizacji.

W odniesieniu do układu człowiek-praca, znaczenie poszczególnych poziomów potrzeb (rys. 1.2) jest dość oczywiste i można je zinterpretować w następujący sposób:

- potrzeby fizjologiczne, to: wynagrodzenie, warunki pracy, materialna pomoc pracodawcy;
- potrzeby bezpieczeństwa: zapewnienie stałego zatrudnienia, bezpieczeństwo stanowiska pracy, emerytury i renty, ubezpieczenie wypadkowe i chorobowe;
- potrzeby społeczne: kontakty międzyludzkie, atmosfera w pracy, wyjazdy integracyjne i inne wspólne działania rekreacyjne;
- potrzeby uznania, to przede wszystkim uznanie okazywane przez pracodawcę, drobne upominki i wyróżnienia, bycie „guru" w firmie, prestiżowe stanowisko lub pozycja;
- samorealizacja: dążenie do robienia kariery w organizacji (jest to raczej osobista potrzeba w oderwaniu do otoczenia czy zespołu, która bardzo często doprowadza do zmiany miejsca pracy).

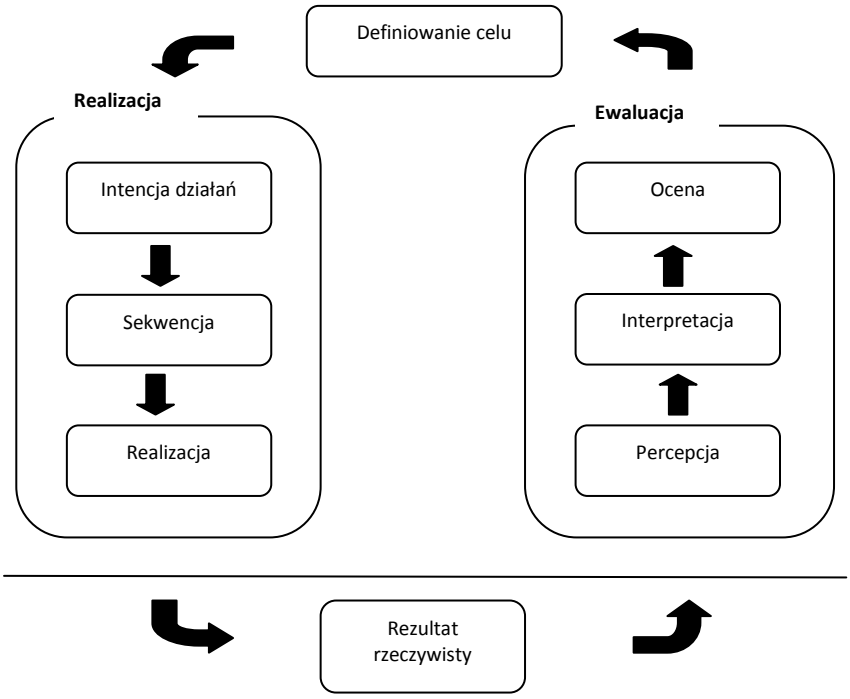
Model ukierunkowanych działań Normana

Model ukierunkowanych działań Normana opisuje działania człowieka używającego dowolnego systemu w określonym celu z dwóch pozycji: realizacji celów i oceny rezultatów. Model definiuje siedem etapów działań, podzielonych na trzy grupy (rys. 1.3):

- definiowanie celu działań;
- realizacja celu, tj. działania ukierunkowane na osiągnięcie celu (w tym etapy: intencja, sekwencja i realizacja działań);
- ocena (ewaluacja) rezultatów działań pod kątem spełnienia oczekiwań zdefiniowanych w stosunku do celu działań człowieka (etapy tej oceny są następujące: percepcja stanu rzeczywistego, interpretacja stanu rzeczywistego i ocena wyniku interpretacji).

W trakcie pracy człowiek realizuje te działania cyklicznie. Obrazuje to rys. 1.3.

Na etapie wykonania i ewaluacji pojawiają się często problemy. Na etapie wykonania pomiędzy intencjami a dopuszczalnymi możliwościami realizacji dochodzi do rozdźwięku. Pojawia się on w różnych sytuacjach, np. gdy intencje są proste, a trzeba je realizować przy pomocy skomplikowanych sekwencji złożonych zadań. Podobny problem występuje na etapie oceny rezultatów działań – odpowiedź na proste pytanie „czy jest dobrze?" może być trudna do udzielenia. W związku z tym, model ten znany jest jako *Norman's Gulfs of Execution and Evaluation*.



Rys. 1.3. Model ukierunkowanych działań człowieka Normana
Źródło: opracowanie własne na podstawie [25]

Model Normana wskazuje na potencjalne działania projektanta systemu, który powinien odpowiednio postrzegać działania człowieka, a więc ułatwiać formułowanie oraz realizację celów i postrzeganie stanu obecnego. W celu uniknięcia problemów, system człowiek-maszyna powinien być tak zaprojektowany, by użytkownik łatwo oceniał jego stan, łatwo określał właściwą sekwencję działań oraz otrzymywał potwierdzenia działań.

Model Rasmussena

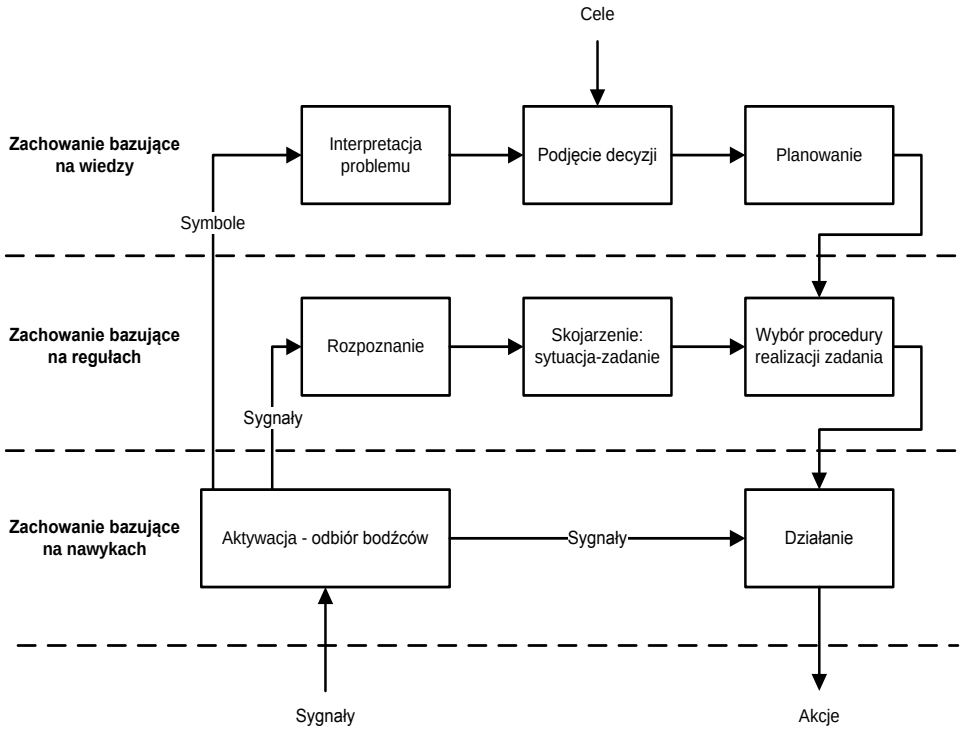
Model Rasmussena opisuje zachowanie się człowieka wobec problemu na trzech poziomach (teoria **SRK**):

- umiejętności, nawyki, odruchy (ang. *Skills*),
- reguł, przepisów (ang. *Rules*),
- wiedzy (ang. *Knowledge*).

Model SRK powstał, aby opisać procesy wykonywania zadań i podejmowania decyzji z poznawczego punktu widzenia. Model Rasmussena prezentuje rys. 1.4. Model ten opisuje procesy przetwarzania informacji w mózgu w różnych sytuacjach.

Zachowanie człowieka wynikające z nawyków po pojawieniu sie problemu jest niemal automatyczne. Wytworzone, opanowane i przećwiczone umiejętności pozwalają człowiekowi reagować (a więc rozpoznawać sytuację, podejmować decyzję i działanie) bardzo szybko (sekundy). Takie umiejętności jak chodzenie czy reakcja na pojawiające się okienko żądające np. potwierdzenia usunięcia pliku jest automatyczna. Często nawet człowiek nie uświadamia sobie, w jaki sposób podjął decyzję w danej sytuacji, bowiem zadziałał odruchowo.

Następny poziom zachowania człowieka to poziom reguł. W jego trakcie wykonanie zadania jest realizowane zgodnie z zapisanymi (lub tylko zapamiętanymi z poprzednich doświadczeń) zasadami – regułami. Działanie człowieka na tym poziomie sprowadza sie do przeglądu reguł i wyborze jednej z nich. Bardzo często człowiek jest wspomagany na tym poziomie zachowania poprzez różnego rodzaju listy sprawdzające (ang. *Checklists*). Proces poszukiwania i doboru reguł wywołany (wyzwalany) jest na skutek pojawienia się określonego sygnału (wyzwalacza) – rys. 1.4.



Rys. 1.4. Model Rasmussena działań człowieka bazujący na różnych poziomach przetwarzania informacji przez mózg
Źródło: opracowanie własne na podstawie [21]

Poziom reguł zwykle pojawia się w procesie realizacji zadań w sytuacji braku nawyków lub też jest wymuszany regulaminami. Przykładem mogą służyć działania przy pomocy listy kontrolnej załogi samolotu przed startem. W systemach informatycznych poziom reguł pojawia się dość często. Przykładem są różnego typu przewodniki, pomocniki, generatory, kreatory czy też mapy drogowe. Podejmowanie decyzji na poziome reguł trwa zwykle kilka minut.

W sytuacji nowej, która charakteryzuje się brakiem reguł, człowiek może rozwiązać problem przy pomocy swojej wiedzy. W trakcie tego procesu, definiowany jest w sposób jawny cel działań i powstają różne plany działań. Plany te są następnie oceniane pod kątem osiągania celu. W wyniku tych działań człowiek podejmuje decyzję wybierając najlepszy plan działań i go wykonuje, korzystając z zasad realizacji zadań. Działania bazujące na wiedzy zwykle trwają dość długo.

Model Rasmussena, poza uporządkowaniem działań człowieka i przypisaniem ich do różnych funkcji jego mózgu, umożliwia ocenę szybkości rozwiązania przez człowieka problemu oraz prawdopodobieństwa pojawienia się błędu w tym działaniu. Prawdopodobieństwo błędu ludzkiego rośnie wraz ze wzrostem jego poziomu zachowania. Zmieniają się także przyczyny błędów. Błędy na poziomie odruchów są najmniej prawdopodobne i powstają wskutek pomyłki w odbiorze sygnałów (np. błędny odczyt wartości), pomyłki działania (np. kliknięcie nie w tym miejscu, co zamierzone) oraz wad ergonomicznych (np. przesłonięcie przycisku). Błędy w działaniach na poziomie reguł powstają zwykle poprzez zaburzenie sekwencji działań (np. przypadkowe pominięcie jakiegoś kroku) lub nieprawidłowe rozpoznanie, bądź też skojarzenie sytuacji (np. błędna interpretacja lub przypisanie sygnału wyzwalacza do reguły). Pomyłki w sytuacji podejmowania decyzji przez człowieka, bazującej na jego wiedzy są najbardziej prawdopodobne. Wynikają one z błędnej interpretacji sytuacji (np. nieprawidłowa interpretacja metafor interfejsu) i braku wiedzy (np. braku analogów lub ich nieprawidłowe użycie, błędy rozumowania itd.).

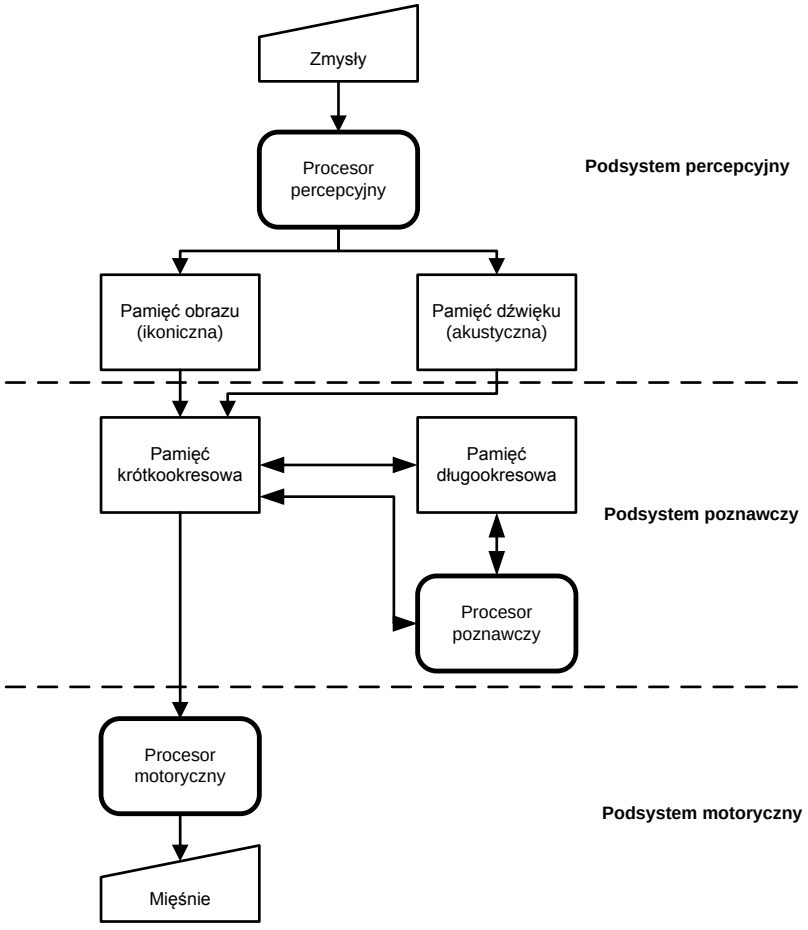
Z modelu Rasmussena wynikają następujące wskazówki projektowania systemów informatycznych:

- należy unikać zadań wymagających wiedzy użytkownika, a jeśli one występują, to należy je wspomagać i przenieść na poziom reguł (poprzez zastosowanie np. wizualizacji danych, automatyzacji ich interpretacji i wnioskowania), podpowiadając użytkownikowi rozwiązania;
- zadania wykorzystujące reguły należy przenieść na poziom nawyków, poprzez zastosowanie automatyzacji powtarzalnych czynności, stosowanie kreatorów oraz systemów pomocy kontekstowej, podpowiadających automatycznie kolejne kroki lub możliwości;
- wykorzystać w jak najszerszym zakresie zadania, których realizacja wymaga odruchów.

Model Human Processor

Model Human Processor (**MHP**) zwany niekiedy modelem przetwarzania informacji przez człowieka (ang. *Human Information Processor*) nawiązuje silnie do architektury współczesnego komputera. Opisuje on człowieka, jako sekwencyjny układ procesorów przetwarzania informacji współpracujących z różnymi pamięciami i układami wejścia-wyjścia. Model składa się z trzech podsystemów (rys. 1.5), zwanych jako:

- percepcyjny;
- poznawczy;
- motoryczny, czyli wykonawczy.



Rys. 1.5. Model przetwarzania informacji przez człowieka (MHP)
Źródło: opracowanie własne na podstawie [25]

W podsystem percepcyjny wchodzą układ wejścia (czyli zmysły człowieka takie jak wzrok, słuch czy dotyk) oraz procesor percepcyjny i pamięć. Przy czym model rozróżnia pamięć obrazu od pamięci dźwięku. Dane pochodzące z otoczenia, po pobraniu (zmysły) i odpowiednim przetworzeniu (percepcja) są przechowywane w pamięci, która odpowiada funkcjonalnie pamięci buforowej w systemach komputerowych. Podsystem poznawczy, składa się natomiast z procesora poznawczego (kognistywicznego) oraz pamięci krótko- i długookresowej. Procesor poznawczy sprawia, że ludzie „myślą". Procesor motoryczny wprawia w ruch mięśnie, które realizują jako organ wykonawczy, polecone działania.

Procesory modelujące różne funkcje mózgu ludzkiego, mają różne czasy (cykle) działania – analogicznie do procesorów w systemie komputerowym. Czasy te zostały określone na podstawie badań psychologicznych. W związku z szeregową architekturą, czas przetwarzania danych w takim sekwencyjnym układzie składa się sumy czasów działania poszczególnych procesorów. Czasy te zależą od procesora (średnio wynoszą one dla procesorów percepcyjnego/poznawczego/motorycznego: 100/70/70 ms), właściwości konkretnego człowieka oraz od warunków pracy (działania). Poza tym szybkość działania całego układu zależy także od szybkości postrzegania bodźców zewnętrznych (układ Zmysły na rys. 1.5) oraz od rodzaju pracy procesora poznawczego. Procesor ten wypracowuje bowiem decyzje w różnych trybach, określanych jako SRK (zgodnych z modelem Rasmussena). Średnio cykl pracy trzech procesorów (przy użyciu najszybszego – odruchowego działania w procesorze poznawczym) wynosi 240 ms. Czas ten wyznacza czas reakcji człowieka na bodziec zewnętrzny. Jest to bardzo ważny parametr w układzie człowiek-maszyna. Nie można bowiem liczyć na szybszą reakcję człowieka w tym układzie ze sprzężeniem zwrotnym: działania-kontrola. Działania niewymagające kontroli (np. powtarzane wielokrotnie działania operatora) mogą osiągać czas cyklu działania procesora motorycznego, czyli średnio 70 ms.

Model ICS

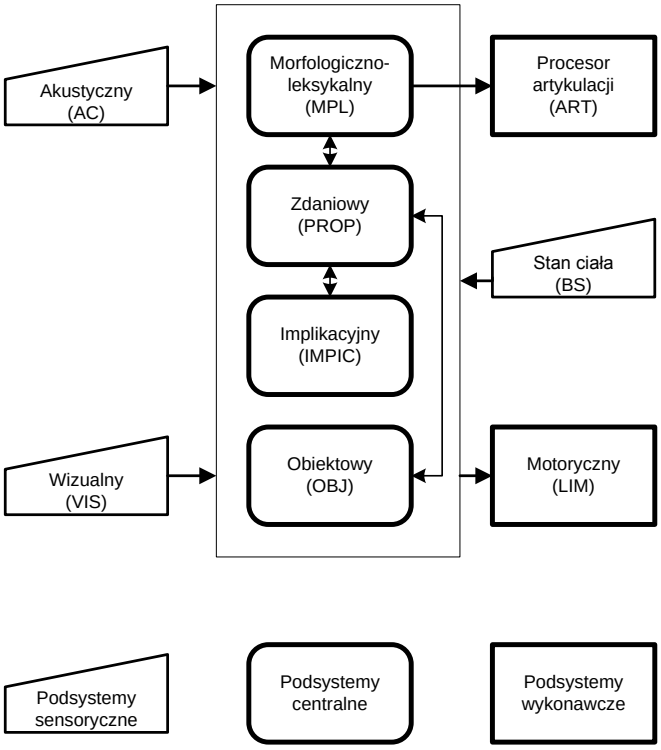
ICS **(ang. *Interacting Cognitive Subsystems*)** to architektura mentalna (psychiczna) człowieka – model opracowany w celu analizy zjawisk poznawczych, zachodzących w mózgu człowieka ze szczególnym uwzględnieniem procesów pamięciowych i przetwarzania informacji. Model ten wykorzystuje fakt przetwarzania (tj. rozumienia i tworzenia) języka mówionego i pisanego. Model ten znalazł wiele różnych zastosowań w analizie i objaśnieniu procesów poznawczych. Składa się na niego dziewięć różnych podsystemów, pogrupowanych w trzy kategorie (rys. 1.6):

- podsystemy sensoryczne, czyli pozyskujące informację;
- podsystemy wykonawcze, czyli generujące rezultaty działań;
- podsystemy centralne, realizujące przetwarzania informacji opisowej i znaczeniowej.

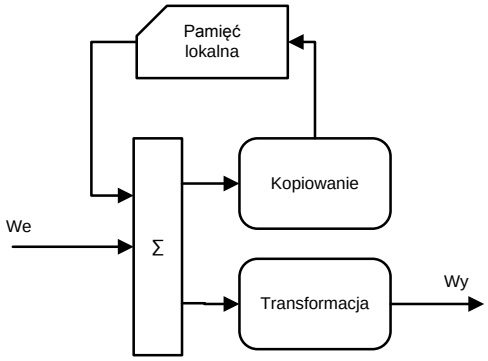
Trzy podsystemy sensoryczne to: akustyczny (ang. *Acoustic, AC*), wizualny (ang. *Visual, VIS*) oraz stan ciała człowieka (ang. *Body State, BS*). Reprezentują one podsystemy pozyskiwania przez człowieka informacji z otoczenia, tak zewnętrznego, jak i wewnętrznego (tj. pochodzące od ciała, np. ból). Odbierają pierwotne informacje od słuchu, wzroku i pozostałych zmysłów (tj. nacisk, smak, zapach, temperatura itd.).

Dwa systemy wykonawcze odwzorowują proces generowania mowy (ang. *Articulatory, ART*) oraz proces ruchu, czyli motoryczny (ang. *Limb, LIM*). Systemy te kontrolują mięśnie człowieka.

Na centralny podsystem składają się cztery główne podsystemy: przetwarzania informacji obiektowej (ang. *Object, OBJ*) oraz morfologiczno-leksykalnej, czyli słownej (ang. *Morphonolexical, MPL*) oraz jej rozumienie na dwóch poziomach: zdaniowym (ang. *Propositional, PROP*) i implikacyjnym (ang. *Implicational, IMPLIC*).



Rys. 1.6. Model architektury mentalnej człowieka (ICS)
Źródło: opracowanie własne na podstawie [1]



Rys. 1.7. Model struktury podsystemu w modelu ICS
Źródło: opracowanie własne na podstawie [1]

Każdy z podsystemów ma swój sposób kodowania informacji, lokalną pamięć i identyczną strukturę, przedstawioną na rys. 1.7. Sygnały wyjściowe są odpowiednio zakodowane w zależności od podsystemu. Są one kopiowane do pamięci lokalnej podsystemu i jednocześnie z pamięci tej przywoływane są zapamiętane informacje. Taka „mieszanka" poddawana jest przekształceniu do informacji wyjściowej, zakodowanej w sposób adekwatny do podsystemu, do którego jest kierowana. Przetwarzanie takie zachodzi równolegle dla dużej ilości informacji wejściowych (We z rys. 1.7 jest wektorem o dużej liczbie elementów) i wykorzystuje istniejącą pętlę informacyjnego sprzężenia zwrotnego: kopiowanie-zapamiętywanie-pozyskiwanie.

Model ICS w jasny sposób rozróżnia typy informacji pod kątem różnych sposobów ich przetwarzania i wykorzystywania przez człowieka. Jednocześnie model ten wskazuje na ograniczenia w zakresie pozyskiwania i wykorzystywania informacji – zakres ten wynika z biologicznych właściwości ludzkich sensorów i organów wykonawczych.

1.3. Modele opisujące współpracę człowieka z interfejsem oprogramowania

System człowiek-komputer jest specyficznym układem człowiek-maszyna, który charakteryzuje się interakcją związaną z wizją – monitorem ekranowym i urządzeniem, pozycjonującym kursor, oraz klawiaturą, jako drugim typowym urządzeniem wejścia. W analizie i opisie interfejsu systemu człowiek-komputer wykorzystuje się prawo ***Fittsa***, model ***KLM*** (ang. *Keystroke-Level Model*) i prawo ***Hicka***.

Prawo Fittsa

Prawo Fittsa (ang. *Fitts's Law*) opisuje działania, realizowane przez człowieka w celu wskazania określonego obszaru docelowego, którym jest obiekt graficzny na ekranie monitora. Obiektem tym może być dowolny wyróżniający się obszar (np. przycisk, pole, napis czy ikona). Wskazanie to może być realizowane przy pomocy różnych technik. Techniki te mogą wykorzystywać urządzenia techniczne (np. pióro świetlne, mysz, trackball) lub też naturalne (zwykle: palec, lub inny organ ciała ludzkiego). Możliwe są też techniki mieszane, np. wzrok i odpowiednie urządzenie techniczne, rozpoznające miejsce wskazania. We wszystkich technikach wskazania ważne jest to, że realizowane czynności kontrolowane są wzrokiem. Poza motorycznym charakterem działań, ważne są także ich elementy poznawcze. Zwykle działania takie poprzedzone są postrzeganiem, wyszukaniem i rozpoznaniem obiektu ekranowego. W trakcie realizacji działania człowiek wykorzystuje pętlę informacyjnego sprzężenia zwrotnego sterowania: postrzeganie-rozpoznanie-działanie motoryczne. W trakcie działania wykorzystywana jest większość procesorów ludzkich z modelu ICS albo MHP.

Model Fittsa pozwala określić czas, jaki jest niezbędny do osiągnięcia obszaru docelowego, rozpoczynając z punktu startowego (tj. położenia kursora lub początkowej koncentracji wzroku) przy pomocy stosowanego urządzenia wskazującego. Schemat problemu przedstawia rys. 1.8.

Rys. 1.8. Model geometryczny osiągania celu na ekranie
Źródło: opracowanie własne na podstawie [25]

Czas realizacji działania wskazania obiektu opisuje formuła (zwana niekiedy formułą Shannona):

$$T = a + b \cdot \log_2\left(c + \frac{D}{S}\right) \tag{1.1}$$

gdzie:

- T – całkowity czas (w sekundach) wymagany do realizacji zadania wskazania obiektu docelowego;
- a – stała opisująca czas (w sekundach) reakcji niezbędnej do rozpoczęcia ruchu – opóźnienie rozpoczęcia ruchu; wynika on z właściwości urządzeń technicznych oraz czasu niezbędnego człowiekowi do zareagowania;
- b – stała czasowa związana z prędkością ruchu urządzenia wskazującego, sterowanego przez człowieka mierzona w jednostce czasu (s);

c – stała zależna od środowiska (człowiek-maszyna) wykonywania ruchu i przyjmująca wartości 0, 0,5 lub 1; dla systemów informatycznych zwykle jest równa 1;

D – odległość od punktu początkowego (Start na rys. 1.8) do środka obiektu docelowego w jednostkach miary odległości (np. piksel lub cm);

S – szerokość obiektu docelowego w jednostkach miary odległości (np. piksel lub cm) w linii ruchu punktu wskazującego (kursora lub punktu fiksacji oczu).

Wartości *a, b* i *c* zależą od sposobu realizacji zadania i urządzenia technicznego służącego do wskazywania celu. Zatem dla danego sposobu realizacji zadania wskazania celu (tj. dla stałych wartości *a* i *b*), czas jego realizacji zależy od indeksu (stopnia) trudności obiektu (ang. *Index of Difficulty*) wyznaczanego jako wartość: $\log_2\left(c+\frac{D}{S}\right)$.

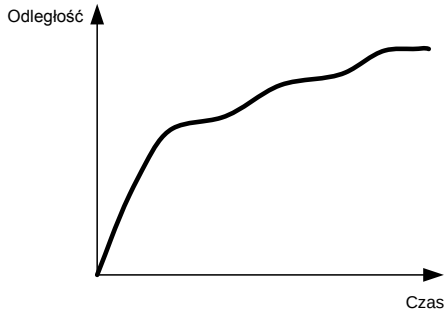
Postać indeksu trudności wynika z analizy zachowania się człowieka. Z wykorzystaniem modelu MHP można go objaśnić następująco: po rozpoznaniu zadania (praca dwóch procesorów: percepcyjnego i poznawczego z rys. 1.5), procesor motoryczny (patrz rys. 1.5) wydaje polecenie mięśniom człowieka do przesunięcia wskaźnika (np. kursora przy pomocy myszy lub punktu wskazywanego palcem). Mięśnie przesuwają wskaźnik do pewnego miejsca, o odległość D. Z różnych powodów (błędy wykonania, zależne i niezależne od człowieka) wskaźnik osiąga cel z błędem ε, tj. osiąga punkt odległy od celu o εD. Błąd ten może być ujemny („niedociągnięcie" wskaźnika do celu), jak i dodatni („przeciągnięcie"). Procesor percepcyjny rozpoznaje położenie, procesor poznawczy rozpoznaje błąd i wypracowuje decyzję, którą procesor motoryczny przekształca do sygnałów sterujących mięśniami. Wskaźnik przemieszczany jest ponownie z błędem ε, tj. na odległość od celu: $\varepsilon^2 D$. Proces się powtarza, geometrycznie zmniejszając błąd. Po n iteracjach wskaźnik osiąga cel, tj. $\varepsilon^n D \le S/2$. Całkowity czas realizacji zadania wynosi zatem: $T = nT_i$, gdzie T_i jest czasem jednej iteracji (jest to suma czasów działania procesora percepcyjnego, poznawczego oraz motorycznego z uwzględnieniem szybkości działania urządzeń). Rozwiązując równanie: $\varepsilon^n D = S/2$ i pozyskując z niego n, po czym wstawiając go do równania na całkowity czas osiągnięcia celu, po stosownych przekształceniach uzyskuje się formułę:

$$T = a + b \cdot \log_2\left(\frac{D}{S}\right) \tag{1.2}$$

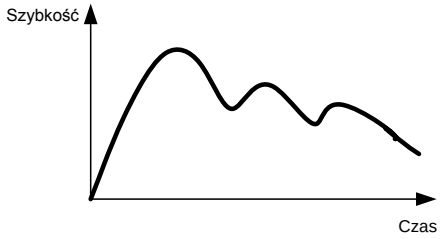
gdzie: $b = -T_i / \log_2(\varepsilon)$, przy czym $0 < \varepsilon < 1$, tak więc $b > 0$.

Typowa trajektoria przemieszczania wskaźnika przedstawiona została na rys. 1.9, a szybkość przemieszczania – na rys. 1.10. Kolejne fale w szybkości

przemieszczania wynikają z „obróbki” przez procesory człowieka sprzężenia zwrotnego: postrzegania i analizy osiągniętego, a wymaganego położenia wskaźnika i wypracowania decyzji oraz jej realizacji.



Rys. 1.9. Trajektoria ruchu wskaźnika przesuwanego przez człowieka na ekranie
Źródło: opracowanie własne na podstawie [25]

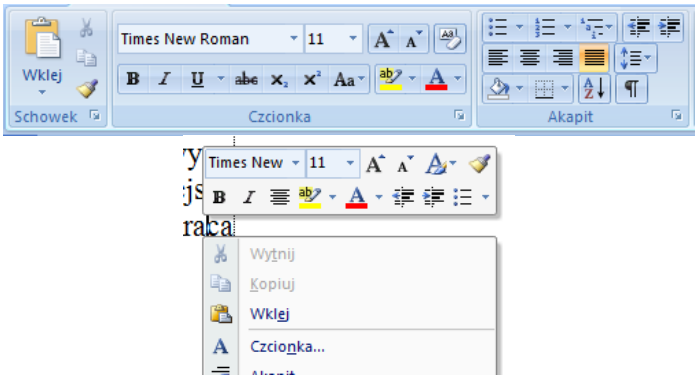


Rys. 1.10. Szybkość ruchu wskaźnika przesuwanego przez człowieka na ekranie
Źródło: opracowanie własne na podstawie [25]

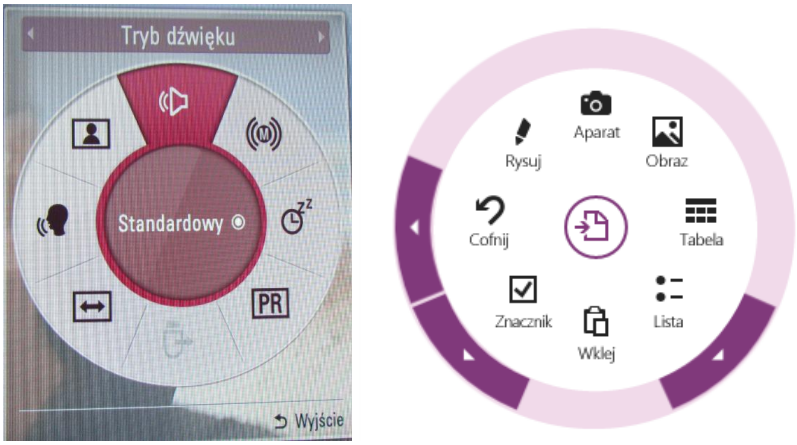
Prawo Fittsa, opracowane pierwotnie na potrzeby przemysłu lotniczego, ma głęboki wpływ na interfejs człowiek-komputer. Im dalej bowiem znajduje się przycisk od aktualnego położenia kursora myszy (czyli D), tym większy będzie czas jego przemieszczenia. Wynika stąd konieczność zbliżania sterujących elementów interfejsu do siebie, a z kształtu indeksu trudności obiektu wynika konieczność zwiększania obiektów (czyli S).

Poza tym, z prawa Fittsa wynika, że wzrost rozdzielczości (i idący za nim wzrost ich fizycznej wielkości) stosowanych monitorów ekranowych powoduje spadek efektywności pracy z nimi. Obserwowany wzrost wielkości obszarów roboczych wymusił m.in. zmianę długich pasków narzędziowych na wstążki lub prostokątne grupy ikon [25]. Odnalezienie i dotarcie wskaźnikiem do właściwej ikony na długim pasku narzędziowym jest długotrwałe. Wprowadzenie prostokątnego obszaru (kontenera) z ikonami (wstążki bądź też panelu z ikonami

w menu podręcznym – rys. 1.11) skróciło ten czas. Rozmiar wstążki jest większy (parametr S we wzorze (1.1)) niżeli w pasku narzędziowym, a pojawiający się prostokąt z ikonami w pobliżu położenia kursora zmniejsza wartość parametru D. Przy niezmienionych innych parametrach, zgodnie z prawem Fittsa skraca to czas dotarcia do właściwej ikony. Analogiczne implikacje prawa Fittsa spowodowały opracowanie alternatywnych do menu liniowego oraz kaskadowego sposobów wyboru opcji z listy – menu promieniowego (ang. *Pie Menu*). Droga dojścia do opcji w menu promieniowym (rys. 1.12) jest znacznie krótsza niźli w liniowym.



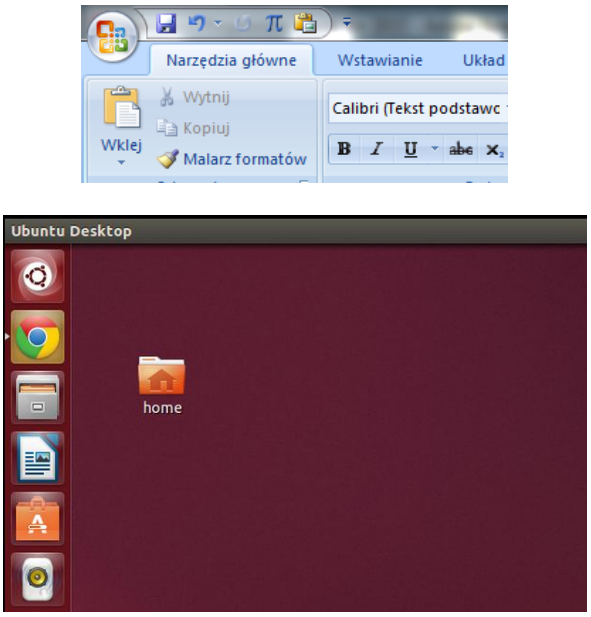
Rys. 1.11. Wstążka i kontener (panel) z ikonami w menu podręcznym
Źródło: opracowanie własne



Rys. 1.12. Menu promieniowe przyspieszające dostęp do opcji
Źródło: opracowanie własne – TV LG i MS OneNote

Z prawa Fitssa wynika też specyficzna rola krawędzi ekranu. Krawędź zatrzymuje bowiem kursor. Zatem osiągnięcie jej jest najszybsze z możliwych – nie wymaga długotrwałego, iteracyjnego pozycjonowania kursora. Stąd też na krawędziach pojawiają się często specyficzne elementy sterujące. Pasek z menu, oddzielony od krawędzi ekranu paskiem z nazwą okna, nie jest najlepszym rozwiązaniem pod kątem efektywności dostępu. Krawędź w połączeniu z innym działaniem (np. najechaniem wskaźnikiem bądź też przeciągnięciem okna na krawędź) umożliwia człowiekowi bardzo szybkie wywołanie określonej akcji. W systemie Windows fakt ten zaowocował umieszczeniem paska „Szybki dostęp" niektórych aplikacji w takim miejscu, by po maksymalizacji okna był on na krawędzi ekranu – rys. 1.13. Częściowo likwiduje to niefortunne umieszczenie nazwy okna na jego górnej krawędzi w tym systemie.

Niektóre działania użytkownika realizowane przy pomocy wskaźnika charakteryzują się większym marginesem błędu niźli wskazanie konkretnego obiektu. Do takich przykładów zalicza się wskazanie tunelowe, tj. ruch wskaźnika w obrębie określonego, zwykle dość dużego, prostokątnego obszaru na ekranie. Sytuacja taka występuje np. w trakcie poruszania się wskaźnikiem w obrębie menu rozwijanego. Schemat poruszania się wskaźnikiem w tunelu przedstawia rys. 1.14.



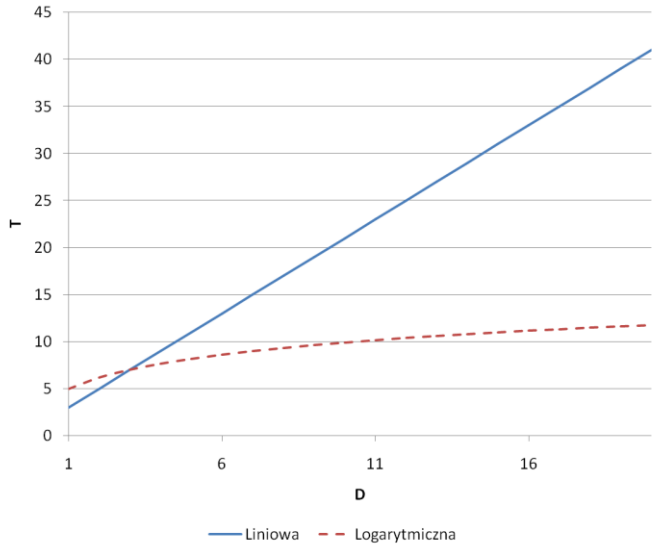
Rys. 1.13. Pasek szybkiego dostępu w aplikacjach MS Office oraz wykorzystanie krawędzi ekranu w Ubuntu
Źródło: opracowanie własne

Rys. 1.14. Model geometryczny ruchu wskaźnika w tunelu
Źródło: opracowanie własne na podstawie [25]

Czas realizacji działania, związanego z przemieszczaniem wskaźnika w tunelu opisuje formuła:

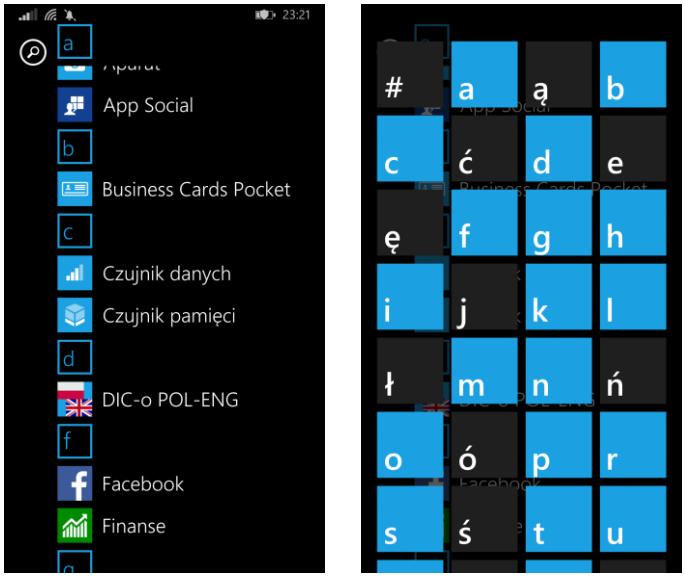
$$T = a + b \cdot \frac{D}{S} \tag{1.3}$$

Zależność czasu realizacji T od odległości od celu D jest zatem liniowa. Czas ten jest zatem ekspotencjalnie dłuższy od czasu wskazywania określonego obiektu, bowiem w takim przypadku wzór (1.1) wskazuje na zależność logarytmiczną. W związku z tym, nieefektywne są układy menu z dużą liczbą nieuporządkowanych opcji wybieranych liniowo. Przykładowe zależności pomiędzy odległością D a czasem realizacji T wskazania dla modelu logarytmicznego (wzór (1.2)) i liniowego (1.3) przedstawia rysunek 1.15.



Rys. 1.15. Zależności czasu realizacji od odległości w modelu logarytmicznym i liniowym
Źródło: opracowanie własne

Praktyczną implikacją tego zjawiska jest stosowanie algorytmów przyspieszających dostęp do opcji w dużych, liniowych układach. Jednym z nich jest algorytm wyszukiwania w uporządkowanym (np. alfabetycznie) zbiorze opcji, połączony z liniowym wyborem w zmniejszonym ich podzbiorze. Takie rozwiązanie jest szeroko stosowane tam, gdzie liniowy dostęp jest wymuszany (np. czynnikami technicznymi takimi jak wielkość ekranu) – rys. 1.16. Niestety wykorzystanie uporządkowania nazw opcji wywołuje inne problemy, a mianowicie konieczność zapamiętywania przez człowieka nazw opcji i uruchomienie procesora poznawczego (model MHP) czy też realizacji zachowań bazujących na wiedzy (model Rasmunssena), które wnoszą bardzo duże opóźnienie w wykonanie zadania.



Rys. 1.16. Tunelowy wybór opcji w aplikacji mobilnej i jego przyśpieszenie z wykorzystaniem alfabetycznego uporządkowania opcji – system Windows Phone 8
Źródło: opracowanie własne

Prawo Fitssa, pierwotnie odkryte przy badaniach interfejsów niekomputerowych, znajduje także zastosowanie w projektowaniu efektywnych interfejsów sprzętowych. Przykładem może być stworzenie klawiatury numerycznej, rozszerzającej (i dublującej funkcjonalnie) klawiaturę, w celu przyśpieszenia wprowadzania cyfr i manipulowania kursorem przy pomocy klawiszy. Układ klawiatury naśladuje układ przycisków kalkulatora, który jest odmienny od układu telefonu – rys. 1.17. Klawiatura numeryczna może być rozszerzeniem klawiatury bądź oddzielnym urządzeniem – rys. 1. 18.

Rys. 1.17. Układ klawiszy na klawiaturze numerycznej i w telefonie
Źródło: opracowanie własne

Rys. 1.18. Klawiatura numeryczna jako oddzielne urządzenie
Źródło: http://pl.wikipedia.org/wiki/Klawiatura_numeryczna#mediaviewer/File:Numpad.JPG

Model KLM

Model KLM (ang. *Keystroke-Level Model*) jest metodą wywodzącą się z techniki o nazwie GOMS (ang. ***G**oals, **O**perators, **M**ethods, **S**election Rules*), umożliwiającą modelowanie zachowań człowieka w układzie: człowiek-komputer. Model ten opisuje i pozwala analizować złożoność zachowań użytkownika w komunikacji z systemami interaktywnymi.

W metodzie KLM zakłada się, podobnie jak w GOMS, posługiwanie się tzw. operatorami, które opisują atomowe, niepodzielne czynności (np. naciśnięcie przycisku klawiatury lub myszy). Zastosowanie tej metody i modelu działań użytkownika pozwala określić całkowity czas niezbędny do osiągnięcia danego celu (tj. realizacji zadania) przy pomocy klawiatury lub

podobnych urządzeń. Takim celem może być np. skopiowanie czy usunięcie ciągu znaków.

Podstawowym założeniem wykorzystania metody KLM do modelowania działań użytkownika jest możliwość dekompozycji zadania do postaci działań elementarnych, odpowiadających zdefiniowanym operatorom. W praktyce może się to okazać trudne, bowiem każde nietrywialne zadanie dekomponuje się na dużą liczbę działań elementarnych.

Każdemu operatorowi przypisuje się średni czas wykonania opisywanego elementarnego działania (czynności), uzyskany na podstawie danych empirycznych. Zsumowanie czasów cząstkowych, odpowiadających kolejnym, szeregowo wykonywanym czynnościom, pozwala określić szacunkowy, średni czas potrzebny do realizacji całego zadania.

W modelu KLM definiuje się sześć rodzajów operatorów, czyli atomowych-elementarnych działań, opisujących możliwe interakcje ze strony człowieka-użytkownika. Należą do nich [25]:

K – pojedyncze uderzenie w klawisz klawiatury (w przypadku kombinacji klawiszy każdy klawisz należy liczyć oddzielnie); czas realizacji tej czynności mieści się w przedziale 0,12-1,2 s (średnia i zalecana wartość wynosi 0,28 s);

P – wskazanie celu na ekranie przy pomocy kursora myszy; czas ten zależny jest od odległości oraz wielkości wskazywanego obszaru; przypisywany czas to wielkość z przedziału 0,8-1,5 s (średnio – 1,1 s);

B – wciśnięcie lub zwolnienie przycisku myszy; średni czas realizacji to 0,1 s;

BB – kliknięcie przyciskiem myszy, które jest liczone jako dwa operatory typu B, a w związku z tym przypisana do danego operatora wartość czasu realizacji wynosi 0,2 s;

H – ułożenie rąk na klawiaturze, bądź myszy (przeniesienie ręki z jednego urządzenia na drugie); ruch ten wykonywany jest często, a taka czynność jest stosunkowo szybka – szacowany czas to 0,4 s;

M – akt rutynowego działania lub percepcji – akcja o charakterze umysłowym; czas potrzebny na wykonanie tej czynności może być bardzo zróżnicowany w zależności od osoby, jej umiejętności, zdolności percepcyjnych oraz sytuacji w jakiej realizowane jest zadanie; czas tego działania mieści się w przedziale 0,6-1,35 s; natomiast średnia empiryczna wartość wynosi 1,2 s; czas tego działania zależy od typu użytkownika – dla eksperta (który realizuje rutynowe działania) wynosi on znacznie mniej niż dla początkującego (w tym przypadku jest to działanie bazujące na regułach – rys. 1.4); w sytuacjach bardziej złożonych, ekspert może działać bazując na regułach, a początkujący poszukiwać rozwiązania, bazując na wiedzy – czynnik ten różnicuje oczywiście czas realizacji działania;

R – oczekiwanie na odpowiedź systemu; w wielu przypadkach operator ten może być pominięty. Jednak w przypadku przetwarzania dużej ilości danych konieczne staje się jego uwzględnienie (długotrwałe operacje dyskowe – np. zapis pliku na dysku, realizacja zapytań do serwera bazy danych lub serwera www); czas ten silnie zależy od architektury i warunków przetwarzania informacji i należy określać go indywidualnie dla konkretnego systemu.

Ocena łącznego czasu realizacji zadanego scenariusza interakcji człowieka z systemem informatycznym polega na:

- dekompozycji procesu na działania proste, wykonywane sekwencyjnie i niekiedy z powtórzeniami;
- klasyfikacji tych działań i przypisanie im odpowiednich operatorów;
- przypisaniu czasów realizacji poszczególnych operatorów (uwzględniając specyficzne warunki, w tym parametry pracy systemu informatycznego);
- obliczeniu czasu realizacji danego scenariusza.

Wyznaczony w ten sposób czas jest oczywiście szacunkiem, tj. wartością z definicji niedokładną. Pozwala on jednak porównywać scenariusze i warianty interfejsów, a także wyznaczać np. normy czasowe operacji realizacji zadań z wykorzystaniem systemu informatycznego.

Model KLM dla konkretnego zadania w interfejsie wymaga opracowania scenariusza jego wykonania z podziałem na poszczególne akcje, dodanie działań umysłowych (**M**) i ewentualnych opóźnień systemu (**R**), a następnie przeprowadzenie obliczeń. Opóźnienia wnoszone przez system są konkretnym, łatwo wyznaczalnym parametrem. Zależą one od systemu informatycznego i zadania, które użytkownik ma wykonać. Działania umysłowe są związane z dużą liczbą różnych czynników, które zależą lub nie- od użytkownika. Czynników niezależnych, zwanych zewnętrznymi, jest dość wiele. Zwykle pochodzą od środowiska pracy i trudno jest obliczyć ich wpływ na możliwości umysłowe człowieka-operatora systemu.

W tab. 1.1 przedstawione zostały dwa przykładowe scenariusze opisujące realizację tego samego zadania w edytorze tekstu. Zadanie polega na wstawieniu wytłuszczonego wyrazu (8. znakowego) w tekst napisany normalną czcionką.

Dla scenariusza numer 1 z tab. 1.1 można zbudować następujący model KLM działań użytkownika: **M+P+BB**+**H**+**M**+9x**K+H+M+P**+**BB**+**M**+**P**+**BB**. Zakładając brak opóźnień wnoszonych przez system i dodatkowych działań o charakterze umysłowym, co oznacza wysoki poziom umiejętności użytkownika, średni czas wykonania zadania wynosi:

1,2+1,1+0,2+0,4+1,2+9x0,28+0,4+1,2+1,1+0,2+1,2+1,1+0,2,

czyli 12,02 s.

Model KLM dla scenariusza numer 2 (tab. 1.1) jest następujący: **M**+**P**+**BB**+**M**+**H**+**K**+**M**+9x**K**. Daje to w konsekwencji następującą sekwencję czasów wykonania:

1,2+1,1+0,2+1,2+0,4+0,28+1,2+9x0,28.

Łączny czas wykonania zadania w scenariuszu 2 wynosi zatem 8,1 s.

Tab. 1.1. Opis scenariuszowy wykonania zadania wpisania wytłuszczonego słowa

Scenariusz 1	**Scenariusz 2**
1. Wskazanie miejsca wstawienia wyrazu 2. Kliknięcie w miejscu wstawienia 3. Przeniesienie ręki na klawiaturę 4. Wpisanie wyrazu 5. Przeniesienie ręki na myszkę 6. Wskazanie wpisanego wyrazu ze spacją kończąca 7. Podwójne kliknięcie na wyrazie by go zaznaczyć 8. Wskazanie ikony B na wstążce w *Narzędzie główne* 9. Kliknięcie na ikonie B	1. Wskazanie miejsca wstawienia wyrazu 2. Kliknięcie w miejscu wstawienia 3. Przeniesienie ręki na klawiaturę 4. Wybór skrótu*: Ctrl+B* 5. Wpisanie wyrazu ze spacją kończącą

Źródło: opracowanie własne

Model KLM, jak każdy model, ma liczne uproszczenia. Nie przewiduje on uwzględnienia konsekwencji błędów popełnianych przez użytkowników w trakcie wykonywania działań. Brak mu jest także właściwego mechanizmu uwzględnienia stopnia zaawansowania, a więc poziomu umiejętności, użytkowników w obsłudze systemu. Wszystkie czasy realizacji czynności związanych z ludźmi-operatorami (oprócz operatora **M,** a i dla niego brak jest stabilnych i uzasadnionych wytycznych) nie mają żadnego związku z poziomem umiejętności użytkownika realizującego scenariusz wykorzystania systemu informatycznego. Na czas realizacji operatora akcji umysłowej (**M**) wpływa szereg czynników, takich jak:

- poziom umiejętności i inteligencji (IQ) – obydwa te czynniki zwiększają szybkość podejmowania decyzji;
- intensywność bodźca (wyzwalacza) – im silniejszy bodziec tym szybsze wykonanie operatora **M**;
- niepewność – im więcej możliwości do wyboru, tym dłużej trwa wybór (zależność jest logarytmiczna – prawo Hicka).

Model KLM może służyć do porównywania czasów realizacji zadań przy pomocy różnych scenariuszy oraz wskazaniu, kiedy jeden scenariusz jest lepszy od drugiego. W tym celu scenariusze poddaje się parametryzacji (w przykładzie z wpisywaniem słowa parametrem jest jego długość) i porównuje czasy realizacji zadań, wyrażone funkcjami od wartości parametru.

W tab. 1.2 przedstawiono dwa scenariusze realizacji zadania polegającego na zmianie wielkości rysunku w tekście. W scenariuszu A zmiana ta jest realizowana przy pomocy wpisania konkretnej wartości skali odwzorowania w formularzu (rys. 1.19) w polu *Wysokość*, a w scenariuszu B do tego celu wykorzystywane są przyciski inkrementacji/dekrementacji tego pola. Założono, że skala jest wartością dwuznakową. Parametrem jest liczba kliknięć przycisku inkrementacji/dekrementacji – ***n***.

Tab. 1.2. Opis scenariuszowy wykonania zadania zmiany wielkości rysunku

Scenariusz A	**Scenariusz B**
1. Wskazanie rysunku 2. Kliknięcie na rysunku prawym przyciskiem myszy 3. Wskazanie opcji *Formatuj obraz…* 4. Kliknięcie na opcji 5. Wskazanie zakładki *Rozmiar* 6. Kliknięcie na zakładce 7. Wskazanie pola *Wysokość* 8. Podwójne kliknięcie na polu 9. Przeniesienie ręki na klawiaturę 10. Wpisanie dwóch znaków 11. Przeniesienie ręki na mysz 12. Wskazanie przycisku *OK* 13. Kliknięcie przycisku *OK*	1. Wskazanie rysunku 2. Kliknięcie na rysunku prawym przyciskiem myszy 3. Wskazanie opcji *Formatuj obraz…* 4. Kliknięcie na opcji 5. Wskazanie zakładki *Rozmiar* 6. Kliknięcie na zakładce 7. Wskazanie pola *Wysokość* 8. ***n*** kliknięć na przyciskach inkrementacji/dekrementacji 9. Wskazanie przycisku *OK* 10. Kliknięcie na przycisku *OK*

Źródło: opracowanie własne

Model KLM dla scenariusza A wyraża się następującą sekwencją operatorów interakcji człowiek-komputer:

P+**M**+**BB**+**P**+**BB**+**P**+**BB**+**M**+**P**+**BB**+**BB**+**H**+**M**+**K**+**K**+**H**+**M**+**P**+**BB**

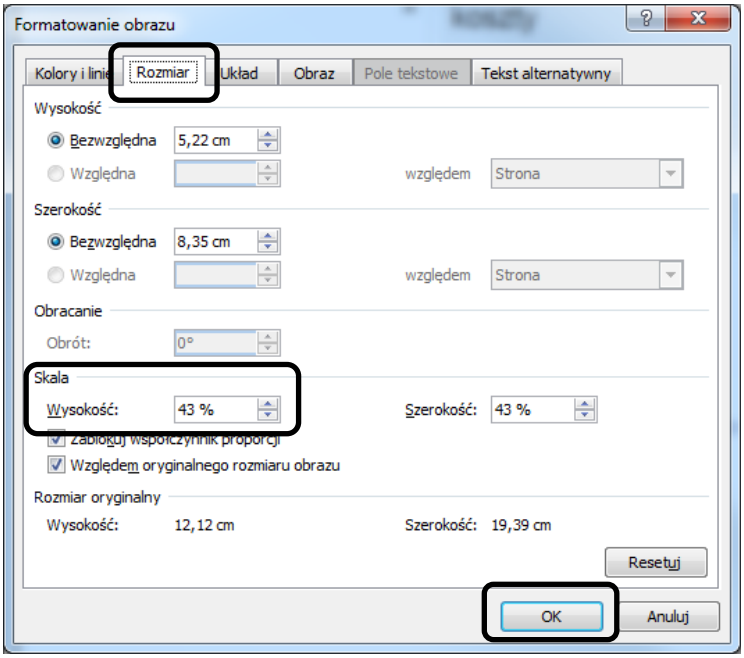
Sekwencja ta dla scenariusza B jest następująca:

P+**M**+**BB**+**P**+**BB**+**P**+**BB**+**M**+**P**+*n*x**BB**+**M**+**P**+**BB**

W konsekwencji czasy realizacji poszczególnych scenariuszy wynoszą (w sekundach):

Scenariusz A: 12,86

Scenariusz B: 9,9+0,2***n***

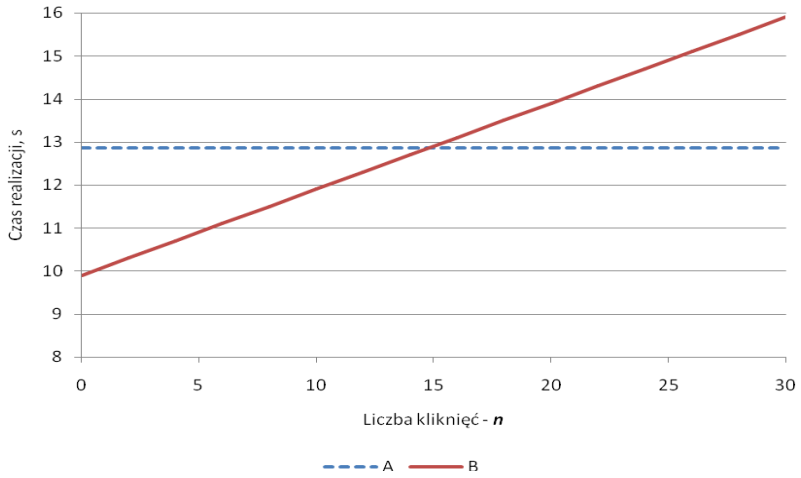


Rys. 1.19. Okno – formularz Formatowanie obrazu z zaznaczonymi obszarami działań
Źródło: opracowanie własne

Wykres na rys. 1.20 przedstawia zależność czasu wykonania poszczególnych scenariuszy od parametru ***n***. Na jego podstawie można wywnioskować, że scenariusz B jest bardziej efektywny (szybszy) od scenariusza A w przypadkach zmiany skali odwzorowania o maksimum ±15 procent (wartości inkre-/dekrementacji).

Metoda KLM ta ma parę istotnych niedostatków, a mianowicie:

- tylko eksperci wykonują rutynowo działania (w przypadku początkujących użytkowników, średnie czasy nie mają zastosowania);
- mierzy tylko efektywność wykonawczą, a nie pozostałe cechy jakości interfejsu;
- nie bierze pod uwagę całego szeregu czynników, takich jak: pojawianie się błędów, akcji równoległych (np. połączenie wciśnięcia klawiszów klawiatury i myszki jednocześnie), czasu potrzebnego na zaplanowanie działań.



Rys. 1.20. Zależność czasu wykonania scenariuszy A i B od liczby kliknięć zmieniających wartość skali odwzorowania
Źródło: opracowanie własne

Prawo Hicka

Prawo Hicka (ang. *Hick's Law*) przedstawia wpływ niepewności na szybkość podejmowania decyzji. Przedstawia ono czas T wyboru opcji z n równoprawdopodobnych możliwości jako:

$$T = b \cdot \log_2(n+1), \tag{1.4}$$

przy czym: b jest stałą (zależną między innymi od umiejętności i poziomu inteligencji konkretnego człowieka-decydenta), a dodanie 1 do n wynika z istnienia jeszcze jednej (poza n) możliwości decyzyjnej – wycofania się. Logarytm o podstawie dwa wskazuje na binarne przeszukiwanie możliwości. Prawo Hicka nie działa przy przeszukiwaniu nieuporządkowanego zbioru możliwości pod kątem znalezienia właściwego elementu – w takim zbiorze czas podejmowania decyzji zależy liniowo od n.

1.4. Wysiłek poznawczy i zapamiętywalność

W trakcie procesu poznawczego, człowiek wytwarza lub modyfikuje reprezentacje umysłowe wiedzy w swoim systemie poznawczym, czyli umyśle. Sam proces poznawczy polega na przetwarzaniu informacji ze wszystkimi funkcjami tego procesu, a mianowicie: pozyskiwania informacji (bodźców) z otoczenia, jej przekształceniu i przechowywaniu oraz wykorzystaniu, zwykle po pewnym czasie w reakcji na bodźce. Podejście poznawcze zakłada, że wiedza (w tym również i emocje) jest reprezentowana w umyśle w symbolicznej

postaci. Poszczególne etapy przetwarzania informacji w procesie poznawczym składają się z elementarnych, następujących procesów elementarnych:

- **uwaga**, czyli utrzymanie organizmu w gotowości do odbioru (i późniejszego przetwarzania) informacji z otoczenia (bodźców); utrzymanie uwagi to zdolność do zorganizowanego i ukierunkowanego działania;
- **percepcja**, czyli uświadomione odbieranie bodźców zewnętrznych – postrzeganie otoczenia przy pomocy zmysłów człowieka; na percepcję składają się procesy odbioru wrażeń (wzrokowych, dźwiękowych, smakowych, zapachowych oraz dotykowych, a także zmiany temperatury) – odbiór sensoryczny, ich identyfikacji i klasyfikacji oraz przygotowania do reakcji na bodziec;
- **pamięć**, czyli zdolność do zarejestrowania i ponownego przywoływania informacji; pamięć ludzka ma całkowicie inny charakter, niżeli pamięć komputerowa (czyli przechowywanie danych w adresowalnych komórkach – niezależnie od implementacji sprzętowej – w chwili obecnej układy scalone, półprzewodnikowe na bazie krzemu).
- **funkcje wykonawcze** (zarządcze), czyli przetwarzanie pozyskanych bodźców i informacji pozyskanej z pamięci, w celu sterowania procesem poznawczym.

Pamięć jest bardzo ważnym elementem procesu poznawczego. U człowieka wyróżniono wiele typów pamięci o różnym charakterze i sposobie działania. Do najważniejszych z nich należą pamięci:

- deklaratywna,
- sensoryczna,
- krótkotrwała,
- długotrwała,
- niejawna (utajniona),
- proceduralna.

Pamięć deklaratywna przechowuje informacje w postaci łatwej do wydobycia (dlatego zwana jest także pamięcią jawną, świadomą), zwykle w postaci informacji o rzeczach (ang. *Things*) i związkach pomiędzy nimi. Są one reprezentowane w pamięci w postaci reprezentacji językowych (abstrakcyjnych lub konkretnych). Zarówno zapamiętywanie jak i wydobycie informacji z pamięci deklaratywnej wymaga wysiłku i czasu. Nie zależy ono jednak od kontekstu.

Pamięć sensoryczna umożliwia rejestrację bodźców i charakteryzuje się dużą szybkością zapamiętywania, ale niewielką pojemnością informacyjną bodźców (typu: gorąco/zimno) i bardzo krótkim czasem przechowywania (do 0,5 s). Pamięć ta, w zależności od typu bodźca, dzieli się na pamięć ikoniczną (bodźce wzrokowe) oraz echoiczną (bodźce słuchowe). Informacje z tej pamięci przekazywane są do pamięci krótkotrwałej.

Pamięć krótkotrwała ze względu na swój charakter zwana jest pamięcią natychmiastową, roboczą lub operacyjną (poprzez analogię do pamięci systemów komputerowych o analogicznym charakterze). Zapamiętywanie w niej jest szybkie, automatyczne i nie wymaga wysiłku. Pamięć krótkotrwała charakteryzuje się niezbyt długim czasem przechowywania informacji (od kilkunastu sekund do kilkunastu minut – przy powtórkach, czyli „odświeżaniu") i niewielką pojemnością. Pamięć ta przechowuje informacje z pamięci sensorycznej oraz te pobrane z pamięci długotrwałej, bądź też będące wynikiem przetwarzania informacji w mózgu. Pamięć krótkotrwała podlega zasadzie Millera. Zasada ta ustala pojemność pamięci krótkotrwałej na 7±2 elementów zapamiętywanych. Ma ona istotny wpływ na konstrukcję interfejsu człowiek-komputer.

Pamięć długotrwała to złożony zbiór procesów kodowania i zapisywania informacji, jej przechowywania oraz wydobywania (przywoływania z pamięci, odtwarzania) informacji z pamięci. Nie jest znana pojemność tej pamięci i czas przechowywania. Podlega ona bardzo skomplikowanym procesom zapominania i ponownego wznawiania „dostępu" po długim okresie jego braku.

Pamięć utajniona, jak sama nazwa sugeruje, jest pamięcią przechowującą informacje niepoddające sie procesowi jawnego przedstawienia (werbalizacji) czy też uświadomienia przez człowieka. Pamięć ta gromadzi informacje na temat nawyków (np. jazda na rowerze), a informacje są z niej wydobywane w sposób automatyczny (na poziomie nawyków w modelu Rasmussena – rys. 1.4). Gromadzenie informacji w tej pamięci wymaga wielokrotnego powtarzania czynności i takiego opanowania umiejętności, by działanie człowieka przeszło, z poziomu bazującego na wiedzy czy regułach (model Rasmussena), na poziom nawyków.

Pamięć proceduralna jest rodzajem pamięci długotrwałej. W niej zapamiętywane są działania związane z czynnościami praktycznymi i związanymi z nią procedurami motorycznymi, koordynacyjnymi oraz czynnościowymi. Dostęp do niej jest automatyczny – z pominięciem świadomości człowieka.

Jednym z istotnych uwarunkowań budowy interfejsów jest proces poznawczy (oraz związany z nim **wysiłek poznawczy**) i jego ograniczenia, w tym problem zapamiętywania jako rezultatu utrwalania percepcji rzeczywistości.

Do najczęstszych uwarunkowań związanych z procesem zapamiętywania, zaliczana jest wspomniana już liczba Millera. Badania wykazały bowiem, że pojemność pamięci krótkotrwałej, czyli tej używanej w codziennej pracy jest ograniczona do 7±2 elementów. Rzutuje to na podejście do struktury i zawartości informacyjnej ekranów aplikacji. Przyjęło się, by regule tej podlegała liczba:

- opcji w menu,
- ikon w pasku narzędziowym,

- opcji w menu *drop-down*,
- elementów danych w oknie/zakładce

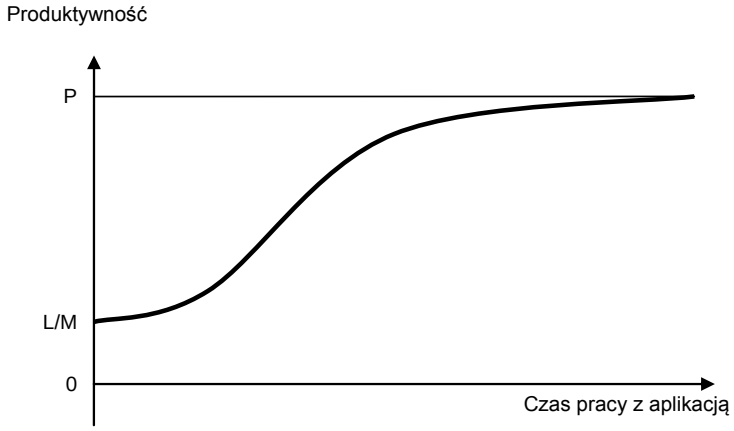
do równoczesnego postrzegania, zrozumienia i wyboru.

Nie jest to jednak do końca właściwe podejście, bowiem człowiek, przeglądając listę opcji, zapomina o tych już rozpatrzonych i odrzuconych jako niepotrzebne. Prawo to stosuje sie jednak do konstrukcji okien/zakładek często stosowanych (tam gdzie szybkość znalezienia właściwego elementu ma istotne znaczenie dla obsługi aplikacji), ale w przypadku okien rzadko używanych (np. konfiguracyjnych) nie jest już takie istotne. Odpowiednio zaprojektowane okno ogranicza liczbę bodźców kierowanych do użytkownika i koniecznych do postrzegania do 5-9.

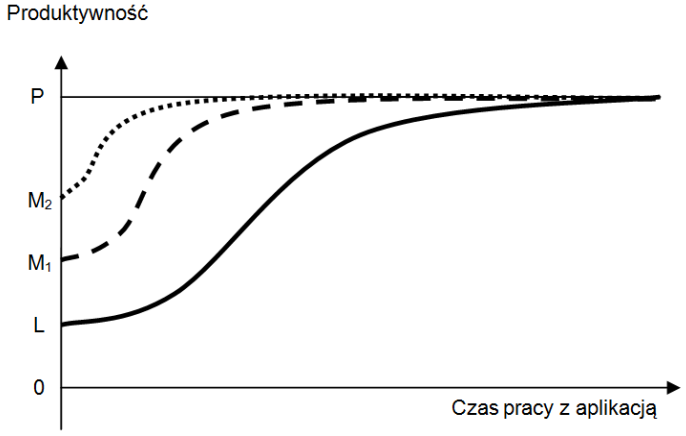
Konieczność zapamiętywania przez człowieka wielu informacji pod rząd w krótkim czasie, powoduje przeciążenie pamięci krótkotrwałej. Skutkuje to brakiem możliwości właściwego zapamiętywania bodźców. Wskazane jest zatem wykorzystywanie w interfejsie możliwości wskazywania (wyboru z możliwości z zapominaniem), a nie wpisywania (z pamiętaniem). W interfejsach zatem promowane powinno być rozpoznawanie obiektów, a nie konieczność ich przywoływania z pamięci. Proces rozpoznawania może być wspomagany poprzez grupowanie opcji, użycie kolorów, stosowanie typowych elementów czy też stosowanie metafor. Metafory wykorzystują codzienne pojęcia i przenoszą je do obszaru interfejsu sztucznego sytemu. Pozwalają na wykorzystanie wiedzy i pamięci długotrwałej w pracy z systemami informatycznymi bez angażowania łatwo przeciążalnej pamięci krótkotrwałej.

Zapamiętywalność, czyli zdolność do zapamiętywania (ang. *Memorability*), to zdolność użytkownika powrotu do biegłości w posługiwaniu się aplikacją po przerwie (dłuższej lub krótszej) w jej wykorzystywaniu. Właściwość ta jest szczególnie ważna dla aplikacji, którymi użytkownicy posługują się okazjonalnie (np. aplikacji internetowych). Jest ona związana z pojęciem produktywności użytkownika w czasie pracy z aplikacją. Produktywność w czasie narasta w miarę zdobywania doświadczeń i wiedzy przez użytkownika – czyli wraz z czasem pracy z aplikacją. Przykładowy przebieg tych zmian przedstawia rys. 1.21.

Na rys. 1.21 wartość P oznacza maksymalną produktywność użytkownika w pracy z danym systemem czy aplikacją. Maksymalna produktywność (w większości przypadków) wynika z fizycznych uwarunkowań i ograniczeń człowieka-operatora systemu. Wartość L/M oznacza początkową produktywność, tj. taką, którą użytkownik osiąga przy pierwszym kontakcie z aplikacją. Wynika ona głównie z jakości interfejsu aplikacji. Im wyższa jakość interfejsu (np. intuicyjność, łatwość rozumienia czy manipulowania) tym L/M ma większą wartość (przy czym możliwa jest także jego wartość ujemna).



Rys. 1.21. Zależność produktywności użytkownika od czasu pracy z aplikacją
Źródło: opracowanie własne

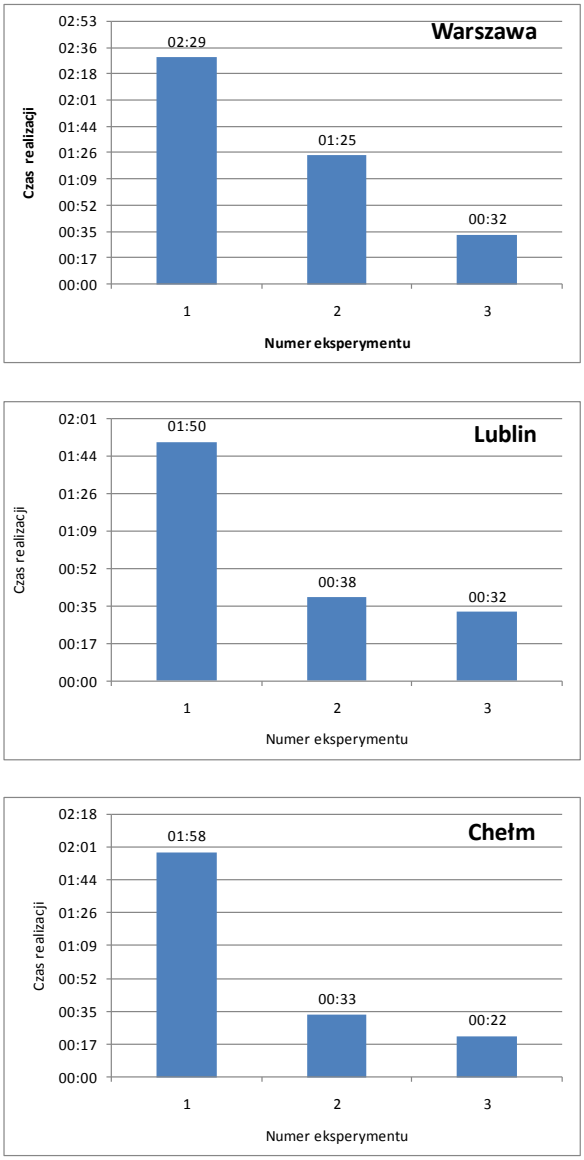


Rys. 1.22. Teoretyczna krzywa pierwotnego i ponownego uczenia się
Źródło: opracowanie własne

Zapamiętywalność to zjawisko zwiększenia początkowej efektywności pracy użytkowników, którzy wykorzystują daną aplikację, przy kolejnych sesjach pracy z nią – rys. 1.22.

Punkty L, M_1 oraz M_2 na rys 1.22 oznaczają odpowiednie produktywności początkowe pracy użytkownika przy pierwszych i ponownych kontaktach użytkownika z aplikacją. Im większe przyrosty wartości M, tym większa jest zapamiętywalność.

Badania zapamiętywalności aplikacji webowych przedstawia m.in. artykuł [11]. Przedstawiono w nim rezultaty badań czasów realizacji tych samych zadań na różnych stronach internetowych, które powinny zawierać takie same informacje wynikające z przepisów prawa (strony BIP urzędów miejskich).



Rys. 1.23. Czas realizacji tych samych zadań na różnych stronach internetowych, przy ponownych odwiedzinach strony przez tego samego użytkownika
Źródło: badania własne – [11]

Uczestnicy badania poszukiwali tych samych informacji w wielu cyklach oddalonych od siebie jednogodzinną przerwą. W trakcie tej przerwy uczestnicy badania wykonywali inne, niezwiązane z danym zadania. Szybkość pracy z takimi aplikacjami w większości przypadków zależy od jakości interfejsu. Czas realizacji zadań w tym eksperymencie [11], obserwowany dla różnych serwisów internetowych (urzędów miejskich Warszawy, Lublina i Chełma) przedstawia rys. 1.23. Czas ten jest odwrotnością produktywności użytkowników serwisu i wyznacza zapamiętywalność serwisów.

1.5. Podsumowanie

Ergonomia systemów ma na celu doprowadzenie do wzajemnej kompatybilności maszyny i człowieka, a także warunków zewnętrznych ich funkcjonowania. Priorytetem jest człowiek. Ergonomia korekcyjna wyszukuje i eliminuje istniejące usterki w istniejących układach człowiek-maszyna, a ergonomia koncepcyjna wykorzystywana jest w trakcie projektowania nowych systemów.

Modele zachowań człowieka bada kognitywistyka. Dzięki tej nauce powstały takie modele jak: hierarchii potrzeb Maslowa, ukierunkowanych działań Normana, Rasmussena, przetwarzania informacji MHP oraz model ICS. Wszystkie one mają za zadanie wyjaśnić działanie człowieka na różnych poziomach abstrakcji i w różnych sytuacjach.

Współpraca człowieka z interfejsem oprogramowania jest uwarunkowana jego biologicznymi właściwościami i współdziałaniem wielu jego układów, wyjaśnianym w modelach zachowań. Bezpośrednio do interfejsu odnosi sie prawo Fittsa, pozwalające wyznaczyć czas wykonania przez człowieka zadania pozycjonowania fokusu. Metoda KLM, dzieląca działania człowieka na szereg prostych operacji, umożliwia natomiast wyznaczenie czasu realizacji zadań przy pomocy klawiatury i urządzeń wskazujących. Umożliwia ona porównanie czasu realizacji różnych scenariuszy działań. Kolejne prawo – Hicka – przedstawia wpływ niepewności na szybkość podejmowania decyzji przez człowieka.

Wysiłek poznawczy wynika z konieczności wytwarzania lub modyfikacji, a także zapamiętywania i przywoływania z pamięci, informacji w trakcie procesu poznawczego i pracy. Różne pamięci człowieka mają różne charakterystyki czasowo-pojemnościowe, które powinny być uwzględniane przy konstrukcji interfejsu oprogramowania.

Zapamiętywalność jest jedną z właściwości interfejsu. Wpływa ona na efektywność pracy człowieka. Ma szczególne znaczenia w przypadku działań realizowanych sporadycznie w dużych odstępach czasu.

2. Interfejs oprogramowania i jego ergonomia

Odpowiednio zaprojektowany interfejs użytkownika powinien dostarczyć takiego zestawu mechanizmów wejściowych i wyjściowych, który w możliwie najbardziej efektywny sposób spełni oczekiwania użytkownika i uwzględni jego możliwości oraz ograniczenia. Cechą dobrze zbudowanego interfejsu jest to, że pozostaje on transparentny dla użytkownika – ułatwia mu skupienie się na szukanych informacjach oraz wykonywanych zadaniach, zamiast na użytych mechanizmach i sposobach prezentacji danych. Interfejs powinien wspomagać percepcję informacji przez użytkownika, a więc być dla niego czytelny i zrozumiały.

Interfejs wskazuje użytkownikowi funkcje dostępne w systemie. Interfejs, który nie jest efektywny oraz intuicyjny, utrudnia pracę i jest przyczyną powstawania wielu błędów i nieporozumień, a w konsekwencji prowadzi do niezadowolenia oraz irytacji użytkowników. Niekiedy poprawność interfejsu decyduje o bezpieczeństwie, a niepoprawny interfejs może doprowadzać do powstawania realnych zagrożeń.

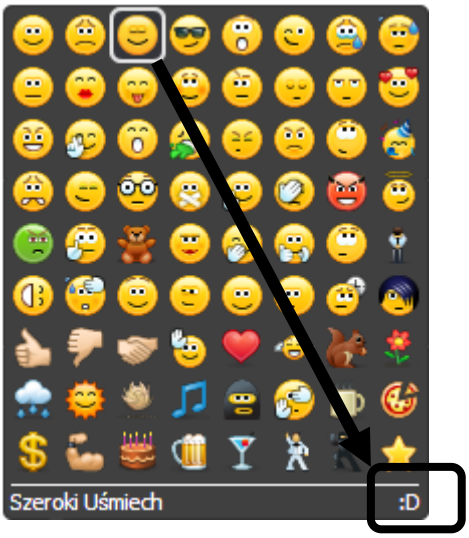
2.1. Typy i obiekty interfejsu oprogramowania

Interfejs oprogramowania jest narzędziem interakcji człowieka z komputerem. Tworzą je odpowiednio oprogramowane urządzenia wejścia-wyjścia. Dostarczają one bodźców człowiekowi i na odwrót – odbierają jego polecenia. Urządzenia te pośredniczą pomiędzy człowiekiem, a komputerem. Człowiek bez tych urządzeń nie jest w stanie bezpośrednio komunikować się z maszyną.

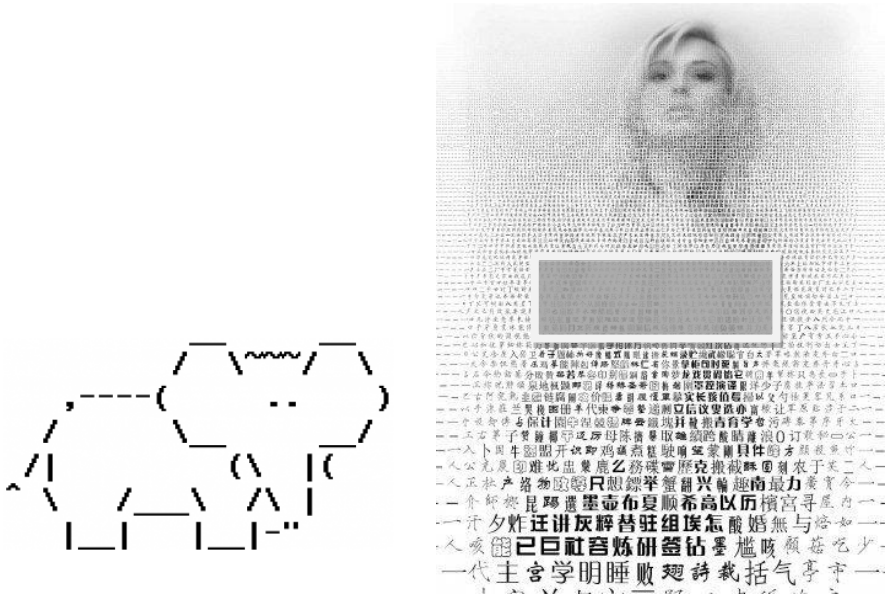
Tryby pracy interfejsów oprogramowania (głównie dotyczy to monitora jako urządzenia wyjściowego) można podzielić na dwie podstawowe grupy: tryb tekstowy i graficzny.

Tryb tekstowy (ang. *Text-based Interface* lub *Text User Interface*, *TUI*) do prezentacji informacji wykorzystuje znaki wyświetlane na monitorze w postaci wierszowej (czyli teksty) i ewentualnie tzw. semigrafikę (tj. układy wielu znaków sugerujących inne niż teksty obiekty ekranowe). Tryb tekstowy może wykorzystywać klawiaturę, ale też i inne urządzenia wejściowe, takie jak myszka, pióro świetlne lub ekran dotykowy.

Semigrafika używana jest do dnia dzisiejszego w e-mailach lub innych formach komunikacji tekstowej (np. SMS czy Skype) w postaci **emotikon**, czyli układów znaków tekstowych (zwykle pochodzącego z podstawowego zbioru ASCII) oznaczających wyraz nastroju jako grupa znaków typu: :) :-) :-) :D :* itd. Emotikony obecnie mają też swoje odpowiedniki graficzne – rys. 2.1. Bardziej złożona semigrafika przedstawiać może skomplikowane obrazy graficzne – rys. 2.2. Tworzenie takich obrazów uzyskało nawet swoją nazwę jako kierunek sztuki: ASCII Art [18].



Rys. 2.1. Graficzny wygląd i kod tekstowy emotikonów w programie Skype
Źródło: opracowanie własne



Rys. 2.2. Przykłady prostej i złożonej (artystycznej –> ASCII Art) semigrafiki
Źródło: http://www.cwd.co.uk/asciimoo/view/11 i http://live4fun.ru/joke/180877

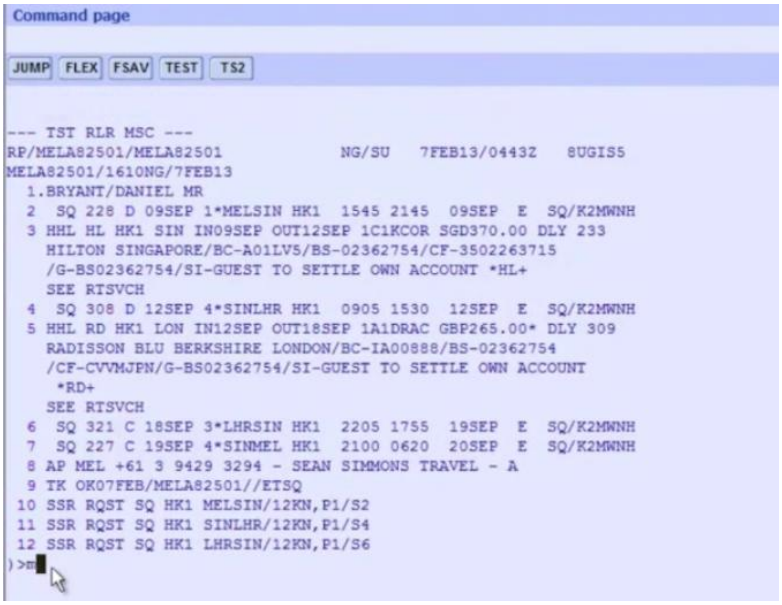
Tryb graficzny (ang. *Graphical User Interface, GUI*) prezentuje informacje (i udostępnia obiekty do ich wprowadzania) w postaci graficznych obiektów ekranowych. Użytkownik może wprowadzać dane do programu poprzez oddziaływanie na obiekty ekranowe przy pomocy urządzeń wskazujących (myszka, pióro świetlne, tablet czy ekran dotykowy). Może, ale nie musi, zawierać takie elementy graficzne jak okna czy menu rozwijane (ang. *Drop-down Menu*).

W zależności od sposobu realizacji sposobu oddziaływania człowieka na komputer, rozróżnia się następujące typy interfejsów:

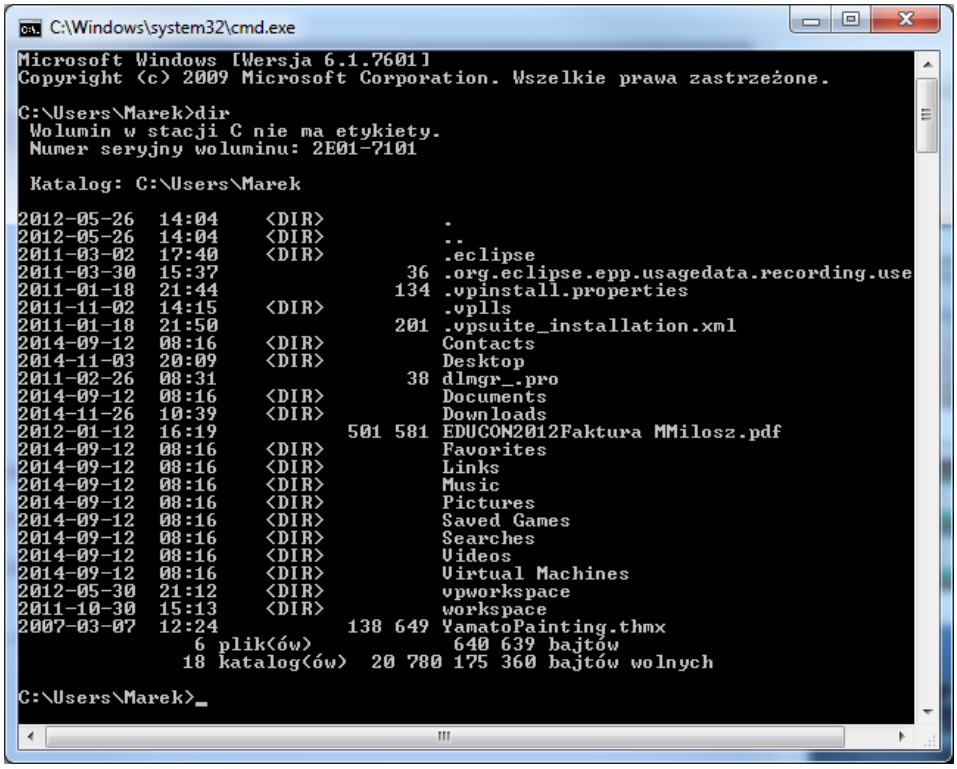
- **wiersz poleceń** (ang. *Command Line Inteface, CLI*), zwany niekiedy bazującym na poleceniach (ang. *Command Based Interface, CBI*), polega na wydawaniu komputerowi poleceń (zwykle przy pomocy klawiatury) i odbiorze rezultatów na monitorze; działa zwykle w trybie znakowym, chociaż możliwe jest jego użycie w GUI);
- **WIMP** (ang. *Windows – Icons – Menus – Pointer)* jest podstawowym GUI, którego koncepcja powstała w latach 70. XX wieku (w laboratorium firmy Xerox) i została rozwinięta do zaawansowanego interfejsu graficznego stosowanego obecnie;
- **organiczny** interfejs użytkownika (ang. *Organic User Interface, OUI*) wykorzystuje elastyczne ekrany dotykowe do organizacji nowatorskiego sposobu wprowadzania danych do komputera-smartfona;
- interfejs **głosowy** (ang. *Voice User Interface, VUI*) wykorzystuje układ pozyskania (mikrofon) i rozpoznawania dźwięków do sterowania oprogramowaniem; stosowany jest tam, gdzie człowiek nie może lub nie powinien (bo np. jest to niezgodne z przepisami prawa) użyć rąk do obsługi systemów – przykładem jest system sterowania telefonem głośnomówiącym w samochodzie;
- interfejs **dotykowy** (ang. *Touch User Interface, TUI*) wykorzystuje specjalne tabliczki lub całe monitory do sterowania; coraz częściej obecnie jest to tzw. multi-touch, czy system rozpoznający dwa i więcej punkty dotyku i reagujący na różne dynamiczne zmiany tych punktów (np. przeciągnięcie, dłuższe przytrzymanie lub obrót);
- interfejs **gestowy** (ang. *Gesture-Based Interface, GBI*) wykorzystuje kamery do śledzenia ruchów rąk, głowy czy oczu a odczytując gesty wykorzystuje je jako polecenia;
- interfejs **multimodalny** (ang. *Multimodal User Interface, MUI*) wykorzystuje równocześnie wiele różnych bodźców w oddziaływaniu na człowieka i *vice versa*; najprostszym przykładem jest wibrujący telefon, wydający równocześnie sygnał dźwiękowy i świetlny;
- **rozszerzona rzeczywistość** (ang. *Augmented Reality, AR*) wykorzystuje obraz świata realnego i nakłada na niego wygenerowane sztucznie obrazy.

Interfejs CLI posiadały systemy operacyjne komputerów w początkowym okresie ich rozwoju. Jest to naturalny sposób komunikacji człowiek-wykonawca, polegający na przekazywaniu polecenia przez człowieka (użytkownika) i otrzymaniu rezultatu jego działań. Jest on też stosowany do sterowania programami, szczególnie tymi bardziej złożonymi, jak na przykład systemami operacyjnymi, AutoCAD czy system rezerwacji miejsc w samolotach – jak np. systemem Amadeus (rys. 2.3). Zachował się on także we współczesnych systemach operacyjnych Linux, UNIX czy Windows (rys. 2.4). Charakteryzuje się użyciem klawiatury i ewentualnie jej klawiszy specjalnych (typu: F1 czy też PrtScr) do podawania szeregowo uporządkowanych w czasie komend. Komendy te są realizowane w trybie interpretacji poleceń, jedna po drugiej. Interfejs CLI jest precyzyjny (wykonywane są tylko prawidłowe, skodyfikowane polecenia z prawidłowymi parametrami, np. `copy a b`), efektywny i szybki. Jego podstawową wadą jest konieczność długotrwałego i pracochłonnego uczenia się składni poleceń przez użytkownika.

Interfejs CLI jest także używany w specyficznych sytuacjach. Wzbogacony o warstwę dźwiękową jest interfejsem, w którym polecenia są podawane dźwiękowo dla osób niewidzących.



Rys. 2.3. Interfejs CLI światowego systemu rezerwacji miejsc Amadeus
Źródło: https://www.youtube.com/watch?v=a0zktPs1bXw



Rys. 2.4. Linia poleceń w systemie Windows 7
Źródło: opracowanie własne

Interfejs WIMP składa się z następujących charakterystycznych obiektów ekranowych:

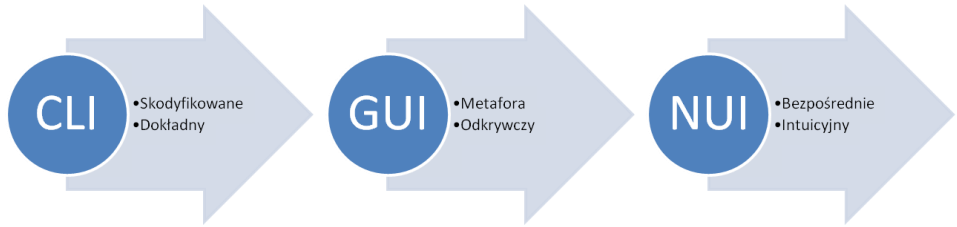
- **Okno** (ang. *Window*) – kontener zawierający elementy interfejsu związane z konkretnym (jednym) programem i izolujące ten program od innych uruchomionych w systemie (dla systemów wielozadaniowych, tj. takich, w których możliwe jest uruchomienie wielu programów w tym samym czasie).
- **Ikona** (ang. *Icon*) – obiekt graficzny umożliwiający wywołanie określonej akcji (np. uruchomienie programu).
- **Menu** (ang. *Menu*) – tekstowy lub ikonowy system, który umożliwia wybór opcji, programu lub też innej wykonywalnej akcji. Menu przybierają różne kształty: menu klasyczne w postaci listy, menu rozwijane, menu podręczne, pasek narzędziowy, wstążka itp.

- **Wskaźnik** (ang. *Pointer*) – symbol na ekranie wskazujący miejsce bieżącego oddziaływania urządzenia kontrolującego jego ruch (mysz, tabliczka dotykowa, klawiatura, joystick itd.). Zwykle w systemie są dwa wskaźniki, jeden związany z klawiaturą (tzw. wskaźnik znakowy – kursor) i drugi związany z urządzeniem manipulującym (tzw. kursor graficzny). Kursor znakowy wskazuje miejsce (lub obiekt ekranowy), do którego będzie przekazywana informacja – jest to miejsce skupienia – fokus (ang. *Focus*), oznaczając obiekt w specjalny sposób. Kursory mogą przybierać różny kształt sugerując różne akcje. W systemach sterowanych poprzez wskazanie na ekranie dotykowym kursor graficzny nie istnieje.

Interfejs WIMP stanowił istotny krok w rozwoju metod interakcji człowiek komputer. Zamienił on konieczność pamiętania (poleceń i składni) na wskazywanie i działanie odkrywcze. Użył więc innych mechanizmów procesów poznawczych i wykonawczych człowieka.

W chwili obecnej interfejs WIMP jest krytykowany ze względu na swoją ograniczoność i paradygmat funkcjonowania, który niewiele sie różni od przestarzałego technicznie i moralnie interfejsu CLI. W dalszym ciągu służy on do wydawania w trybie sekwencyjnym rozkazów w stylu: wybierz obiekt, wybierz akcję i ją wykonaj (np. kliknij na ikonie, wybierz z menu opcję i ją zatwierdź do wykonania).

Rozpatrywane są obecnie (istnieją już prototypy, a nawet działające, sprzedawalne i popularne systemy) interfejsy wykorzystujące rzeczywistość, czyli ciało człowieka i jego naturalne zachowanie (ang. *Natural User Interface, NUI*) [8]. Są one przedstawiane jako post-WIMP interfejsy. NUI zmienia paradygmat sterowania oprogramowaniem (rys. 2.5) ze „starego", wykorzystującego metafory i podejście odkrywcze, na nowy: bezpośredni i intuicyjny (czyli naturalny). W chwili obecnej powstają już programy używające tego interfejsu, które wykorzystują wielopunktowe ekrany dotykowe lub różnorodne urządzenia wizyjne – rys. 2.6.



Rys. 2.5. Ewolucja interfejsów użytkownika
Źródło: opracowanie własne na podstawie [15]

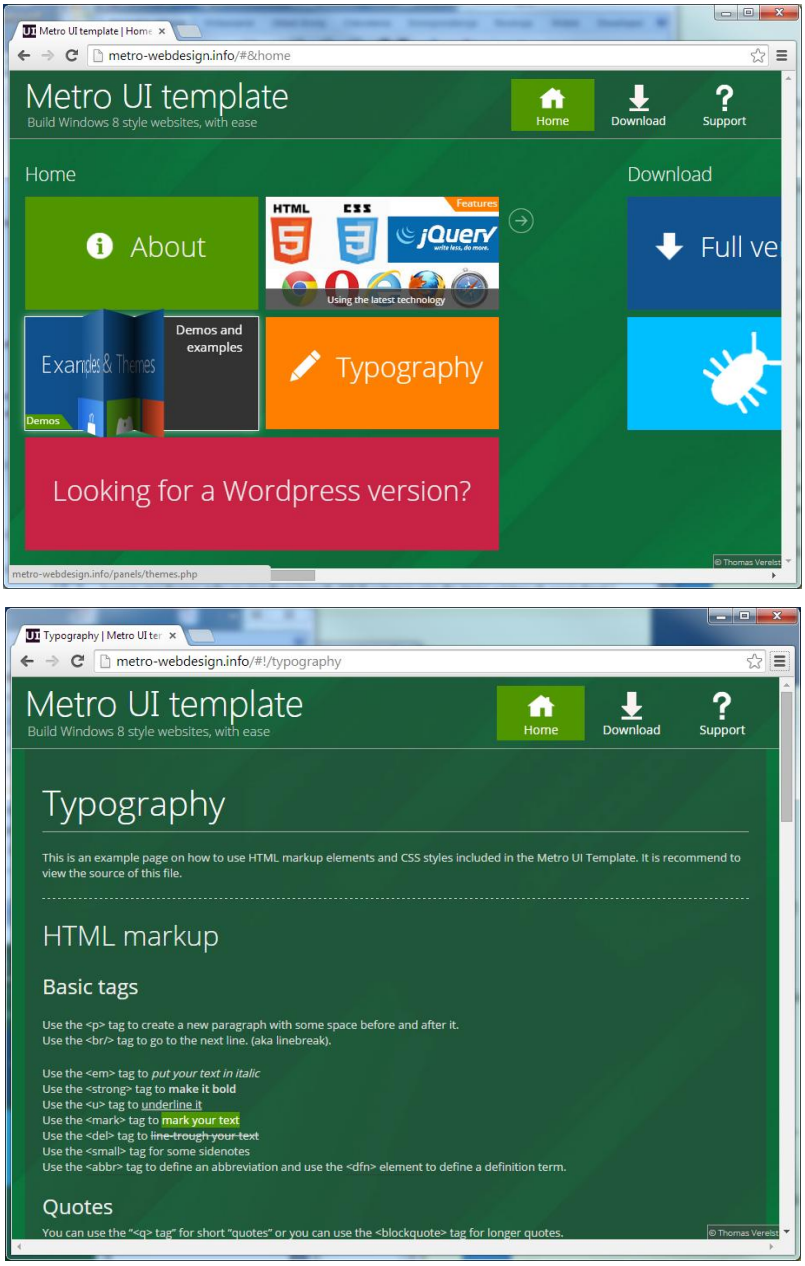
Rys. 2.6. NUI – przykłady
Źródło: http://en.wikipedia.org/wiki/File:Peterbilt_application_on_Microsoft_Surface.ogv i http://www.microsoft.com/en-us/pixelsense/pixelsense.aspx

Jednym z nowych pomysłów na interfejs NUI jest propagowany przez firmę Microsoft interfejs **MUI** (ang. *Modern User Interface*), znany do 2012 roku jako *Metro User Interface*. Interfejs ten jest połączeniem tradycyjnego pulpitu z minimalistycznym podejściem do napisów, ikon, kafli czy innych obiektów ekranowych. Stosuje duże, przejrzyste napisy, wykonane specjalnie opracowaną czcionkę o nazwie Segoe. Przykłady tego interfejsu, zastosowanego w Windows Phone 8 prezentują rys. 1.16, 2.9 i 2.11. Poza smartfonami interfejs ten zastosowano w Windows 8 i konsoli Xbox 360/One. Interfejs charakteryzuje się dużymi nazwami okien/pulpitów i przewijaniem nazw wewnątrz nich (rys. 1.16). Poza tym pulpity są często przewijane bocznie. W MUI stosuje się naturalne, „żywe" animacje. Interfejs MUI powoli przenika do aplikacji internetowych. Coraz więcej serwisów jest realizowanych zgodnie z jego zasadami – rys. 2.7.

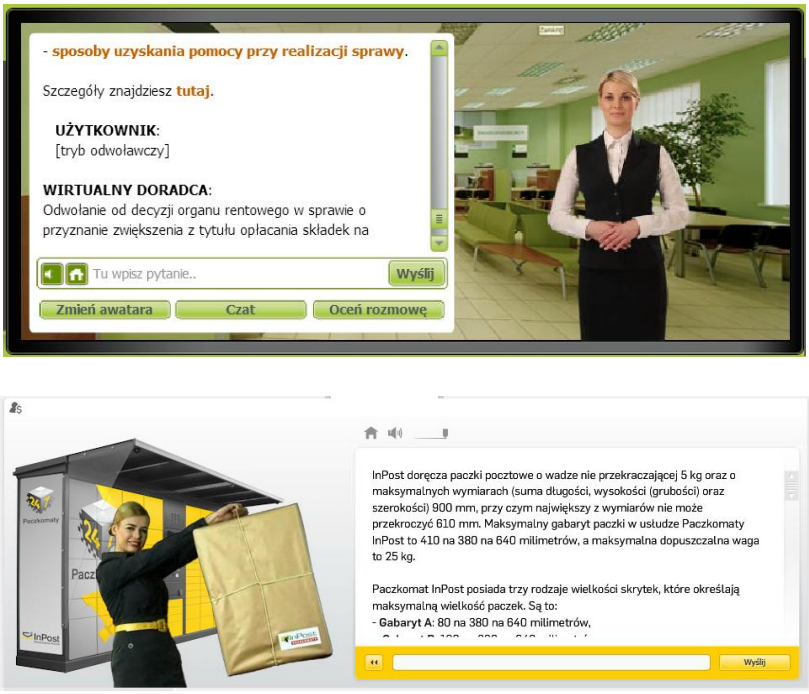
Organiczny interfejs użytkownika wykorzystuje wielokształtny i giętki wyświetlacz (lub jego analog w postaci projekcji holograficznej 3D), do którego dane wprowadzane są gestami lub dotykiem ręki człowieka poprzez deformację wyświetlacza. Należy do grupy interfejsów naturalnych (NUI). W interfejsach OUI miejsce wprowadzania danych i ich wyprowadzanie jest to samo, podczas gdy w tradycyjnych GUI używa się do tego różnych urządzeń: myszki i ekranu. Forma interfejsu, jego kształt i sposób wykorzystania powinny być dostosowane do funkcji.

Interfejs głosowy umożliwia człowiekowi komunikowanie się z komputerem tak jak z innym człowiekiem, czyli przy pomocy kanału dźwiękowego. Umożliwia to korzystanie z komputera w sytuacjach, kiedy człowiek nie może angażować do tego rąk (ang. *Hands-free*) i oczu (ang. *Eyes-free*). Poza urządzeniami technicznymi: mikrofonem i głośnikiem, stosuje sie w interfejsie głosowym oprogramowania rozpoznawania głosu i syntezatory mowy, a także zaawansowane oprogramowanie, bazujące na metodach sztucznej inteligencji, do interpretacji poleceń i wypracowywania odpowiedzi. Interfejs głosowy wykorzystywany jest także w sytuacjach „uczłowieczenia" komputera (antropomorfizm), np. w systemach doradczych dla ludzi – rys. 2. 8.

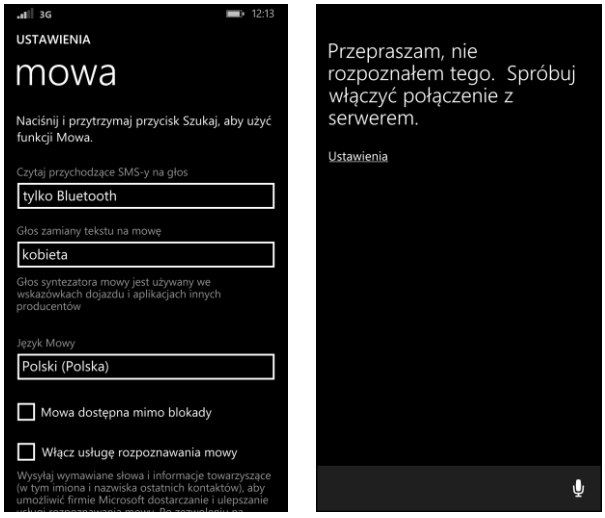
Interfejs głosowy ma wiele wad. Przede wszystkim jest mało odporny na zakłócenia. W sytuacjach dużego szumu lub niewyraźnie wydawanych poleceń systemy takie źle pracują. Przykładowe ustawienia interfejsu głosowego w telefonie i problemy w rozpoznawaniu mowy w urządzeniu mobilnym przedstawia rys. 2.9.



Rys. 2.7. Modern User Interface na stronie internetowej
Źródło: http://metro-webdesign.info

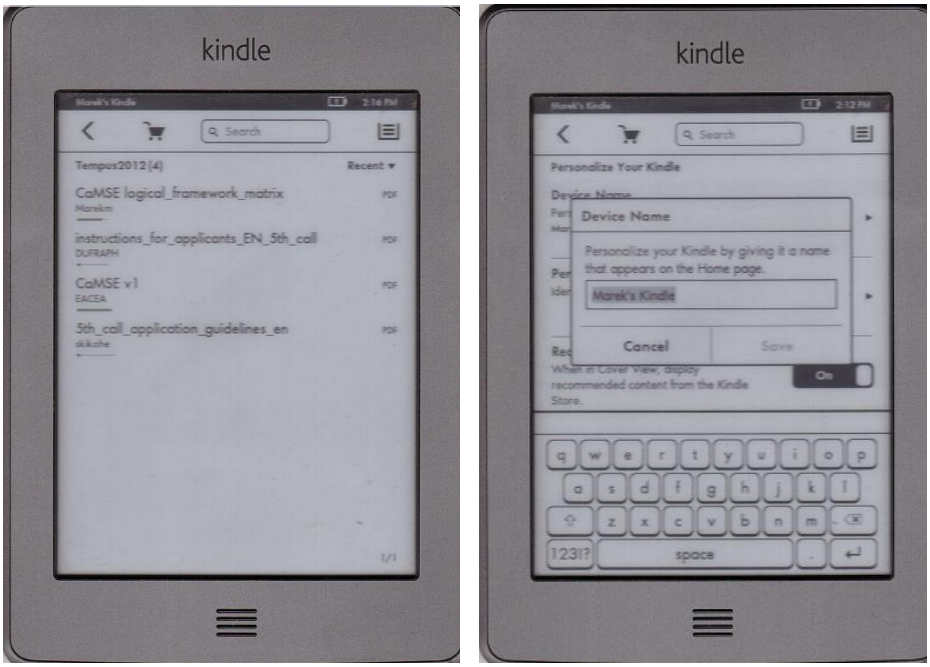


Rys. 2.8. Głosowo-tekstowy systemy doradcze wykorzystujący awatary
Źródło: http://www.zus.pl/, http://www.old.paczkomaty.pl/wirtualny_doradca,10.html

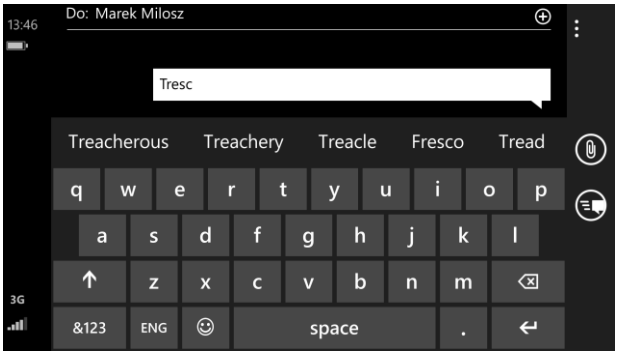


Rys. 2.9. Ustawie systemu rozpoznawania mowy i błędy jego wykonania – system Windows Phone 8
Źródło: opracowanie własne

Interfejs dotykowy wykorzystuje zmysł dotyku do przekazywania poleceń do komputera. W szczególnych przypadkach osób z upośledzeniem narządu wzroku, interfejs ten umożliwia dostarczanie informacji dotykowej także przy pomocy alfabetu Brailla. Interfejs ten zmniejsza znacznie barierę stosowania komputerów, poprzez zbliżenie ich obsługi do normalnego życia. Przewijanie stron w czytniku poprzez dotknięcie palcem jest bardzo naturalne i nie sprawia żadnych problemów – rys. 2.10. Pisanie na wirtualnej klawiaturze (rys. 2.11) poprzez dotykanie klawiszy jest zwykle dla normalnego użytkownika znacznie szybsze, niż korzystanie z dużej klawiatury sprzętowej. Poza ekranem dotykowym, na którym i tak wyświetlane są rezultaty interakcji, nie potrzeba żadnych innych dodatkowych urządzeń technicznych. Jest to olbrzymia zaleta interfejsu dotykowego ujawniająca się szczególnie w urządzeniach mobilnych i małogabarytowych.



Rys. 2.10. Czytnik Kindle z ekranem dotykowym
Źródło: opracowanie własne



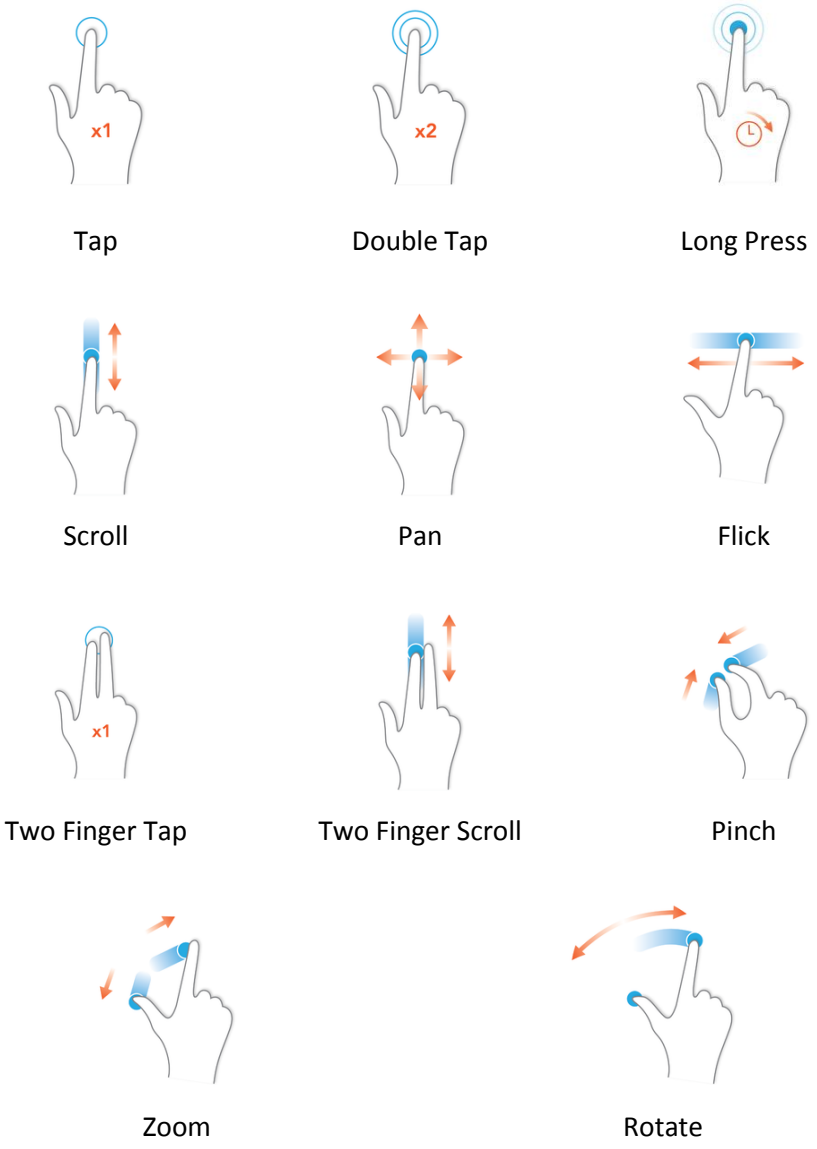
Rys. 2.11. Wirtualna klawiatura na ekranie dotykowym
Źródło: opracowanie własne

Wprowadzenie urządzeń rozpoznających więcej niż jeden wskazywany palcem punkt (ang. *Multi-touch*) pozwoliło rozbudować liczbę gestów wykonywanych palcami na powierzchni i rozpoznawanych przez interfejs. Listę gestów i ich znaczenie przedstawia rys. 2.12. Znaczenie gestów jest oczywiste i bardzo naturalne – od przewijania poprzez przemieszczanie i rozciąganie oraz obracanie. Gesty pozwalają także wykorzystać nowe formy interfejsu, jak np. przestrzenny układ obiektów ekranowych w interfejsie Androida – rys. 2.13.

Interfejsy dotykowe pojawiają się także w dedykowanych rozwiązaniach. Są one stosowane wszędzie tam, gdzie umieszczenie klawiatury czy też urządzenia wskazującego typu myszka nie jest uzasadnione lub wręcz niemożliwe. Znacznie bardziej ergonomicznym jest ekran dotykowy w systemie nawigacji satelitarnej w samochodzie lub też w kiosku informacyjnym w dużej galerii handlowej. Sprawdza się interfejs dotykowy wszędzie tam, gdzie nie ma miejsca na klawiaturę lub inne dodatkowe urządzenie lub też w trudnych warunkach otoczenia. Przykładem jest samoobsługowy bar piwny – rys. 2.14.

Interfejs dotykowy ma dużą zaletę: ekran służy zarówno do prezentacji informacji jak i jej wprowadzania. Cecha ta jest szczególnie ważna w urządzeniach mobilnych.

Kolejną zaletą interfejsu dotykowe, zwykle niedocenianą i nieoczywistą, jest jego programowa realizacja. O ile trudno zmienić układ przycisków na klawiaturze, a nawet ich opis (np. alfabet), to w interfejsie dotykowym nie ma z tym najmniejszego problemu.



Rys. 2.12. Gesty wykonywane palcami na ekranie dotykowym
Źródło: http://en.wikipedia.org/wiki/Multi-touch#Multi-touch_gestures

Rys. 2.13. „Przestrzenny” interfejs Androida
Źródło: opracowanie własne

Rys. 2.14. Interfejs dotykowy zintegrowanego systemu informatycznego w barze piwnym
Źródło: opracowanie własne, Berlin, Niemcy

Interfejs gestowy polega na tym, że praktycznie cała postać (ciało) człowieka jest źródłem poleceń dla komputera. Człowiek (lub jego część) jest filmowany przy pomocy kamery lub częściej wielu kamer, a jego gesty są interpretowane przez oprogramowania. Urządzeniem wyjściowym w takim interfejsie jest zwykle monitor lub projektor. Niekiedy (w specyficznych zastosowaniach) może to być głośnik. W interfejsach gestowych wykorzystuje się następujące urządzenia pozyskujące ruch ciała ludzkiego:

- rękawice – podają do komputera pozycję i ruchy rąk; mogą tez mierzyć kąty zginania palców a także powodować nacisk na dłoń, doprowadzając do człowieka bodźce;
- pojedyncze kamery 2D – umożliwiają filmowanie gestów rąk i mimiki twarzy (np. w laptopach, tabletach i smartfonach) w celu identyfikacji użytkownika lub też przyjmowania jego poleceń;
- podwójne (stereo) kamery – umożliwiają to, co pojedyncza kamera, tylko możliwe jest pozyskanie obrazu trójwymiarowego i rozpoznawanie większej liczby gestów i ruchów ciała;
- kamery wykorzystujące strukturalne oświetlenie – oświetlają przestrzeń niewidzialną chmurą punktów (promieni) i rejestrują ich położenie; na podstawie odczytów położenia istnieje możliwość wykrycia sylwetki, i zmiany w położeniu – gestów; do najbardziej znanego urządzenia działającym na tej zasadzie należy Kinect firmy Microsoft – rys. 2.15;
- kontrolery ruchu – trzymane w rękach przez człowieka, przekazują dane do komputera dotyczące różnych gestów; przykładowo mysz (poza ruchem i przyciskami) może przekazywać gesty np. z wykorzystaniem prawego przycisku i ruchu bocznego czy pionowego (rozpoznawalne przez przeglądarkę Opera), bądź też różnych sekwencji wciskania przycisków; możliwe jest także łączenie akcji myszą z wciskaniem klawiszy klawiatury (zwykle przycisków sterujących takich jak Ctrl, czy też Alt).

Rys. 2.15. Czujnik ruchu w 3D – Kinect
Źródło: http://pl.wikipedia.org/wiki/Kinect#mediaviewer/File:KinectSensor.png

Interfejs gestowy znajduje zastosowanie w aplikacjach [24] medycznych do obróbki dużych ilości danych (np. zdjęć rentgenowskich lub obrazów tomografu komputerowego), w systemach zarządzania kryzysowego, komunikacji człowiek-robot (np. uczenie robota określonych działań), rozrywki (np. sterowanie grami komputerowymi lub telewizorami), a także do komunikacji z osobami niepełnosprawnymi.

Zastosowania medyczne wynikają z oczywistych względów. Interfejs gestowy nie wymaga dotykania urządzeń, a więc jest bardziej higieniczny. Poza tym eksploracja dużych ilości danych przy pomocy gestów jest bardzo efektywna i intuicyjna [24].

By interfejs gestowy znalazł zastosowanie w aplikacjach powinien posiadać cały szereg właściwości, takich jak [24]:

- **niska cena** – suma wartości sprzętu i oprogramowania powinna być rozsądna z punktu widzenia zastosowań;
- **responsywność** – system wyposażony w interfejs gestowy powinien rozpoznawać gesty w czasie rzeczywistym, bez zbędnych opóźnień (opóźnienia poniżej 45 ms są odbierane przez człowieka jako natychmiastowa reakcja, a powyżej 300 ms – jako reakcja opieszała) i przy małej liczbie błędów;
- **adaptowalność** – właściwość uczenia się nowych gestów poprzez ich zapamiętywanie przez interfejs, a także korygowanie akcji użytkownika przez interfejs, tj. uczenie użytkownika właściwych gestów;
- **zapamiętywalność** – gesty rozpoznawane przez interfejs powinny być łatwe do zapamiętywania i ponownego użycia; osiąga się to poprzez użycie tzw. naturalnych gestów człowieka lub też poprzez wykorzystanie istniejących i rozpowszechnionych, w niektórych grupach ludzi, standardów (np. język migowy osób niesłyszących lub też język gestów nurków – patrz: http://www.nurkomania.pl/nurkowanie_znaki_rekami.htm);
- **dokładność** – na to pojęcie składają się trzy elementy poprawnie realizowane: wykrywanie dłoni, jej śledzenie oraz rozpoznawanie gestów dłonią lub jej częściami;
- **niskie obciążanie psychiczne użytkownika** – gesty powinny być proste, krótkie, naturalne i wykorzystujące poziom nawyków przetwarzania przez człowieka;
- **intuicyjność** – nawiązywanie gestami do typowych, stosowanych powszechnie przez ludzi;
- **komfortowość** – unikanie gestów silnie i/lub długotrwale obciążających mięśnie człowieka, mogących doprowadzać to tzw. syndromu „ramion goryla”;
- **łatwość użycia** – interfejs nie powinien wymagać użycia dodatkowych, specyficznych i trudnodostępnych elementów technicznych, takich jak: markery, rękawiczki, obcisłe ubranie, specjalne tło lub oświetlenie;

- **rekonfigurowalność** – możliwość łatwego dostosowywania się do użytkowników, którzy mogą sie różnić cechami antropometrycznymi, tj. kolorem skóry i włosów, długością włosów i ich układem, rozmiarami, kształtem twarzy czy innych części ciała);
- **odporność** – niektóre gesty człowieka przed kamerą mogą nie być w pełni zamierzone lub kontrolowane przez niego (np. podrapanie się po swędzącym nosie) i mogą być nieprawidłowo interpretowane jako sterujące;
- **mobilność** – system interfejsu gestowego, w szczególnych przypadkach, powinien być mobilnym i podążać za użytkownikiem przy zmianie jego położenia.

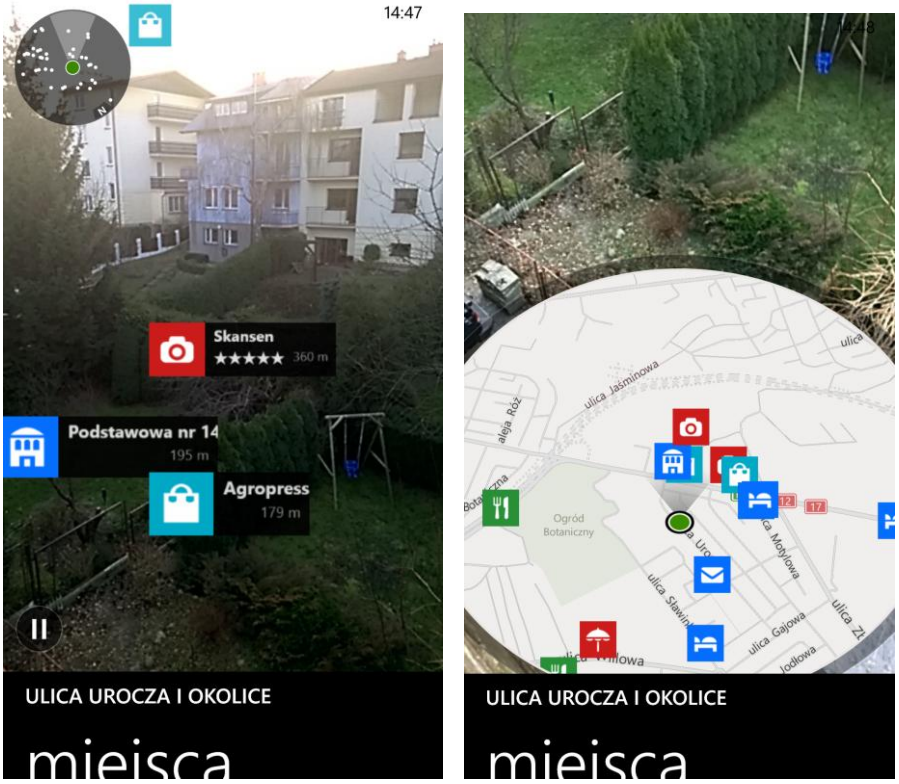
Rozszerzona rzeczywistość łączy obrazy rzeczywiste, zwykle powstałe poprzez filmowanie na bieżąco otoczenia człowieka, z obrazami sztucznie wygenerowanymi. Jednym z pierwszych tego typu interfejsów były te stosowane w lotnictwie bojowym – rys. 2.16. Pozwalają one projektować obrazy na szybie samolotu, co łączy je z widokiem za szybą. Celem takiego interfejsu jest dostarczenie pilotowi informacji wzrokowej bez przemieszczania wzroku na monitor w kokpicie. Skraca to czas reakcji pilota i przeciwdziała brakowi postrzegania otoczenia w czasie odczytywanie przyrządów.

Rys. 2.16. Projekcja obrazu na przedniej szybie samolotu
Źródło: http://www.pcworld.pl/news/373049/Rozszerzona.rzeczywistosc.ulatwia.zycie.html

Wraz z upowszechnieniem się smatfonów i tabletów wyposażonych w kamery i wyświetlacze coraz większej rozdzielczości, zaczynają sie pojawiać aplikacje wykorzystujące rozszerzoną rzeczywistość. Przykładem może służyć wyszukiwarka obiektów rzeczywistych w smartfonie, która wykorzystuje następujące technologie i urządzenia:

- system pozycjonowania (połączenie systemu GPS z pozycjonowaniem względem stacji bazowych GSM);
- mapę cyfrową;
- bazę danych obiektów, pogrupowanych wg. typów;
- wbudowany w smartfon kompas;
- obraz z kamery wideo smartfonu;
- ekran telefonu.

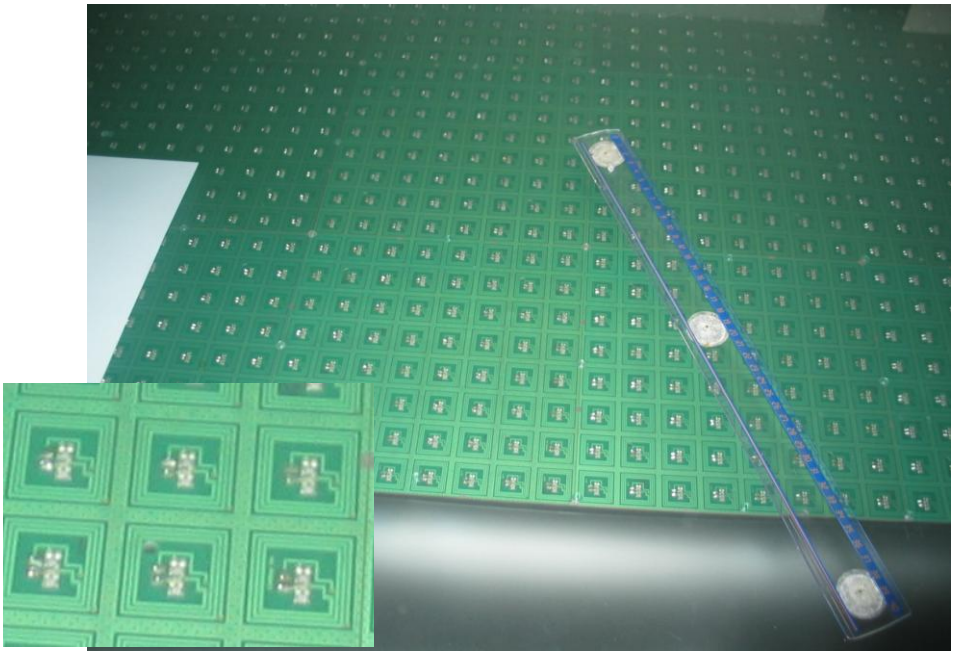
Całość może mieć wiele różnych widoków – rys. 2.17.



Rys. 2.17. Wykorzystanie wirtualnej rzeczywistości w wyszukiwarce obiektów
Źródło: opracowanie własne

Interfejsy do pracy grupowej (ang. *Shareable Interface*) muszą mieć duże powierzchnie, tak by parę osób mogło równocześnie oglądać generowane obrazy, a także wprowadzać swoje interakcje, tj. pracować w grupie, dużym zespole. Technologicznie mogą być to bardzo różne urządzenia: od uniwersalnych stołów rozpoznających i rejestrujących położenie różnych przedmiotów (rys. 2.18), poprzez różnego typu dedykowane rozwiązania (rys. 2.19), aż po tablice interaktywne (rys. 2.20).

Interfejsy do pracy grupowej znajdują zastosowanie wszędzie tam, gdzie duża liczba osób musi mieć dostęp do dynamicznie zmieniającej się informacji multimedialnej. Dodatkowo, część tych osób powinna mieć możliwość zmiany tej informacji. Interfejsy niwelują wadę pojedynczego stanowiska pracy i umożliwiają równoległą interakcję wielu osób. Znajdują zastosowanie w edukacji i rozrywce oraz w trakcie koncepcyjnej pracy zespołowej.

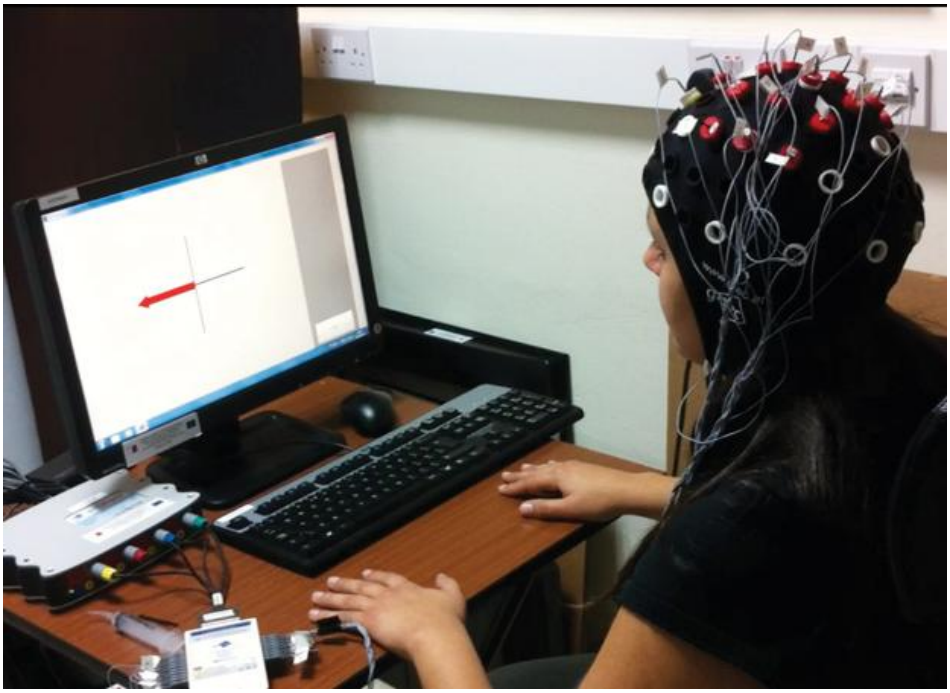


Rys. 2.18. Stół z czujnikami RFID (ang. *Radio-Frequency Identification*)
Źródło: opracowanie własne

Rys. 2.19. Dedykowany stół do działań zespołowych
Źródło: opracowanie własne

Rys. 2.20. Tablica interaktywna w szkole
Źródło: http://www.portel.pl/artykul.php3?i=63294

Sterowanie mózgiem (ang. *Brain-Computer Interface, BCI*) wykorzystuje czujniki odbierające promieniowanie elektromagnetyczne ludzkiego mózgu (tzw. fale mózgowe), które po odpowiedniej interpretacji używane są do sterowania komputerem. BCI czasami nazywane jest *Mind-Machine Interface (MMI), Direct Neural Interface (DNI)* czy też *Brain–Machine Interface (BMI)*. Interfejsy BCI poza technicznymi problemami (w tym głównie pozyskania i interpretacji sygnałów, konieczności odpowiedniego przeszkolenia) mają negatywny wydźwięk moralny i etyczny. Odczyt i prawidłowa interpretacja fal mózgowych zbliża człowieka do kontroli myśli. Systemy BCI bazują zwykle na systemach elektroencefalograficznych (ang. *Electroencephalogram, EEG*) – rys. 2.21.



Rys. 2.21. Sterowanie mózgiem
Źródło: http://www.um.edu.mt/think/moving-wheelchairs-with-your-thoughts/

W tab. 2.1 przedstawiono podsumowanie różnych metod interakcji w systemie człowiek-komputer (dotyczących różnych technik stosowanych w CLI, GUI i NUI) oraz ich podstawowe zalety i wady. Powinny być one uwzględniane w procesie projektowania i oceny rezultatu tego etapu przez klienta i użytkowników.

Tab. 2.1. Zalety i wady różnych metod interakcji człowiek-komputer

Metoda	Opis	Zalety	Wady
Wiersz poleceń	Wymaga poleceń ze ściśle określonego zestawu oraz o określonej składni	• oferuje wiele możliwości • jest elastyczny • przyciąga zaawansowanych użytkowników • pozwala na oszczędność przestrzeni ekranu • inne (np. uruchomienie wielu procesów przy pomocy jednego polecenia)	• komendy muszą być skatalogowane • wymaga uczenia się • brak tolerancji na błędy • trudny w użyciu dla początkujących użytkowników
Wybór z menu	Zestaw dostępnych funkcji, z których użytkownik może dokonać wyboru	• minimalizuje złożoność interakcji • wspomaga proces podejmowania decyzji • minimalizuje konieczność pisania • wspomaga użytkowników początkujących	• może spowalniać zaawansowanych użytkowników • wymaga dużo miejsca na ekranie • umożliwia tworzenie skomplikowanych struktur hierarchicznych
Formularze	Zestaw pól umożliwiających wpisanie danych lub ich wybór	• klasyczny format interakcji • ułatwia wprowadzanie danych • wymaga minimalnego treningu od użytkownika	• potrzebuje znacznej przestrzeni ekranu • wymaga ostrożności i efektywności w projektowaniu • nie ostrzega w przypadku błędów (w przypadku braku walidacji)
Manipulacja bezpośrednia	Umożliwia bezpośrednią interakcję z elementami prezentowanymi na ekranie	• przyspiesza uczenie się • ułatwia zapamiętywanie • umożliwia wstawianie podpowiedzi oraz wskazówek • ułatwia odkrywanie błędów • dostarcza kontekst działania • oferuje szybką reakcję	• większa złożoność przy projektowaniu • wymaga manipulacji oknem • wymaga rozpoznawalności ikon • duże prawdopodobieństwo zaburzenia układu elementów interfejsu • duże prawdopodobieństwo utraty skupienia się użytkownika na funkcjach
Antropomor-fizm	Personifikacja, nadawanie elementom cech ludzkich i motywów postępowania	• naturalność działania	• trudny w implementacji

Źródło: opracowanie własne na podstawie [3]

2.2. Środowisko fizyczne eksploatacji systemów informatycznych

Środowisko fizyczne opisujące stanowisko pracy składa się z następujących elementów:

- pomieszczenie,
- sprzęt komputerowy,
- meble,
- oświetlenie,
- człowiek i warunki jego pracy.

Na problemy natury ergonomicznej w trakcie pracy z systemami informatycznymi składają się:

- **zła organizacja pracy** – szybkie tempo narzucane przez warunki pracy (np. zniecierpliwiona kolejka przed kasą w supermarkecie i presja pracodawcy rejestrującego liczbę zeskanowanych kodów na minutę) oraz nieergonomiczne oprogramowanie (wymuszające dodatkowe, niepotrzebne czynności od operatora), prowadzące do stresu, zmęczenia, frustracji i wypalenia zawodowego;
- **intensywne wykorzystywanie wzroku** – konieczność odczytu z monitorów obrazowych, prowadzące do dolegliwości, zmęczenia itp., zwiększającego możliwość pomyłek;
- **długotrwała praca w siedzącej pozycji** – prowadzi ona do dolegliwości bólowych mięśni, kręgosłupa, systemu krążenia, szczególnie w kończynach dolnych (np. może powodować tzw. zakrzepicę).

Poza tym, praca przy komputerze naraża człowieka na oddziaływania pola elektromagnetycznego (przy monitorach LCD to zagrożenie praktycznie nie istnieje), jonizującego pola elektrostatycznego, ruchu powietrza (i związanego z tym zagrożeniem pyłkami, alergenami itp.) oraz na styczność z wydzielanymi przez urządzenia techniczne (komputery, monitory, urządzenia peryferyjne) gazami rakotwórczymi.

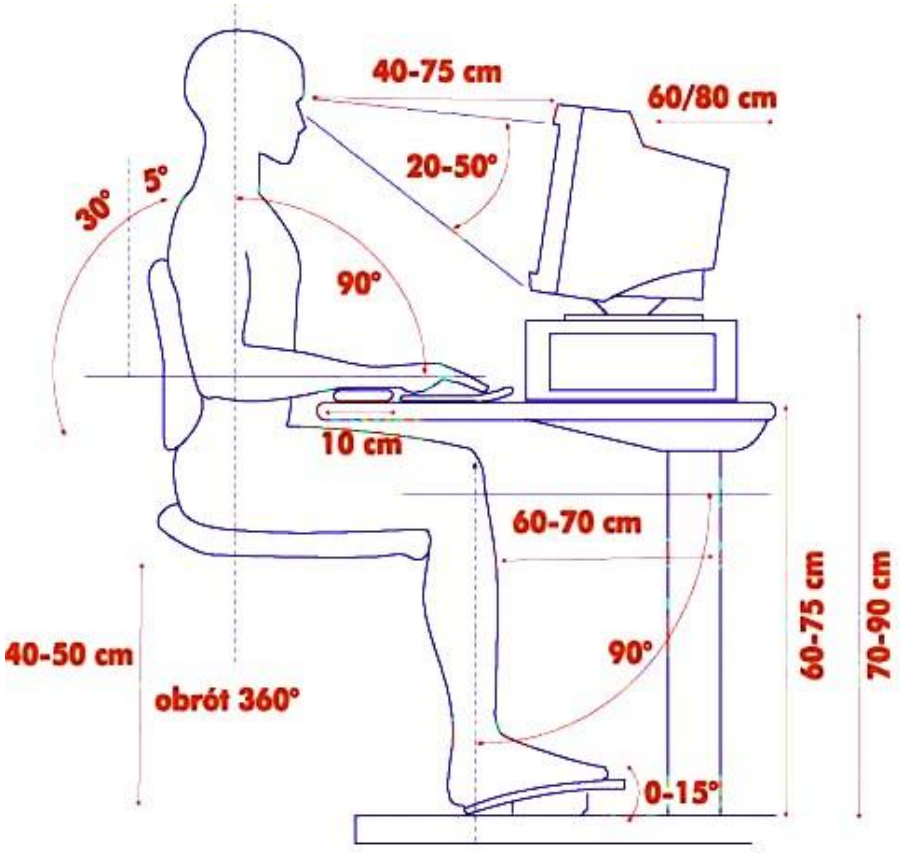
Organizację pracy stanowisk przy komputerze określa Rozporządzenie Ministra Pracy i Polityki Socjalnej w Dz. U. Nr 148 z 1998 r. Podstawowe wytyczne wyznaczają, że na każde stanowisko powinno być 13 m^3 objętości wolnego wnętrza i 2 m^2 wolnej powierzchni podłogi w pomieszczeniu. Poza tym ustawienie monitorów ekranowych względem siebie powinno uwzględniać następujące wymagania:

- odległość pomiędzy monitorami nie powinna być mniejsza niż 0,6 m;
- odległość od pracownika, a tyłem sąsiadującego monitora nie powinna być mniejsza niż 0,8 m.

Wspomniane rozporządzenie reguluje także inne parametry otoczenia i urządzeń pracy użytkownika systemów informatycznych, a mianowicie:

- parametry siedziska (krzesła) pracownika;
- parametry stołu roboczego (wysokość, powierzchnia, zakres regulacji wysokości, przestrzeń na nogi, materiał);
- jakość obrazu na monitorze (wielkość, ostrość, brak drgań i zniekształceń) oraz parametry odległościowe od operatora;
- parametry i umieszczenie klawiatury, tym kąt jej nachylenia w stosunku do powierzchni stołu.

Podstawowe parametry techniczne stanowiska komputerowego przedstawione zostały na rys. 2. 22.



Rys. 2.22. Stanowisko pracy z komputerem – wymagania formalne
Źródło: [25]

Poza technicznymi uwarunkowaniami pracy z komputerem, Rozporządzenie Ministra Pracy i Polityki Socjalnej zaleca działania związane z organizacją pracy, a mianowicie:

- robienie 5-10 minutowych przerw w pracy co najmniej raz na godzinę;
- w szczególnych przypadkach przerwy powinny być 15 minutowe realizowane co 2 godziny,
- przerwy powinny być w innej pozycji niż siedząca, bez korzystania w tym czasie z komputera.

Poza tym, zaleca się ograniczanie pracy z monitorami ekranowymi kobiet w ciąży do maksymalnie 4. godzin dziennie. Wszyscy pracownicy powinni być poddaniu badaniom, w tym okulistycznym, co maksimum 3 lata. Pracownikom pracującym z monitorami ekranowymi lekarz może/powinien zlecić pracę w okularach ochronnych, których zakup odbywa się na koszt pracodawcy (tylko wartość szkieł) pod warunkiem pracy przy komputerze w wymiarze min. 50% czasu ich dobowego wymiaru.

2.3. Jakość interfejsu oprogramowania

Interfejs oprogramowania, a w zasadzie cały system człowiek-komputer-otoczenie, powinien podlegać dwóm zasadom wspomagania pracy człowieka, tzw. prawom Raszkina (ang. *Raskin's Laws of Interface Design*) [5]:

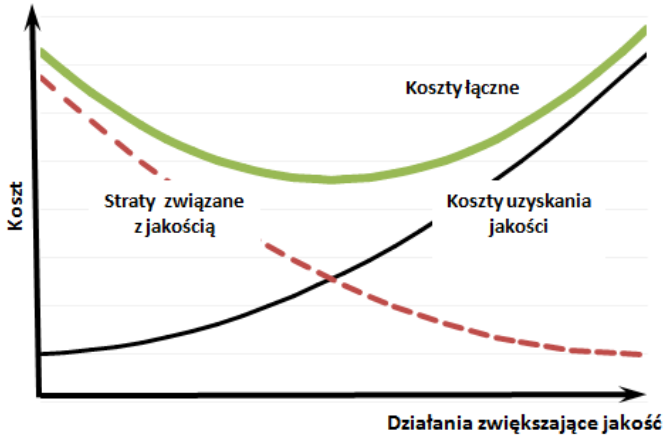
1. Komputer nie może zaszkodzić pracy człowieka lub poprzez swoją bezczynność doprowadzać do takiej szkody.
2. Komputer nie może tracić czasu człowieka i wymagać więcej jego pracy niż jest to absolutnie konieczne.

Jakość oprogramowania ma wymiar ekonomiczny. Zła jakość doprowadza do znacznego wzrostu kosztów eksploatacji systemów informatycznych. Oprócz kosztów błędów i ich usuwania, pojawiają się dodatkowe koszty związane z dodatkowymi czynnościami, które musi wykonać użytkownik w trakcie pracy. W opracowaniu [3] autor wylicza ile dodatkowych osobolat kosztuje eksploatacja dużego systemu. Przy konieczności obsługi 4,8 mln. ekranów rocznie i konieczności spędzenia przy obsłudze jednego ekranu o 1. sekundę więcej niż wynosi potrzeba, straty te wynoszą 0,7 osobolat, a przy 5 sekundach – aż 3,6 osobolat. Zaproponowana w [13] metodyka pozwala wyznaczyć metodą obliczeniowo-eksperymentalną koszt złej jakości interfejsu.

Istnieje wiele definicji jakości systemów informatycznych. Zwykle posługują się one pojęciem „zgodności z wymaganiami użytkownika" i „zaspokojenia potrzeb klienta", czasem „zgodności ze standardami" lub też definiują jakość jako „ogół cech i właściwości wyrobu lub usługi, które decydują o zdolności wyrobu lub usługi do zaspokojenia stwierdzonych lub przewidywanych potrzeb"

(nieaktualna, ale wciąż stosowana norma ISO 8402). Definicje te bardzo ogólnikowo traktują problem. Oprogramowanie może być zgodne z wymaganiami użytkownika, ale całkowicie bezużyteczne, bo np. użytkownik nie dodefiniował pewnych obszarów funkcjonalności. Systemy informatyczne mogą także zaspokajać „stwierdzone potrzeby" pomimo, że raz na godzinę system się zawiesza. Z drugiej strony, jeśli potrzeby użytkownika są związane ze 100% gotowością systemu (np. internetowy czy mobilny dostęp do konta bankowego), to żaden system ich nigdy nie spełni.

Jakość oprogramowania, niezależnie jak jest pojmowana i mierzona, związana jest w ścisły sposób z kosztami. Na koszty jakości składają się dwa elementy: koszt zapewnienia jakości i koszt strat, związanych z jej brakiem. Oba koszty mają przeciwstawne tendencje, tzn. pierwszy rośnie wraz z wysiłkami projakościowymi a drugi maleje (rys. 2.23). Łączny „koszt jakości" posiada zatem pewien punkt optymalny – jest to właściwa jakość oprogramowania. Działania zwiększające jakość powyżej optymalnej nie mają uzasadnienia ekonomicznego. Powodują one wzrost łącznych kosztów posiadania systemu przez użytkownika.

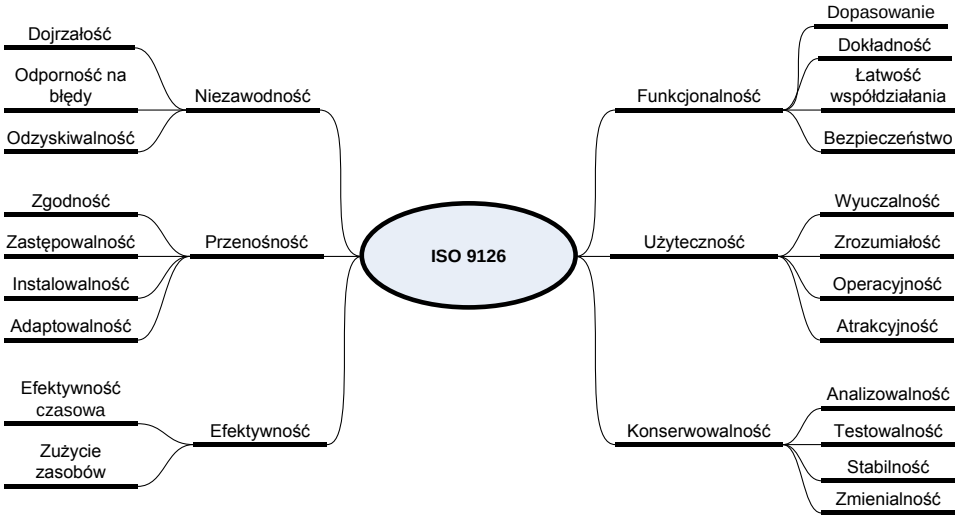


Rys. 2.23. Tendencje zmiany kosztów oprogramowania, związanych z działaniami projakościowymi
Źródło: opracowanie własne

Cechy opisujące jakość oprogramowania, jej parametry czy też kryteria oceny nie są i nie mogą być jednorodne. Na system informatyczny można spojrzeć z wielu różnych stron – przekrojów. Każdy z nich oceniany jest przy pomocy różniących się kryteriów. Łączna ocena nie jest prostą (czy nawet

ważoną) sumą poszczególnych ocen. Kompleksowa ocena jakości nie jest trywialnym problemem. Każdy system informatyczny można przedstawić z różnych stron i wyróżnić w nim (zgodnie z kanonami inżynierii oprogramowania) struktury-przekroje: funkcjonalną, informacyjną, oprogramowania, techniczną i przestrzenną. W zasadzie w każdym z tych przekrojów można zdefiniować specyficzne kryteria jakości.

Cechy jakości oprogramowania są definiowane w wielu metodykach, normach i publikacjach. Definicje te bardzo często są dość zróżnicowane. Norma ISO 9126:2001 definiuje sześć grup cech opisujących jakość zewnętrzną (ang. *External Metrics*) oprogramowania, składających się z całego szeregu cech składowych – (rys. 2.24).



Rys. 2.24. Drzewo cech jakości oprogramowania wg ISO 9126:2001
Źródło: opracowanie własne na podstawie [7]

Norma ISO 9126:2001 wprowadza także pojęcie **jakości użytkowej** (ang. *Quality in Use Metrics*) oprogramowania. W jej skład wchodzą cztery cechy: efektywność, produktywność, bezpieczeństwo i satysfakcja – rys. 2.25. Ich definicje przedstawia tab. 2.2. Jakość użytkowa jest pochodną zewnętrznej jakości oprogramowania, wpływającej na skutki działania oprogramowania.

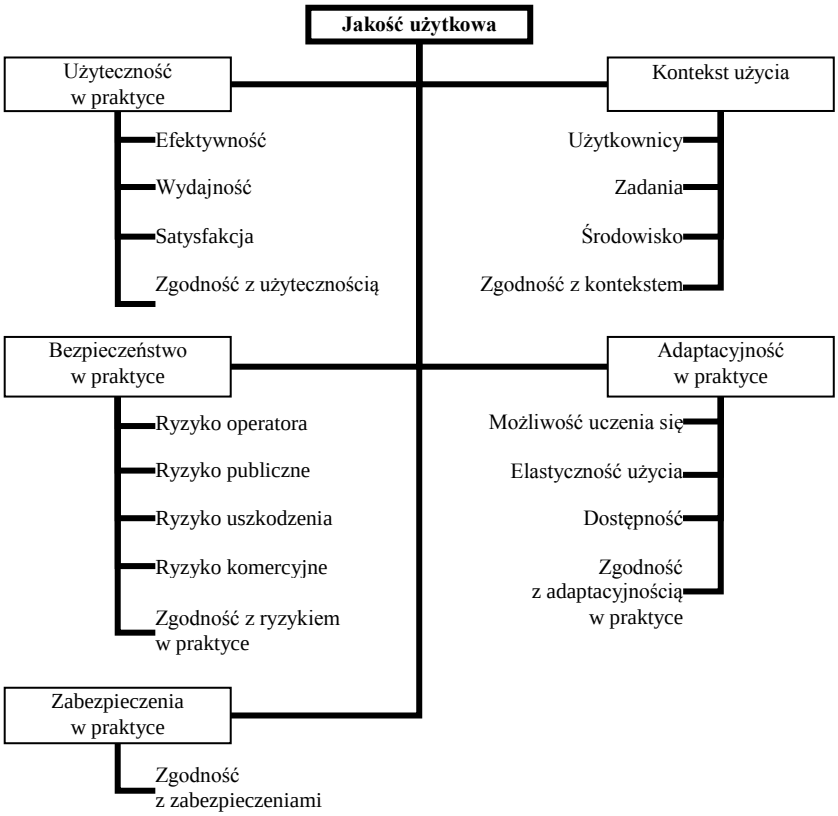
Rys. 2.25. Drzewo cech jakości użytkowej oprogramowania wg ISO 9126:2001

Źródło: opracowanie własne na podstawie [7]

Tab. 2.2. Definicje cech jakości użytkowej oprogramowania wg ISO 9126:2001

Cecha	Definicja
Efektywność (ang. *Effictiveness*)	Zdolność oprogramowania do osiągania zadanych celów z zadaną dokładnością
Produktywność (ang. *Productivity*)	Efektywna (potrzebna) wydajność
Bezpieczeństwo (ang. *Safety*)	Zdolność do długotrwałej, poprawnej pracy oprogramowania
Satysfakcja (ang. *Satisfaction*)	Zadowolenie klienta w trakcie użytkowania oprogramowania (a także po)

Źródło: opracowanie własne na podstawie [7]



Rys. 2.26. Drzewo cech jakości użytkowej oprogramowania wg ISO/IEC 25010:2011
Źródło: opracowanie własne na podstawie [13]

Norma ISO 9126:2001 została zastąpiona w 2011 roku normą ISO/IEC 25010:2011 [13], która definiuje już osiem grup cech jakości (identycznych dla jakości zewnętrznej i wewnętrznej) – tab. 2.3 i skomplikowała pojęcie jakości użytkowej – rys. 2.26.

Tab. 2.3. Cechy jakości oprogramowania wg ISO/IEC 25010:2011

Cecha	**Podcecha**
Funkcjonalność (ang. *Functionality*)	• Przydatność • Dokładność • Zgodność z funkcjonalnością
Bezpieczeństwo (ang. *Security*)	• Odporność dostępu • Odporność kopii • Szyfrowalność • Odporność na manipulacje • Uodpornienie • Zgodność
Interoperacyjność (ang. *Interoperability*)	• Zgodność z OSI • Kompatybilność programu • Kompatybilność danych • Możliwość śledzenia • Zgodność
Niezawodność (ang. *Reliability*)	• Dojrzałość • Tolerancja błędów • Możliwość odzysku • Zgodność niezawodności
Użyteczność (ang. *Usability*)	• Odpowiedniość • Możliwość nauczenia się • Operacyjność • Pomocniczość • Zadowolenie • Zgodność użyteczności
Efektywność (ang. *Efficiency*)	• Zachowanie czasowe • Zużycie zasobów • Zgodność efektywności
Możliwość utrzymania (ang. *Maintainability*)	• Możliwość analizy • Możliwość zmian • Stabilność • Możliwość testowania • Zgodność utrzymania
Przenośność (ang. *Portability*)	• Adaptacyjność • Współegzystencja • Zamienność • Zgodność przenośności

Źródło: opracowanie własne na podstawie [13]

Istnieje cały szereg niestandardowych cech jakości (jakościowych charakterystyk oprogramowania), do których można zaliczyć takie cechy, rzadko spotykane w literaturze, jak:

- **agnostycyzm danych** – wspieranie wszystkich możliwych formatów danych (w najprostszym przypadku: przecinka oraz kropki dziesiętnej) oraz radzenie sobie z zakłóceniami – brak radzenia prezentuje rys. 2.27;
- **intuicyjność interfejsu** – zrozumiałość elementów interfejsu; jest to cecha bardzo trudna do uzyskania w przypadku oprogramowania przeznaczonego na różne rynki, w takim przypadku dość trudno wykorzystać typowe metafory w różnych obszarach kulturowych na Ziemi – problem ilustruje rys. 2.28;
- **zachęcanie** – interfejs zachęca użytkownika do jego poznawania w sposób pasywny (np. poprzez elementy interfejsu wywołujące chęć ich wybrania) lub aktywny – pojawianie się różnych „podpowiadaczy", które wspomagają proces uczenia się użytkowników poprzez wskazanie miejsca/wywołania informacji sugerowanej użytkownikowi;
- **minimalizm** – brak zbędnych „upiększaczy" interfejsu, które nic nie wnoszą dla użytkownika i jego postrzegania, a tylko skutecznie przeszkadzają, rozpraszając go;
- **charyzma interfejsu** – posiadanie przez oprogramowanie „tego czegoś", co powoduje, że aplikacja jest postrzegana przez użytkowników jako poprawna jakościowo.

Rys. 2.27. Przykład braku agnostycyzmu danych: formularz z wypełnionymi polami danych i reakcja aplikacji na dane
Źródło: opracowanie własne

Rys. 2.28. Przykład różnego zrozumienia znaczenia „Co robi proszek do prania?” przez osoby czytające od lewej strony do prawej (kultura Zachodu) i odwrotnie (kultura Wschodu)
Źródło: bardzo stare – utracone

Charyzma interfejsu (czyli „to coś” w interfejsie aplikacji) jest nieformalnym parametrem jego jakości. Żadna norma nie standaryzuje tego pojęcia. Na charyzmę składają się (lub przynajmniej przyczyniają się do niej) następujące cechy i właściwości oprogramowania:

- **unikalność** – aplikacja wyraźnie wyróżnia się swoimi cechami na tle innych podobnych (lub zrealizowanych w podobnych technologiach); dla znacznej liczby użytkowników jest to bardzo ważny parametr, rzutujący na kupno oprogramowania;
- **profesjonalizm** – interfejs wywołuje u użytkownika wrażenie, że aplikacja została wykonana w sposób profesjonalny na najwyższym poziomie; oprogramowanie powinno posiadać nie tylko „wrażenia”, ale też realne informacje o realnej realizacji celów;
- **zaciekawienie** – interfejs odwołuje się do cechy człowieka (ciekawości) i zainteresowania użytkownika do eksploracji możliwości oprogramowania (tzw. wciągnięcie użytkownika);
- **nowość** – stosowanie nowinek w interfejsie, które można wykorzystać w promocji produktu lub też w celu zwiększenia zainteresowania użytkownika w trakcie szkolenia i eksploatacji systemu;
- **innowacyjność** – oprogramowanie spełnia najświeższe oczekiwania użytkownika, rozbudzone reklamami nowości oraz wynikające z jego naukowych badań lub przemyśleń;
- **pierwsze wrażenie**, czyli pozytywne przedstawienie interfejsu i aplikacji, tak by użytkownik chciał do niej powracać lub, by powracał z innych powodów, ale z pozytywnym, odczuciem;
- **historia** – w historii rozwoju aplikacji są ciekawe, może nawet medialne fakty i historie; takie „rodzynki” zawsze zwiększają zainteresowanie, a więc i charyzmę produktu.

2.4. Użyteczność i dostępność

Użyteczność (ang. *Usability*) jako cecha jakości zewnętrznej oprogramowania, występuje we wszystkich systemach i normach jakości oprogramowania. Cecha ta nabiera istotnego znaczenia dla aplikacji, które nie są wdrażane tradycyjnymi sposobami. Są to aplikacje, których użytkownik ma się nauczyć sam, bez wyraźnej pomocy kogokolwiek. Do takich aplikacji zalicza się przede wszystkim aplikacje internetowe i mobilne.

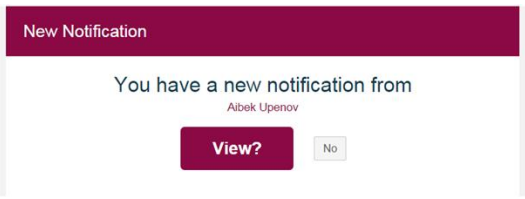
Użyteczność nie jest właściwością jednowymiarową. Składa się ona ze zbioru elementów, które definiują cechy oprogramowania, a mianowicie:

- **łatwość nauki** (ang. *Learnability*) – aplikacja powinna być łatwa do nauki, tak by użytkownik przy pierwszym kontakcie z nią był w stanie wykonać zadania, które ma do wykonania, a produktywność jego pracy z aplikacją szybko narastała w czasie;
- **efektywność** (ang. *Efficiency*) – aplikacja powinna być efektywna w trakcie korzystania z niej, tj. powinna wymagać od użytkownika dokładnie tyle czasu na wykonanie zadania, ile jest to niezbędnie konieczne;
- **zapamiętywalność** (ang. *Memorability*) – system powinien być łatwy do zapamiętania, tak aby użytkownicy mogli szybko przypomnieć sobie jak korzystać z systemu po dłuższej przerwie, tj. by ich produktywność po dłuższej przerwie w pracy z aplikacją nie spadała zbyt dużo;
- **stopa błędów** (ang. *Errors*) – system powinien mieć niski współczynnik błędów, aby podczas korzystania z systemu błędy nie występowały, a w razie ich wystąpienia, by było możliwe odzyskanie poprzedniego stanu systemu;
- **satysfakcja** (ang. *Satisfaction*) – system powinien być taki w obsłudze, by użytkownicy byli subiektywnie zadowoleni z korzystania z niego.

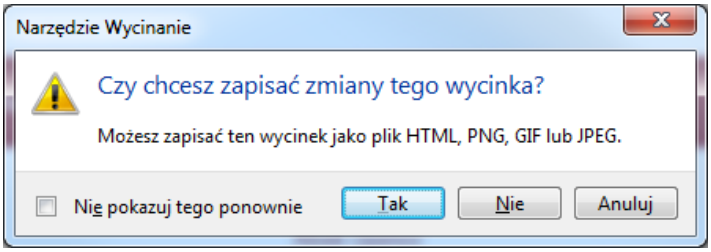
Podczas projektowania interfejsu użytkownika, w tym GUI, przyjęte rozwiązania silnie rzutują na jego użyteczność. Do często spotykanych problemów należą:

- Trudność uczenia się – projektanci oprogramowania powinni pamiętać o tym, że użytkownicy każdego systemu muszą poznać metody i sposoby pracy z jego interfejsem. Przyjęte rozwiązania powinny maksymalnie skracać czas i wysiłek związany z nauką korzystania z oprogramowania i sprzyjać zapamiętywalności. Dotyczy to wszystkich elementów graficznych interfejsu, ich układu, ale także elementów tekstowych, i ich zachowań. Sprzyja temu stosowanie **metafor** (ang. *Metaphor*) związanych z życiem codziennym.
- Przeciążenie ekranu czy też okna nadmierną liczbą elementów, nielogicznie i nieestetycznie rozmieszczonych powoduje wydłużenie czasu poszukiwania właściwego obiektu.

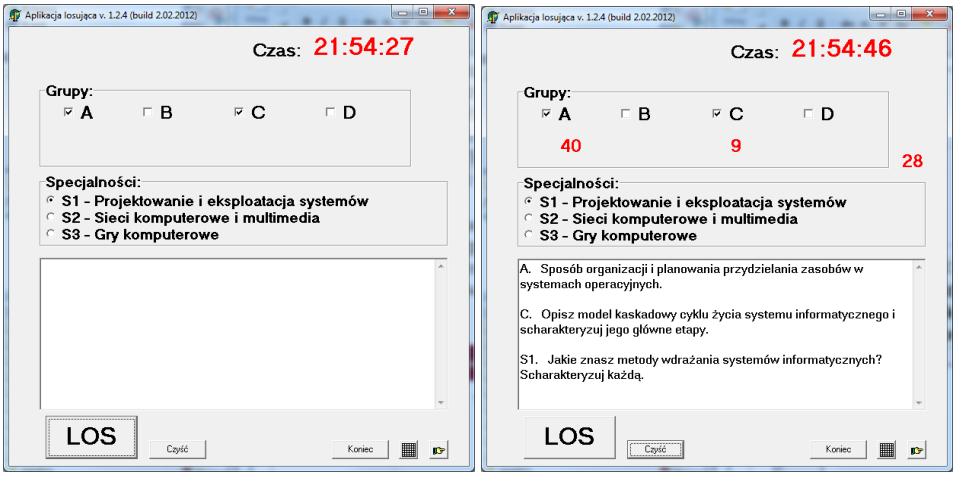
- Nadmierna innowacyjność, czy też oryginalność interfejsu – twórcy interfejsu dążą do uczynienia swojej aplikacji wyróżniającej się od innych, poprzez wprowadzenie nietypowych elementów interfejsu. Takie niestandardowe rozwiązania wywołują u użytkowników dezorientację, zwiększają wysiłek uczenia się obsługi systemu, a także przyczyniają się do popełniania przez użytkowników błędów – rys. 2.29.
- Brak spójności i przewidywalności interfejsu – dotyczy to stosowanych elementów, ich układu oraz słownictwa. Użytkownik zaskakiwany pytaniem typu: „Czy chcesz anulować?" z opcjami: Anuluj/Potwierdź nie bardzo wie co robić, co może doprowadzić do błędu, a na pewno wydłuża czas obsługi.
- Nieprawidłowy kształt lub rozmiar obiektów – rozmiary zbyt małe powodują trudności w postrzeganiu, a niekiedy nawet przyczyniają sie do zniekształcania danych (np. obcinanie napisów). Może to powodować utrudnienia w pracy z aplikacją. Zbyt duże rozmiary niepotrzebnie zajmują powierzchnię ekranu, przyczyniając się do zwiększenia czasu potrzebnego do wyboru funkcji interfejsu.
- Nieprawidłowy dobór kolorów, powodujący utratę kontrastu lub szybkie zmęczenie oczu (np. zielone napisy na czerwonym tle lub odwrotnie).
- Ograniczenie możliwości korzystania z klawiatury, wymuszające nieustanne przemieszczanie ręki z klawiatury na mysz (np. wypełnianie formularza na zasadzie: kliknij w polu, wpisz z klawiatury, kliknij w następnym polu…). Poza tym, część użytkowników (zwykle tych bardziej zaawansowanych) woli korzystać w dużym zakresie z klawiatury podczas pracy z aplikacją GUI. Robi to, wykorzystując skróty klawiszowe, przemieszczanie się przy pomocy *Tab* czy też *Shift+Tab* lub innych klawiszy i ich kombinacji znacznie szybciej niżeli przy wykorzystaniu myszy. Należy też umiejętnie zdefiniować kolejność przemieszczania się po elementach GUI przy pomocy klawisza *Tab* (ang. *Tab Order*). Tam gdzie się da, miejsce skupienia – fokus (ang. *Focus*) powinno być ustawiane automatycznie na obiekcie najbardziej prawdopodobnym do wybrania w ramach kolejnego kroku interakcji – rys. 2.30 (na przycisku *Tak*) i 2.31 (odpowiednio na przyciskach: *Los* i *Czyść* w zależności od stanu aplikacji).



Rys. 2.29. Niezwyczajne zróżnicowanie wielkości przycisków
Źródło: opracowanie własne



Rys. 2.30. Ustawienie skupienia w miejscu najbardziej prawdopodobnym (pożądanym)
Źródło: opracowanie własne



Rys. 2.31. Automatyczne ustawianie skupienia w zależności od stanu aplikacji i stosowanie skrótów klawiszowych: *a, b, c, d* oraz *1, 2, 3* do zmiany stanu odpowiednich elementów
Źródło: opracowanie własne

Zła użyteczność oprogramowania może doprowadzać do następujących problemów pracy z nim:

- **zniechęcenie** – oprogramowanie nie zachęca do odkrywania jego możliwości; użytkownik w takiej sytuacji zwykle przestaje wykorzystywać aplikację (np. internetową) bądź też znacznie się ogranicza funkcjonalnie;
- **brak spójności** – zaskakiwanie użytkownika; zwykle wydłuża to czas realizacji operacji za pomocą interfejsu oraz może doprowadzać do niezamierzonych błędów użytkownika;
- **brak dostępności** – użytkownik nie może w pełni korzystać z aplikacji, z powodu swoich osobistych ograniczeń (np. brak rozróżnialności kolorów lub też niewielki monitor).

- **ukrycie funkcjonalności** – działania w aplikacji są trudno odkrywalne (GUI aktywnie korzysta z tego mechanizmu poznawczego – rys. 2.5) przez użytkownika bez czytania instrukcji lub szkoleń; przykładem tego problemu może służyć interfejs niektórych samochodów, w którym wykorzystanie funkcji wymaga wykonania trudno zapamiętywanych sekwencji czynności (typu: przekręć kluczyk trzy razy w czasie nie dłuższym niż 2 sekundy, wciśnij przycisk przełączania trybu pracy komputera i przytrzymaj go przez co najmniej 3 sekundy);
- **brak poczucia kontroli nad oprogramowaniem** (jest to jeden z elementów satysfakcji) – system robi coś bez poinformowania użytkownika lub też robi coś, czego użytkownik nie spodziewa się; użytkownik jest zaskakiwany działaniami oprogramowania i traci związek przyczynowo-skutkowy pomiędzy własnymi działaniami a reakcją systemu;
- **brak możliwości dopasowania** – użytkownik nie może zmienić domyślnych ustawień zgodnie ze swoją wolą; przykładowo, nie może wyłączyć wyskakującego „podpowiadacza" jeżeli już go nie potrzebuje i musi tracić czas, usuwając go za każdym razem po jego pojawieniu się; pozytywnym przykładem jest opcja „Nie pokazuj tego ponownie" na rys. 2.30 ze skrótem klawiszowym *Alt+e*.

Dostępność oprogramowania (ang. *Accessibility*) to jego cecha, świadcząca o tym, że jego funkcje lub właściwości może używać szerokie grono ludzi. Brak dostępności uniemożliwia, lub w znacznej mierze ogranicza, użycie oprogramowania. Brak ten może być związany z ograniczeniami pochodzącymi od:

- samego człowieka,
- używanego sprzętu i oprogramowania,
- otoczenia, tj. środowiska pracy.

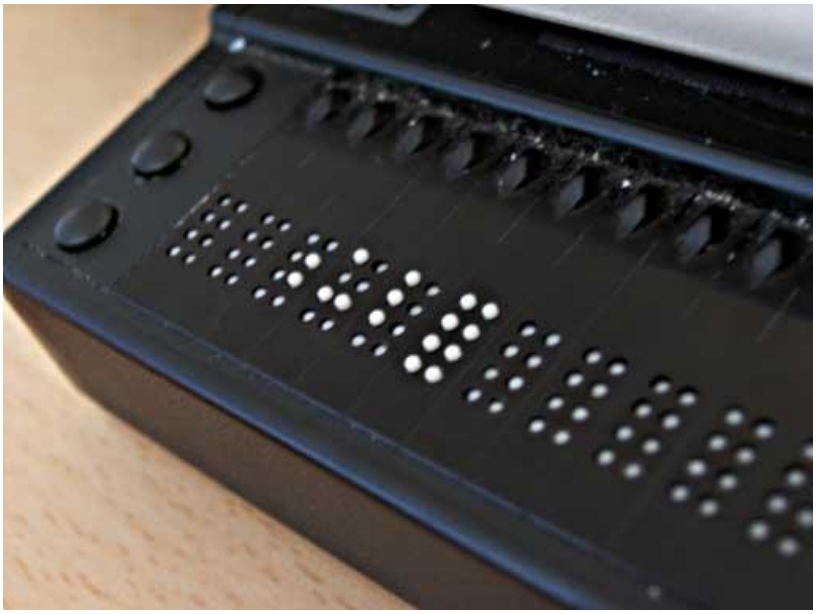
Ograniczenia wprowadzane przez człowieka związane są z jego niepełnosprawnością:

- **wzrokową** – polega ona na różnym poziomie ograniczeniu widzenia, od obniżenia ostrości wzroku lub nieprawidłowego postrzegania kolorów do całkowitej ślepoty;
- **słuchową** – polega ona na niedosłyszeniu w różnym zakresie: od tylko niektórych dźwięków, aż po całkowitą głuchotę;
- **motoryczną** – obejmuje ona trudności lub brak możliwości wykonania pewnych działań fizycznych (np. przesunięcia myszy, precyzyjnego pozycjonowania kursora lub jednoczesnego wciśnięcia dwóch klawiszy na klawiaturze),
- **poznawczą** (kognitywistyczną) – polega na braku możliwości zrozumienia elementów interfejsu (ograniczenie umysłowe spowodowane niskim poziomem inteligencji użytkownika lub językowe – brak umiejętności zrozumienia języka interfejsu) lub poprawnego napisania poleceń (tzw. dysleksja).

Pozostałe ograniczenia, które zmniejszają liczbę użytkowników oprogramowania, mają naturę sprzętową (np. niska rozdzielczość ekranu, brak wymaganego urządzenia, niska wydajność procesora), programową (np. wersja używanej przeglądarki) oraz środowiskową (np. słabe oświetlenie lub na odwrót – bardzo jasne oświetlenie miejsca pracy).

Istnieje wiele elementów wpływających na dostępność oprogramowania w praktyce. Są to między innymi [23]:

- zawartość informacyjna i jej struktura (ang. *Content*);
- oprogramowanie użytkownika i jego interfejs (np. przeglądarki WWW, odtwarzacze multimediów);
- technologie wspierające (np. czytniki ekranu, lupy ekranowe, rozpoznawanie mowy, specjalne klawiatury, urządzenia wskazujące czy monitory wykorzystujące pismo Braille'a – rys. 2.32);
- użytkownicy i ich kompetencje (wiedza, doświadczenie, umiejętności);
- twórcy oprogramowania i dostawcy treści;
- narzędzia do oceny jakości interfejsu oprogramowania (np. oprogramowanie do oceny zgodności serwisu WWW z zaleceniami WCAG – ang. *Web Content Accessibility Guidelines*).



Rys. 2.32. Monitor Braille'a
Źródło: [25]

Zapewnienie dostępności interfejsu oprogramowania wynika z czysto biznesowych przesłanek oraz uwarunkowań prawnych.

Biznesowe uwarunkowania są dość oczywiste – producent oprogramowania zwykle chce by go mogło używać jak największe grono potencjalnych użytkowników.

Uwarunkowania prawne dostępności oprogramowania, związane są z polityką równości w dostępie do informacji. W konsekwencji we wszystkich wiodących dokumentach Unii Europejskiej (UE) pojawiają się wymagania niedyskryminacji w dostępie do danych (szczególnie tych publicznych). Dostępność gwarantuje m.in. Kodeks Unijnych Praw Internetowych, uchwalony w grudniu 2012 r. (http://ec.europa.eu/digital-agenda/en/code-eu-online-rights) oraz Prawo konsumentów w UE obowiązujące od 13 czerwca 2014 r. (http://ec.europa.eu/justice/consumer-marketing/rights-contracts/directive/).

Poza uregulowaniami UE, w Polsce cały szereg aktów prawnych gwarantuje równość w dostępie do informacji. Podstawowy akt prawny – Konstytucja RP – gwarantuje (art. 32 i 61) wszystkim obywatelom niedyskryminację i „prawo do uzyskiwania informacji o działalności organów władzy publicznej oraz osób pełniących funkcje publiczne". Brak dyskryminacji w tym zakresie ugruntowują ustawy „o dostępie do informacji publicznej" oraz „o informatyzacji działalności podmiotów realizujących zadania publiczne". Równe szanse zawarte są także w Karcie Osób Niepełnosprawnych.

2.5. Podsumowanie

Tryby interfejsów oprogramowania (tekstowy i graficzny) wynikają z właściwości urządzeń wyświetlających – monitorów. W typie tekstowym, w ograniczonym zakresie, można tworzyć obrazy z wykorzystaniem znaków. Z chwila pojawienia się trybu graficznego większość systemów informatycznych w nim pracuje.

Typy interfejsu oprogramowania zależą od sposobu oddziaływania człowieka na komputer i są następujące: wiersz poleceń, WIMP, organiczny, głosowy, dotykowy, gestowy, multimodalny, rozszerzona rzeczywistość. Interfejsy ewoluują od CLI, poprzez GUI, do NUI. Do specyficznych interfejsów należą interfejsy pracy grupowej i wykorzystujące fale mózgowe jako źródło poleceń.

Obiekty interfejsu oprogramowania wzbogaciły sie znacznie – szczególnie ich szata graficzna. Ze względów ergonomicznych i konieczności zapewnienia intuicyjności obsługi, konstruowane są one jako metafory świata rzeczywistego. Metafory te niestety są w znacznej mierze uwarunkowane kulturowo, co znacznie utrudnia skonstruowanie uniwersalnych obiektów.

Środowisko fizyczne eksploatacji systemów informatycznych ma istotny wpływ na ergonomię i produktywność pracowników je wykorzystujących. Istnieje cały szereg zagrożeń związanych z długotrwała pracą z wykorzystaniem

systemów informatycznych. Warunki eksploatacji systemów komputerowych określają przepisy prawa. Regulują one dość precyzyjnie cały szereg parametrów geometrycznych, organizacyjnych i zdrowotnych.

Jakość interfejsu oprogramowania ma istotny wymiar ekonomiczny. Zła jakość powoduje straty a wytworzenie wysokiej jakości – koszty. Cechy jakości oprogramowania definiują normy. Jakość może być zewnętrzną (postrzeganą przez użytkownika) i wewnętrzną. Grupy cech jakości to: funkcjonalność, bezpieczeństwo, interoperacyjność, niezawodność, użyteczność, efektywność, możliwość utrzymania i przenośność. Istnieje cały szereg niestandardowych jakościowych charakterystyk oprogramowania, takich jak charyzma, agnostycyzm czy minimalizm.

Użyteczność oprogramowania to łatwość nauki interfejsu, efektywność, zapamietywalność, niska stopa bledów oraz satysfakcja użytkowania. Cały szereg zaleceń, prawidłowo zastosowanych zwiększa użyteczność.

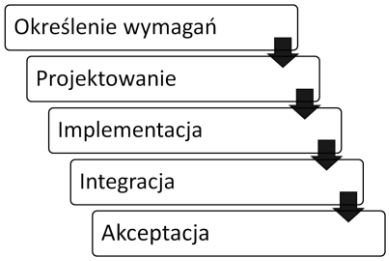
Dostępność oprogramowania wpływa na liczbę jego użytkowników. Konieczność jej zapewnienia wynika z przesłanek biznesowych i uregulowań prawnych.

3. Projektowanie interfejsów

Proces projektowania interfejsu jest jedną z faz wytwarzania oprogramowania definiowaną w inżynierii oprogramowania (ang. *Software Engineering, SE*). Ze względu na fakt, że dowolny system jest postrzegany przez użytkownika poprzez interfejs, jest to jedna z ważniejszych, aczkolwiek dotychczas mało docenianych, faz. Projektowanie i implementacja interfejsu jest kosztowna – w szczególnych przypadkach wynosi nawet 90% kosztów wytworzenia oprogramowania.

3.1. Ogólny schemat projektowania interfejsu

Proces projektowania interfejsu jest wpleciony w cykl wytwarzania oprogramowania i w konsekwencji silnie zależy od zastosowanego modelu. W typowym modelu kaskadowym (ang. *Waterfall Model*) miejsce projektowania interfejsu jest w fazach określania wymagań i projektowania (rys. 3.1).



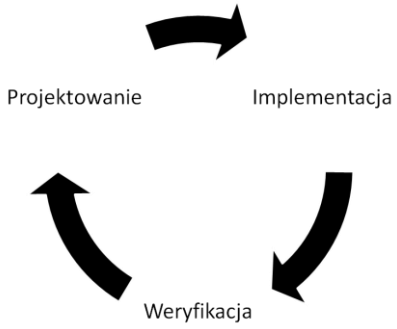
Rys. 3.1. Typowy, kaskadowy model wytwarzania oprogramowania
Źródło: opracowanie własne

W modelu kaskadowym każda faza jest oceniana pod kątem spełnienia wymagań zdefiniowanych w fazie poprzedzającej. Jeśli więc zostały zidentyfikowane wymagania w stosunku do interfejsu w fazie określania wymagań, to projekt sprawdzany jest pod kątem ich spełnienia. Analogicznie, rezultat implementacji (tj. oprogramowanie) jest testowany nie tylko pod kątem poprawności jego wykonania (tzw. testy modułów), ale też pod kątem spełnienia wszystkich elementów (w tym i interfejsu) projektu. Takie weryfikacje (a właściwie walidacje – ang. *Validations*) przeprowadzane są dość formalnie i nie wymagają uczestnictwa użytkownika. W modelu kaskadowym rozróżnia się dwie specyficzne fazy: określanie wymagań i ich akceptacja (testy akceptacyjne). W obydwu tych fazach użytkownik aktywnie uczestniczy w weryfikacji rezultatów fazy: specyfikacji wymagań i gotowego oprogra-

mowania. Wywołuje to bardzo poważne problemy. W fazie definiowania wymagań są one określane „na sucho" i dość ogólnikowo. Użytkownik nie jest w stanie w prawidłowy sposób określić wymagania w stosunku do interfejsu i jego jakości. Koncentruje się raczej na funkcjonalności oprogramowania, marginalnie traktując ergonomię. Projektant oprogramowania, bez kontaktu z użytkownikiem, też nie jest w stanie zaprojektować poprawnego i ergonomicznego interfejsu. Ergonomia silnie zależy bowiem od użytkownika i jego cech. Wadliwość interfejsu jest identyfikowana dopiero w fazie testowania akceptacyjnego, które jest wykonywane z dużym udziałem użytkownika. Niestety błędy identyfikuje się w gotowym i przetestowanym oprogramowaniu. Ich poprawienie wymaga cofnięcia się do fazy określania wymagań i przejścia całego procesu od początku. Trwa to długo i dużo kosztuje.

Wiele różnych programów wytwarzanych jest w modelu cyklicznym, tj. takim, w którym określone fazy wielokrotnie się powtarzają. Modeli cyklicznych jest wiele. Można wymienić takie jak: iteracyjny (ang. *Iterative Model*), przyrostowy (ang. *Incremental Model*), i spiralny(ang. *Spiral Model*). Cykliczne podejście wykorzystują też metodyki zwinne (ang. *Agile*), takie jak: Scrum, programowanie ekstremalne (ang. *eXtreme Programming*) czy też FDD (ang. *Feature Driven Development*).

Uogólniony model cykliczny wytwarzania oprogramowania został przedstawiony na rys. 3.2.

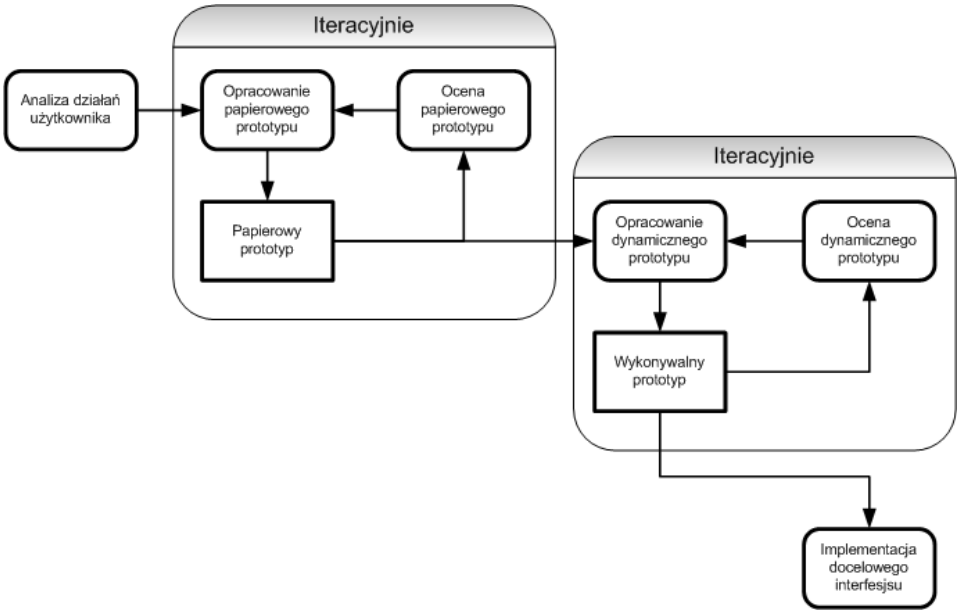


Rys. 3.2. Ogólny cykliczny model wytwarzania oprogramowania
Źródło: opracowanie własne

Projektowanie interfejsu użytkownika musi uwzględniać następujące aspekty związane z użytkownikiem:

- psychiczne, umysłowe i motoryczne zdolności użytkowników;
- ograniczenia w pamięci krótkoterminowej człowieka;
- skłonność do popełniania błędów.

Najlepszą zatem metodą projektowania interfejsu jest metoda doświadczalno-badawcza, w której przyszły użytkownik jest aktywnym uczestnikiem, testującym jakość interfejsu. Proces ten ilustruje rys. 3.3.



Rys. 3.3. Proces projektowania interfejsu z oznaczonymi procesami cyklicznymi
Źródło: opracowanie własne na podstawie [22]

Proces projektowania interfejsu metodą doświadczalno-badawczą (rys. 3.3) polega na cyklicznym tworzeniu i testowaniu prototypów interfejsu, równolegle do faz określania wymagań i projektowania w modelu kaskadowym (rys. 3.1). Na początkowym etapie tworzone są proste (i zużywające mało zasobów) prototypy interfejsu, które w kolejnych fazach są komplikowane, pozwalając uzyskać lepsze charakterystyki ergonomiczne. Kolejne iteracje powinny polepszać interfejs. Możliwa jest także równoległa budowa wielu prototypów (ang. *Parallel Design*), ich przetestowanie i wybór najlepszego.

Poza ukierunkowaniem na użytkownika, do podstawowych cech efektywnego i ergonomicznego interfejsu, które powinny być brane pod uwagę w trakcie jego projektowania, można zaliczyć:

- **Przewidywalność** – użytkownik w każdej chwili powinien przewidywać działania oprogramowania, nie powinien być nimi zaskakiwany.
- **Dostępność**, czyli możliwość pełnego korzystania ze wszystkich funkcjonalności oferowanych przez oprogramowanie.

- **Przystępność** – składa się na nią prostota, brak błędów (lub minimalizacja częstotliwości i konsekwencji ich występowania) oraz możliwość obsługi systemu przez osoby o zróżnicowanych umiejętnościach, doświadczeniu i zdolnościach motorycznych. Do tej cechy zalicza się także posługiwanie się w interfejsie pojęciami i kategoriami wziętymi z dziedziny podmiotowej, zrozumiałej dla użytkownika.
- **Estetyka** – wpływa na nią wiele czynników, takich jak: stosowanie estetycznych elementów interfejsu, ich właściwe grupowanie, topologia oraz właściwa separacja grup. Na estetykę wpływa także właściwe wykorzystanie barw. Ułatwia to podkreślenie wagi elementów interfejsu, a jednocześnie zwiększa jego atrakcyjność.
- **Przejrzystość interfejsu**, tj. graficzna i lingwistyczna poprawność jego elementów. Wszystkie elementy składowe interfejsu powinny być określone w sposób konkretny i zwięzły a terminologia musi być ujednolicona. Elementy graficzne powinny być intuicyjne i jakościowo poprawne.
- **Konfigurowalność**, tj. możliwość personalizacji oprogramowania, dostosowująca interfejs do wymagań i upodobań różnych grup użytkowników, a także zmieniająca go w miarę ich uczenia się. Elementy konfigurowalności to przede wszystkim możliwość określania własnych skrótów klawiaturowych, zmiana układu elementów interfejsu, wyłączenie różnego typu automatycznych pomocników, czy też dostosowanie kolorystyki interfejsu do własnych potrzeb.
- **Spójność** – polega na ujednoliceniu wyglądu i sposobu działania elementów o podobnej funkcjonalności i znaczeniu. Powinny one mieć tożsamy wygląd i umiejscowienie. Ta sama akcja powinna za każdym razem zwracać ten sam oczekiwany rezultat, a funkcje i pozycja poszczególnych elementów nie powinny być zmieniane. Niekiedy ta zasada jest łamana – przykładem jest ukrywanie opcji niemożliwych do wykonania lub też dawno nieużywanych w układach menu niektórych programów. Łamanie to wynika z chęci uproszczenia interfejsu poprzez zmniejszenie liczby jego elementów.
- **Kontrola nad działaniami oprogramowania** – użytkownik powinien wiedzieć co aktualnie dzieje się w systemie. Kontrola ta objawia się poprzez: szybko wykonywane akcje, wyczerpujące komunikaty o błędach i wskazówki odnośnie działania systemu, informowanie o chwilowym opóźnieniu w działaniach, a także w możliwości odwoływania uruchomionych działań.
- **Standaryzacja**, tj. wykorzystanie wzorców interfejsu (elementów oraz zachowań człowieka) znanych użytkownikowi i zgodnych z ogólnie używanymi w danym układzie (np. systemie operacyjnym). Zwiększa to intuicyjność pracy oraz zastępuje działania oparte na wiedzy czy regułach przez nawyki. Wzrasta w takim przypadku szybkość pracy użytkownika.

- **Efektywność i wydajność pracy użytkownika** – jest realizowana poprzez taką konstrukcję interfejsu, która minimalizuje ruch gałek ocznych oraz rąk, a także ogranicza konieczność użycia pamięci krótkookresowej.
- **Tolerancja na popełniane błędy** – jest to właściwość interfejsu i oprogramowania polegająca na przewidywaniu i neutralizowaniu błędnych działań użytkownika. Poza kontrolą działań użytkownika (np. kontrola poprawności wprowadzanych danych) polega ona na dostarczaniu zrozumiałych i wyczerpujących komunikatów oraz podpowiedzi na temat popełnionych błędów. Poprawnie zaprojektowany komunikat wyjaśnia sytuację, która spowodowała jego wyświetlenie i wskazuje użytkownikowi rozwiązanie danego problemu.
- **Intuicyjność**, w tym łatwość uczenia się i szybkiego zrozumienia poszczególnych funkcji. Użytkownik w każdej chwili powinien wiedzieć jaką operację wykonuje i dlaczego.

Projektując interfejs należy w szczególności zwrócić uwagę na jego intuicyjną strukturę, która wyraża się w zorientowaniu interfejsu na użytkownika, jego właściwościach i ograniczeniach, a także potrzebach. Proces projektowania interfejsu oprogramowania, uwzględniający użytkownika na wszystkich jego etapach, nazywany jest **projektowaniem ukierunkowanym na użytkownika** (ang. *User-Centered Design, UCD*). UCD jest częścią szerszego paradygmatu wytwarzania ergonomicznego i użytecznego oprogramowania. UCD zakłada działania cykliczne, które składają się z czterech kolejno następujących po sobie faz (rys. 3.4):

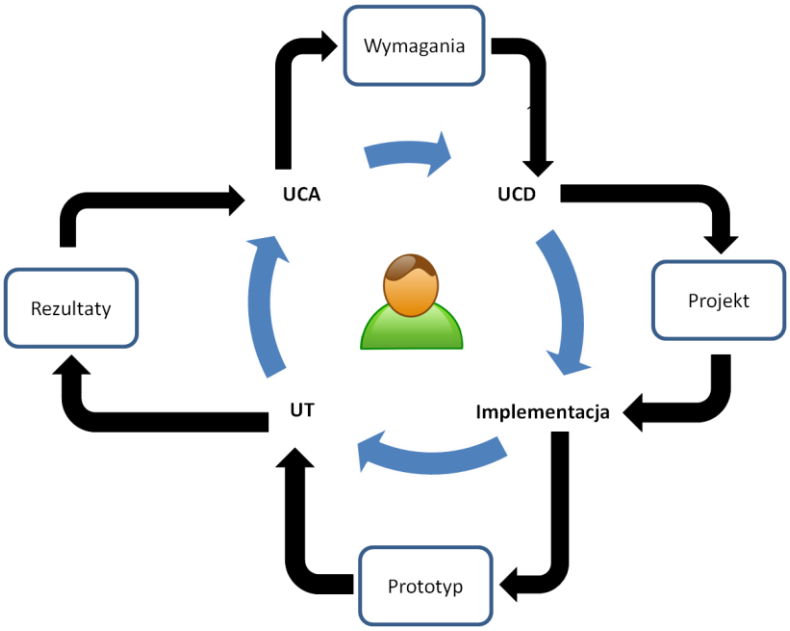
- **analiza** ukierunkowana na użytkownika (ang. *User-Centered Analysis, UCA*);
- **projektowanie** ukierunkowane na użytkownika (UCD);
- **implementacja** projektu (zwykle: budowa prototypu);
- **testowanie** użyteczności (ang. *Usability Testing, UT*) – ocena jakości prototypu.

W trakcie poszczególnych faz powstają różne produkty, które są wykorzystywane w następnych fazach. Są to (rys. 3.4):

- **wymagania** użytkownika w stosunku do interfejsu (ale też i całego systemu);
- **projekt** nowej wersji interfejsu, wynikający ze zidentyfikowanych wymagań;
- **prototyp** kolejnej wersji interfejsu;
- **rezultaty** testowania (wyniki pomiarów, uwagi, pomysły i kierunki zmian w interfejsie itd.).

Proces projektowania interfejsu ukierunkowanego na użytkownika realizowany jest w **modelu spiralnym**. Model ten zakłada wielokrotne powtarzanie tych samych faz (tj. działania cykliczne) z kolejnym uszczegóławianiem rezultatów. W początkowych cyklach działań definiowane

są ogólne wymagania, projektowane i implementowane proste prototypy (np. w postaci rysunków na papierze). W końcowych cyklach wymagania są znacznie bardziej uszczegóławiane, a prototypy – oprogramowywane, urealniane i uzupełniane o elementy dynamiczne. Takie typowe podejście: „od ogółu do szczegółu" podyktowane jest faktem, że w początkowym okresie użytkownik nie ma ukształtowanych wymagań i ekonomicznie nieuzasadnione jest opracowywanie szczegółowych projektów-prototypów-badań. W kolejnych cyklach użytkownik zdobywa wiedzę oraz uzmysławia sobie swoje wymagania, co w konsekwencji pozwala tworzyć bardziej dokładne modele i ich implementacje interfejsów.



Rys. 3.4. Proces projektowania interfejsu z procesami cyklicznymi i ich produktami
Źródło: opracowanie własne na podstawie [20]

Proces powstania interfejsu ukierunkowanego na użytkownika został ustandaryzowany w normie ISO [20]. Standard ten wprowadza sześć kluczowych czynników świadczących o tym, że proces jest ukierunkowany na użytkownika. Są to [25]:

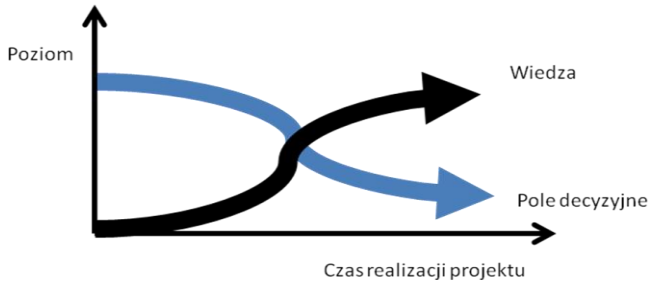
1. Proces bazuje na dogłębnym zrozumieniu użytkowników, ich celów i zadań oraz środowiska ich realizacji.
2. Użytkownicy są aktywnie zaangażowani w trakcie całego procesu projektowania i implementacji systemu.

3. Proces projektowania jest sterowany przez opinie i oceny użytkowników.
4. Proces jest cykliczny.
5. Proces wykorzystuje na bieżąco wszystkie wrażenia użytkowników (ang. *User eXperience, UX*) o produkcie.
6. Zespół realizujący projekt posiada interdyscyplinarne umiejętności i wieloperspektywiczne podejście.

Projektowanie ukierunkowane na użytkownika uwzględnia trzy podstawowe elementy [25]:

- **Odbiorcę** programu (użytkownika), w tym jego wiek, miejsce życia, płeć, pochodzenie etniczne (uwarunkowania kulturowe), wykształcenie i doświadczenie, cechy indywidualne (w tym ograniczenia) i inne, które mają wpływ na ergonomie pracy z oprogramowaniem.
- **Cel** w jakim będzie wykorzystywane oprogramowanie, w tym zadania jakie ma do wykonania użytkownik z wykorzystaniem oprogramowania.
- **Kontekst użycia** oprogramowania, czyli sytuacje w jakiej oprogramowanie będzie używane – ogół czynników zewnętrznych w jakich oprogramowanie będzie eksploatowane, w tym czynników socjalnych i kulturowych.

Projektowanie ukierunkowane na użytkownika ma wiele zalet. Najważniejsze jest zwiększenie wiedzy na temat celów użytkownika oraz uświadomienie użytkownikowi tych celów, na wstępnych etapach wytwarzania oprogramowania. Zwykle na początku projektu zwykle wiedza o jego istocie jest mała a decyzje dotyczące produktu nie zostały jeszcze podjęte – twórcy mają duże pole decyzyjne. W trakcie realizacji projektu wiedza wzrasta, a pole decyzyjne zawęża się na skutek już podjętych i zrealizowanych decyzji. W uproszczony sposób obrazuje to rys. 3.5. Zwiększenie poziomu wiedzy w początkowym etapie projektu umożliwia podjęcie bardziej trafnych decyzji, bowiem na tym etapie projektant ma szersze pole decyzyjne.



Rys. 3.5. Zmiana poziomu wiedzy i pola decyzyjnego w czasie trwania projektu
Źródło: opracowanie własne

Proces UCD w zakresie projektowania interfejsu jest zwykle jednym z etapów modelu wytwarzania oprogramowania (na przykład kaskadowego). Interfejs projektowany jest w dwóch fazach [10]:

- Faza 1 – projekt ogólny, realizowany z uwzględnieniem UX (ang. *User eXperience*).
- Faza 2 – projekt graficzny, opracowywany przez grafika komputerowego na podstawie uzgodnionego projektu interfejsu z fazy 1.

Faza 1 realizowana jest zwykle kaskadowo lub spiralnie i składa się z następujących etapów:

- Opracowanie specyfikacji wymagań.
- Utworzenie koncepcji.
- Projekt struktury aplikacji.
- Budowa prototypu interfejsu.
- Badania prototypu.
- Opracowanie raportu z badań z użytkownikami.
- Opracowanie dokumentacji funkcjonalnej oprogramowania.

Faza 1 kończy się po opracowaniu dokumentacji i zaakceptowaniu jej przez klienta. Dokumentacja (w tym dokumentacja funkcjonalna, prototypy) po uzupełnieniu przez zweryfikowane projekty graficzne opracowane w Fazie 2. Przekazywana jest do programistów w celu ich implementacji.

Takie podejście, silnie uporządkowane i sformalizowane, ma określone wady. Można do nich zaliczyć [10]:

- Dużą pracochłonność opracowywania dokumentacji, zarówno funkcjonalnej jak i raportów z badań.
- Możliwość utraty spójności koncepcji w wyniku dwufazowego podejścia i dużego rozdzielenia faz w czasie.
- Szybką utratę aktualności opracowanej dokumentacji.
- Kolejni wykonawcy (graficy, programiści, testerzy,…) nie czują się współautorami produktu – wykonują jedynie zlecone prace, nie wnosząc do oprogramowania swoich pomysłów.

W celu minimalizacji wad procesu UCD w odniesieniu do projektu interfejsu powstała metoda **Lean UX** [10]. Łączy ona w sobie podejście zwinne (ang. *Agile*) z *lean startup* [25].

Lean UX minimalizuje ilość wytwarzanej dokumentacji i zwiększa komunikację w zespole realizującym projekt. Koncentruje ona wysiłki zespołu na wymyślaniu rozwiązań i ich doskonaleniu w spiralnym procesie iteracyjnym. Lean UX wykorzystuje typowe przyrostowe podejście, składające się z następujących działań:

- Opracowanie ogólnego prototypu interfejsu oprogramowania (zespół wykonawczy składa się z projektanta, grafika i klienta), w tym schematów i szablonów oraz stylów. Zbadanie go pod kątem użyteczności i UX.
- Akceptacja koncepcji przez klienta.

- Podział oprogramowania na mniejsze moduły.
- Sukcesywne projektowanie szczegółowe poszczególnych modułów (zespół wykonawczy składa się z projektanta i grafika), które po zatwierdzeniu przez klienta przekazywane są do oprogramowywania.

Podejście Lean UX, poza pozbyciem się balastu niepotrzebnej dokumentacji, umożliwia szybsze rozpoczęcie prac programistycznych i zrównoleglenie etapów projektowania i implementacji.

3.2. Analiza potrzeb użytkownika

Analiza potrzeb użytkownika (UCA) jest procesem poznawania i ewidencjonowania potrzeb użytkownika. Analiza obejmuje następujące obszary (w nawiasie podano pytania, na które odpowiada analiza):

- **Użytkownicy** (Kto jest użytkownikiem?).
- **Zadania** (Co użytkownik ma do zrobienia?).
- **Domena/dziedzina** (W jakim otoczeniu użytkownik pracuje?).
- **Wymagania** (Jakie wymagania, wynikające z poprzednich trzech części analizy, należy postawić w stosunku do projektu?).

W trakcie UCA, w zależności od konkretnych potrzeb, stosuje się różnorodne techniki pozyskiwania informacji, takie jak:

1. Ankietowanie.
2. Wywiady z użytkownikami.
3. Zogniskowany wywiad grupowy (ang. *Focus Group*).
4. Obserwacje działań użytkownika.
5. Zespołowe dyskusje użytkowników – burze mózgów (ang. *Brain Storms*),
6. Sesje JAD (ang. *Join Application Development*).
7. Sortowanie kart (ang. *Card Sorting*).
8. Analiza dzienników i zapisów.
9. Analiza zadań i działań.
10. Analiza zapisów programów śledzących (np. Google analytics).
11. Analiza opinii klientów.
12. Analiza raportów centrów obsługi klienta.

Pierwsze siedem technik zalicza się do tzw. metod bezpośrednich, a pozostałe do pośrednich.

Większość technik pozyskiwania informacji jest ogólnie znana. Do specyficznych technik można zaliczyć sesje JAD i sortowanie kart.

Technika JAD polega na zorganizowaniu wspólnej, grupowej pracy twórców i użytkowników oprogramowania. Uczestnicy sesji JAD odgrywają ściśle określone role: od sponsora oprogramowania poprzez ekspertów technicznych i dziedzinowych, aż po obserwatorów.

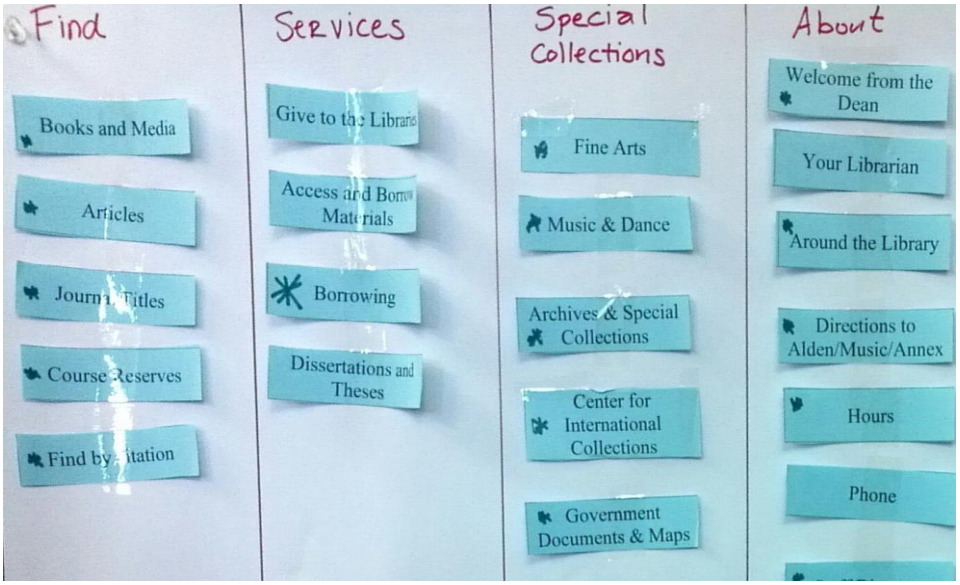
Przygotowanie do sesji JAD składa się z następujących podstawowych dziewięciu etapów [25]:

- Identyfikacja celów i ograniczeń projektu.
- Identyfikacja **kluczowych czynników sukcesu** (ang. *Critical Success Factors. CSF*) projektu.
- Definiowanie **produktów** dostarczanych w rezultacie projektu (ang. *Project Deliverables*).
- Opracowanie harmonogramów **spotkań roboczych** (ang. *Workshops*).
- Dobór uczestników spotkań roboczych.
- Przygotowanie materiałów do spotkań roboczych.
- Organizacja spotkań roboczych.
- Przygotowanie (poinformowanie. przeszkolenie) uczestników spotkań.
- Koordynacja i logistyka spotkania roboczego.

Sesję JAD przeprowadza się w trybie spotkania bezpośredniego.

Technika zwana **sortowanie kart** jest zespołową metodą klasyfikacji informacji. Polega ona na wspólnym (zespołowym) uporządkowaniu informacji, zwykle umieszczonych na samoprzylepnych karteczkach (stąd nazwa metody). Tworzy się w niej kategorie i przypisuje do nich poszczególne zidentyfikowane informacje. Umożliwia stworzenie architektury informacji, struktury zadań czy też układu menu itd. Technika ta może być stosowana w grupie lub poprzez serię indywidualnych sortowań z następującym potem uogólnieniem wyników. Sortowanie kart może się odbywać w trybie otwartym lub zamkniętym. **Otwarte sortowanie** (ang. *Open Card Sorting*) dopuszcza tworzenie nazw kategorii przez uczestników, w przeciwieństwie do **zamkniętego** (ang. *Closed Card Sorting*), w którym nazwy kategorii są predefiniowane (narzucone) i uczestnicy tylko przyporządkowują im obiekty informacyjne. Technika sortowania kart może być stosowana w trakcie roboczych spotkań zespołów złożonych z twórców i użytkowników oprogramowania – rys. 3.6. Dostępne są także komputerowe narzędzia wspomagające i obniżające koszt stosowania tej techniki.

Jakob Nielsen [16] wskazuje, że rekomendowana liczba uczestników badań techniką sortowania kart wynosi 15. Jest to trzy razy więcej niż w większości badań, związanych z użytecznością. Liczba ta wyjaśniana jest odmiennym charakterem badań – sortowanie kart jest techniką poznawczą (tj. tworzącą nową wiedzę), a nie oceniającą istniejące rozwiązanie [16]. Przy bardzo dużych projektach (a więc dużej liczbie informacji i kategorii) rekomendowane jest zwiększenie liczby uczestników badań nawet do 30. osób [16].



Rys. 3.6. Technika sortowania kart – biała tablica, kategorie i przyklejone karteczki
Źródło: https://www.library.ohiou.edu/2012/07/asking-our-users-card-sorting/

W trakcie UCA należy równolegle stosować wiele różnych technik pozyskiwania informacji. Pozwala to uniknąć tendencyjności rezultatów lub ominięcia czegoś, co może się zdarzyć, jeśli wykorzystuje się tylko jedną technikę. Przykładowo, w trakcie obserwacji działań użytkownika mogą nie wystąpić wyjątkowe sytuacje, które zostały opisane w dziennikach. Pominięcie analizy dzienników może doprowadzić do braku wykrycia i opisania tych sytuacji.

Proces UCA składa się z badania użytkowników oraz architektury informacji. Składa się on z następujących zadań [4]:

1. Opracowanie strategii (ang. *Design Strategy*) – planowanie działań.
2. Wyszukanie profili (ang. *Profiles*) i person (ang. *Personas*).
3. Opracowanie scenariuszy użycia oprogramowania (ang. *User Scenarious*).
4. Planowanie zadań (ang. *Tasks Design*).
5. Opracowanie architektury informacji (ang. *Content Architecture*).
6. Opracowanie nawigacji (ang. *Navigation Design*).

Strategię UCA tworzy się na początku projektu. Jest to etap definiowania najbardziej istotnych elementów przedsięwzięcia tworzenia oprogramowania, które potem są wykorzystywane w następnych etapach UCA.

Na strategię w analizie potrzeb ukierunkowanej na użytkownika składają się następujące elementy:

- **Cele marki** (ang. *Brand Goals*) – strategiczne cele w które wpisuje się dane oprogramowanie (np. wzrost zysków z działalności sklepu, wzrost niezawodności i jakości obsługi klientów).
- **Cele biznesowe** (ang. *Business Goals*) – uporządkowany zestaw celów użytkownika, do osiągnięcia którego ma przyczynić się tworzone oprogramowanie (np. pozyskanie nowych klientów, wzrost średniej wielkości koszyka zakupowego).
- **Użytkownicy końcowi** (ang. *Objective Users*) – grupa lub grupy osób, które będą bezpośrednio wykorzystywać rezultaty pracy oprogramowania (np. wszyscy potencjalni klienci sklepu internetowego, w tym upośledzeni ruchowo).
- **Zadania główne** (ang. *Main Tasks*) – zadania, które będą realizowali użytkownicy końcowi przy pomocy oprogramowania (np. przeglądanie i wybór towaru, zakup koszyka, śledzenie przesyłki).
- **Ograniczenia** (wymagania) **techniczne** (ang. *Technological Restrictions*) – niefunkcjonalne, istotne wymagania stawiane aplikacji (np. możliwość uruchomienia na urządzeniach stacjonarnych i mobilnych, wielojęzykowość interfejsu).
- **Czynniki krytyczne sukcesu** (ang. *Critical Success Factors, CFS*) – uwarunkowania (zwykle pochodzące z otoczenia), które jeśli się nie spełnią zagrożą sukcesowi projektu wytworzenia oprogramowania (np. wzrost cen usług kurierskich, pojawienie się bardziej konkurencyjnych internetowych sklepów zagranicznych dofinansowywanych przez rząd).

Zdefiniowane w strategii cele powinny być mierzalne. Pomiar może być binarny (spełniony vs. niespełniony) lub też wyrażać się wartością zdefiniowanego i sprawdzalnego po realizacji projektu miernika. Mierniki te mogą być ekonomiczne (np. wzrost zysków z działalności sklepu internetowego o 20%), jak i operacyjne (np. zmniejszenie do 20% liczby wejść do sklepu bez zrobienia zakupów lub też wzrost liczby wejść o 20%). W większości przypadków mierniki mają charakter przyrostowy. Wskazują bowiem na efektywność inwestycji w projekt. Wymagają więc pomiaru przed i po realizacji przedsięwzięcia. Jeśli stan początkowy wartości mierników nie jest znany to pomiar powinien być elementem UCA.

Często warunkiem realizacji przedsięwzięcia poprawy ergonomii interfejsu jest ekonomiczna efektywność takiej inwestycji. Wyraża się ona **wskaźnikiem zwrotu nakładów** na przedsięwzięcie (ang. *Return of Investment*, ROI) przez poprawę parametrów pracy z aplikacją. O ile koszty realizacji przedsięwzięcia są dość łatwo szacowane, to trudniej jest określić zyski.

Zysków z przedsięwzięcia związanego z poprawa jakości interfejsu oprogramowania należy szukać w następujących obszarach:

- wzrost produktywności pracowników wykorzystujących oprogramowanie,
- zmniejszenie obciążenia biura obsługi klientów (ang. *Help Desk*),
- zwiększenie liczby odwiedzin sklepu internetowego,
- zmniejszenie kosztu szkoleń (np. zmniejszenie liczby osobodni szkolenia w wyniku poprawy dostępności interfejsu),
- zmniejszenie liczby wyjścia ze sklepu bez zakupów,
- zmniejszenie strat czasu pracy na samo uczenie się pracowników,
- zmniejszenie liczby popełnianych błędów przez użytkowników.

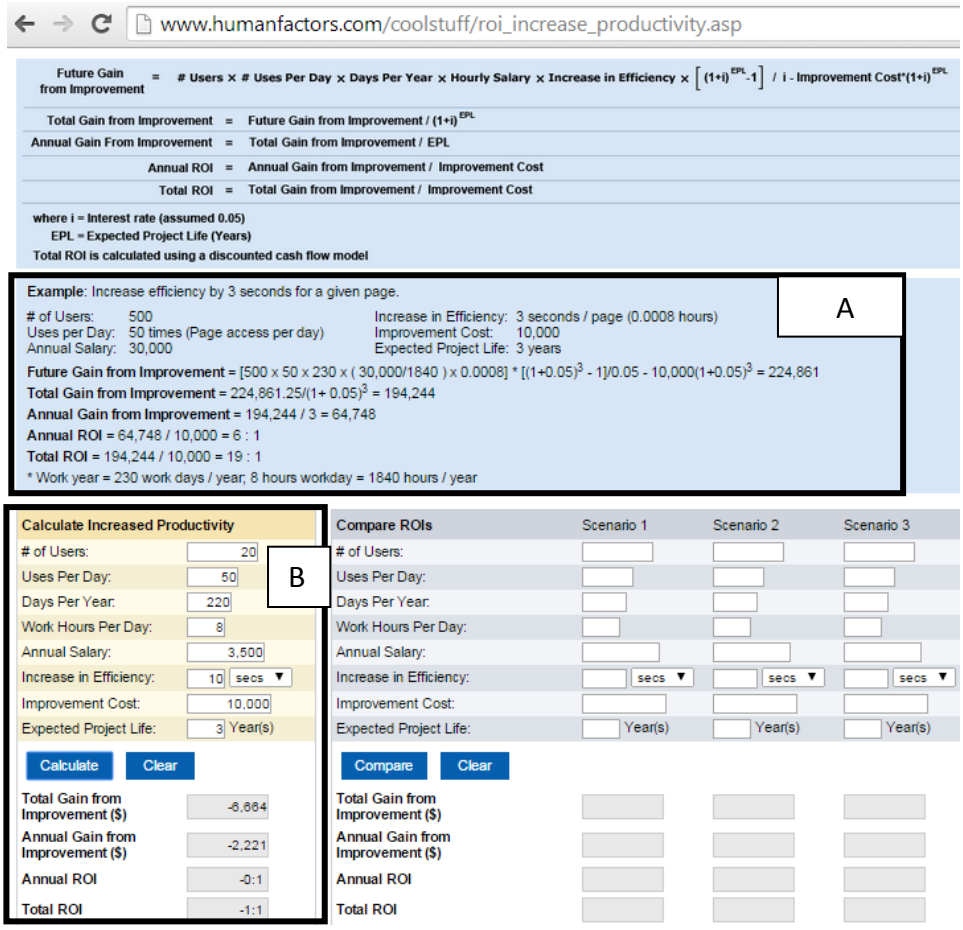
ROI jest wskaźnikiem ekonomicznym. Jego kalkulacja zawsze związana jest z porównaniem zysków i kosztów. Przykład kalkulacji ROI przedstawia rys. 3.7.

Kalkulacja w przykładzie (ramka A na rys. 3.7) wskazuje, że inwestycja w planowanym okresie trwałości rezultatów projektu zwraca się 19 krotnie. ROI może być także ujemne – oznacza to, że inwestycja przyniesie straty. Taki przypadek jest zilustrowany w ramce B na rys. 3.7. Inwestycja w polepszenie produktywności 20 pracowników przy użyciu oprogramowania, które jest używane 50 razy dziennie i zysku 10 s na jedno użycie przy pozostałych parametrach (wynagrodzenie, liczba dni pracy w roku, koszt przedsięwzięcia i 3 letni okres eksploatacji jego rezultatów) jest nieopłacalna. Przynosi ono bowiem straty w wysokości ponad 2,2 tys. zł rocznie, a całkowite straty przekraczają 6,6 tys. zł. Kalkulacja ROI powinna być przeprowadzona bardziej dokładnie niż oferują to ogólnie dostępne kalkulatory, bowiem w wyniku realizacji jednego projektu na ogół uzyskuje się równoczesne polepszenie wielu mierników (np. wzrasta produktywność i liczba klientów oraz zmniejsza się czas potrzebny na szkolenie lub też samokształcenie się personelu).

ROI może być użyte do wskazania tego, które cele projektu są ważniejsze i odpowiedniego zaplanowania działań w przypadku ograniczenia zasobów. W tym celu rozpatruje się różne warianty (scenariusze realizacji projektu – rys. 3.8) i szereguje je według wartości ROI. Należy jednak pamiętać, że często motorem przedsięwzięć zwiększających ergonomię interfejsu nie są pieniądze, ale wymagania prawne, prestiż firmy czy tylko zwykła moda.

Wyszukanie profili i person to kolejna faza procesu UCA. W fazie opracowania strategii określani są końcowi użytkownicy oprogramowania (odbiorcy). W kolejnej fazie procesu UCA, użytkownicy ci są identyfikowani i szczegółowo opisywani – modelowani. Wykorzystuje się przy tym pojęcia: profil i persona.

Grupy użytkowników są identyfikowane (np. przy pomocy techniki sortowania kart lub wywiadu) jako zbiory (klasy) odbiorców oprogramowania o identycznym charakterze w stosunku do projektowanego oprogramowania, np. klient sklepu internetowego zalogowany lub nie.



Rys. 3.7. Przykład kalkulacji ROI
Źródło: opracowanie własne przy pomocy kalkulatora ze strony http://www.humanfactors.com

Profil to zbiór ról użytkownika w systemie informatycznym, tj. zbiór zadań, które użytkownik może wykonać przy pomocy oprogramowania. Do specyfikacji profilu zalicza się także parametry środowiska wykonania zadań. Prowadzi to przykładowo, do zdefiniowania profili klienta aplikacji webowej i mobilnej lub też korzystającego z oprogramowania w miejscu pracy lub w domu. Konkretna grupa użytkowników może posiadać wiele profili. Profile można definiować dla zadań (np. kluczowe vs. wspomagające) i środowiska użytkowania oprogramowania (np. mobilne vs. webowe).

Persona to egzemplifikacja użytkownika w UCA (fikcyjny przedstawiciel-obiekt grupy użytkowników). Jest to sztuczna osoba, zdefiniowana na potrzeby projektu UCA jako zestaw określonych cech. Jest modelem abstrakcyjnym, który pozwala na:

- łatwiejsze zapamiętywanie właściwości grup użytkowników poprzez identyfikację ich z konkretną (aczkolwiek wyimaginowaną) osobą – uczłowieczenie fikcyjnego użytkownika;
- ułatwienie komunikacji pomiędzy członkami zespołu;
- pozbycie się paradoksu tzw. typowego, uśrednionego użytkownika – taki użytkownik łączy w sobie różne właściwości, w konsekwencji jest np. słonio-żyrafo-orło-osłem, czyli nie istnieje;
- połączenie w jedno charakterystyk z trzech profili: użytkownika, zadaniowego i środowiska;
- koncentracji na działaniach użytkownika;
- identyfikację i dołączanie do danych o projekcie, wydawało by się mało-ważnych, charakterystyk użytkowników; takie charakterystyki mogą się przydać w późniejszych etapach UCA.

Persona posiada imię i nazwisko oraz wygląd zewnętrzny (zdjęcie), także inne detale (np. wiek, płeć, pozycja socjalna, zawód), które pozwalają ją personifikować i mają ułatwiać zapamiętywanie i odróżnianie jej od innej persony. Takie podejście ułatwia zapamiętywanie właściwości i cech grup użytkowników.

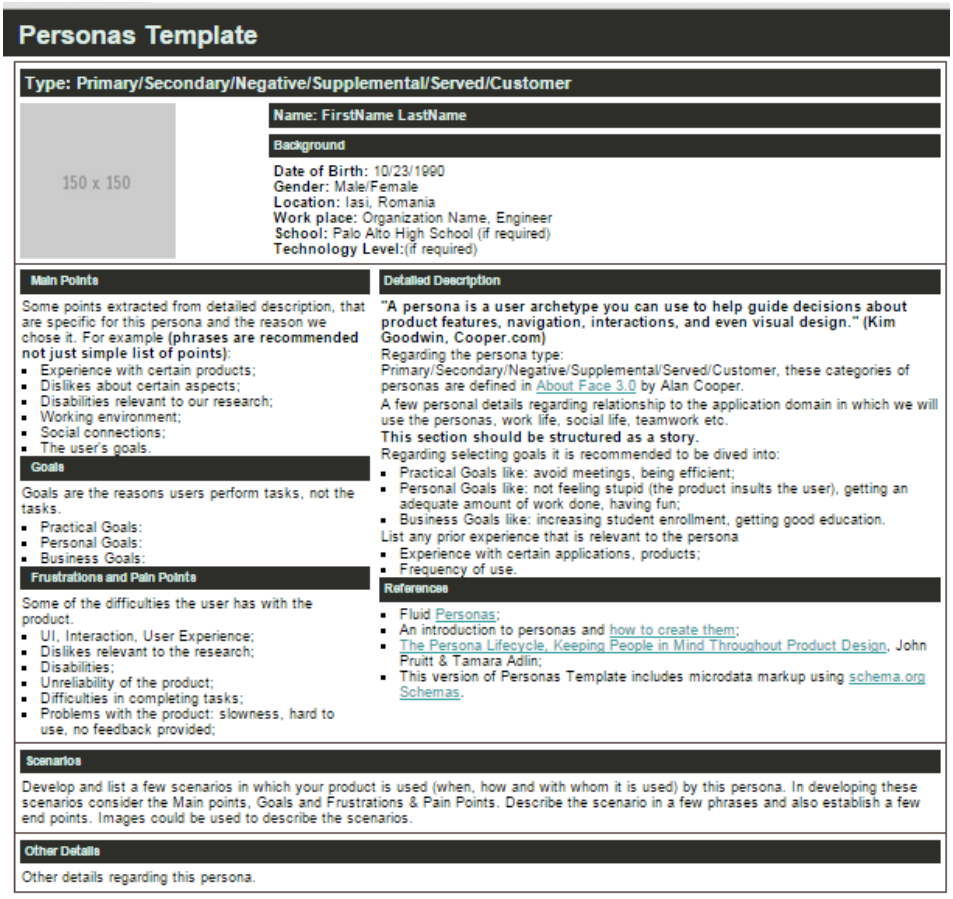
Poza wyimaginowanymi danymi, personie przypisuje się cechy związane z jej profilami: użytkownika, zadaniowym i środowiska.

Personę można opisać wykorzystując formularze (listy cech) – rys. 3.8 lub przedstawić w postaci opisowej. Postać opisowa jest rzadziej stosowana. Wynika to z faktu, że opis słowny jest mniej formalny i dopuszcza duże pole różnej interpretacji i rozumienia.

Opracowanie **scenariuszy użycia** oprogramowania polega na zdefiniowaniu dla każdej persony konkretnych działań realizowanych z użyciem oprogramowania. Scenariusz jest opisem tych działań i powinien zawierać:

- Określenie rezultatu jaki persona chce uzyskać w trakcie jego realizacji,
- Motywacje do wykorzystania scenariusza,
- Szczegółowy opis sekwencji działań realizowanych w trakcie scenariusza,
- Opis stanu systemu po wykonaniu scenariusza.
- Wymagania środowiska wykonawczego.

Scenariusze są typową techniką, znaną z inżynierii oprogramowania. Alternatywą do scenariuszy są przypadki użycia (ang. *Use Cases*).



Rys. 3.8. Przykład struktury, danych i sposobu ich prezentacji dla persony
Źródło: http://profs.info.uaic.ro/~stefan.negru/personas/

Planowanie zadań to faza UCA, w której tworzone są formalne modele zadań, jakie będzie wykonywał użytkownik w systemie. Do zobrazowania tych modeli używa się typowych technik: schematów blokowych (ang. *Block Diagrams*), diagramów aktywności (ang. *Activity Diagrams*) lub diagramów drzew zadań (ang. *Concur Task Trees, CTT*) [14].

Zadanie powinno posiadać cel (ang. *Goal*), listę działań (ang. *Action Model*), opis środowiska (ang. *Domain Model*) oraz związki pomiędzy tymi modelami. Modele zadań mają zwykle charakter hierarchiczny.

W trakcie modelowania zadań ważnym procesem jest wyszukanie potencjalnych błędów, które mogą pojawić się w trakcie realizacji każdego zadania. Zidentyfikowane błędy powinny być opisane i użyte do zaprojektowania procesów ich obsługi.

Zadania są klasyfikowane (i szeregowane) z wykorzystaniem różnych parametrów, takich jak:

- stopień ważności zadania (główne, wspomagające, …, dodatkowe);
- zależność od innych zadań (centralne, zależne);
- stopnia ważności (bardzo ważne, …, uzupełniające);
- częstotliwość realizacji;
- krytyczność (krytyczne czasowo, obciążające zasoby, obsługujące błędy);
- współdzielone (lub nie) pomiędzy różnymi personami.

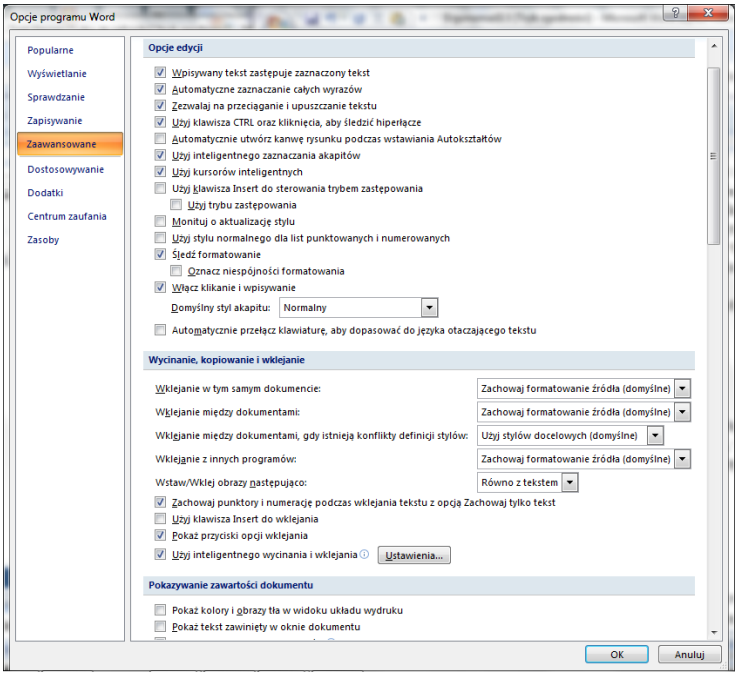
Klasyfikacja wspomaga szeregowanie zadań i umożliwia optymalizację działań projektowych i implementacyjnych. Przykładowo, zadania wykonywane często są bardzo istotne i w nich należy szukać wzrostu produktywności użytkowników (bardzo ważne dla ROI) poprzez dopracowanie interfejsu. Zadania rzadko wykonywane (np. konfiguracyjne– rys. 3.9) mogą mieć gorszy interfejs.

Architektura informacji to sposób grupowania treści występujących w oprogramowaniu. Dotyczy ona zawartości informacyjnej oprogramowania, począwszy od struktury bazy danych, a skończywszy na interfejsie. Prawidłowe pogrupowanie informacji zwiększa ergonomię pracy z aplikacją. O ile struktura bazy danych jest ukryta przed użytkownikiem, to interfejs powinien odwzorowywać jego zrozumienie istoty aplikacji i sposobu pracy z nią. Prawidłowa architektura informacji zwiększa efektywność pracy użytkownika, skupia jego uwagę na zawartości ważnej w danym momencie oraz przedstawia treści w sposób dla niego zrozumiały i intuicyjny. W związku z tym, to od użytkownika powinna pochodzić wiedza na temat architektury informacji, ma ona odwzorowywać jego sposób myślenia. Pogrupowanie danych może być:

- **dokładne** – alfabetyczne, geograficzne, chronologicznie (rys. 1.16);
- **kategoryczne** – tematyczne-zadaniowe, wg grup użytkowników, produktów, wydziałów czy ról;
- **hybrydowe** – łączące kategorie z dokładnym grupowaniem.

W wielu przypadkach grupowanie może być wielowymiarowe, tzn. dana informacja może należeć do wielu różnych grup. Do której konkretnie (a także, w jakiej kolejności, przy grupowaniu wielopoziomowych) powinna być przydzielona określa się w wyniku badań z użytkownikami.

Do architektury informacji należy także nazewnictwo (etykiety), czyli meta dane. Pojawiają się one potem w interfejsie. Ich dokładne określenie jest wstępem do projektowania interfejsu. Użytkownicy powinni je rozumieć i rozróżniać. W UCA to właśnie użytkownicy mają głos decydujący w określaniu słownictwa.



Rys. 3.9. Przykład przeciążonego możliwościami interfejsu
Źródło: opracowanie własne

Opracowanie nawigacji po informacji w oprogramowaniu to ostatnia faza procesu UCA. Nawigowanie pomiędzy informacjami umożliwia użytkownikom:

- poruszanie się pomiędzy informacjami i funkcjami oprogramowania,
- uwypuklenie kontekstu informacji,
- uwypuklenie stopnia ważności informacji,
- wskazanie co jest z czym powiązane,
- umożliwienie użytkownikom dotarcia do informacji, które są im nieznane.

Nawigacja po informacji dostarcza użytkownikowi:

- możliwości w zorientowaniu się, w jakim miejscu aplikacji jest,
- dostępu do pełnej gamy opcji nawigacyjnych w aplikacji,
- zrozumiałych mechanizmów nawigacyjnych,
- możliwości wyjścia z miejsca w aplikacji do miejsca mu znanemu,
- możliwości nawigacyjnych przewidujących jego potrzeby w danej chwili.

Model nawigacji stanowi podstawę do zaprojektowania układu sterującego oprogramowania.

3.3. Metody projektowania interfejsu

W modelach cyklicznych wytwarzania oprogramowania opracowuje się wiele kolejnych, bardziej dokładnych wersji interfejsu. W trakcie każdego cyklu wersje są poddawane ocenie w procesie testowania przez użytkownika (UT). Takie podejście powoduje generowanie interfejsów coraz to bardziej realistycznych i lepiej spełniających wymagania użytkowników. Ciąg powstających projektów interfejsów ilustruje rys. 3. 10.

Rys. 3.10. Przykład wielu wersji interfejsu
Źródło: http://profs.info.uaic.ro/~busaco/teach/courses/hci/

W trakcie projektowania interfejsu używa się następujących metod:

- symulacja interfejsu, metoda ***Czarnoksiężnik z Oz*** (ang. *Wizard of Oz*);
- prototypowanie;
- storyboards;
- wykorzystanie standardów (and. *Standards*) i zaleceń (ang. *Guidelines*);
- modelowanie przy pomocy formalnych języków opisu interfejsu (ang. *User Interface Description Language, UIDL*).

Stosowane są także inne metody i techniki typowe dla inżynierii oprogramowania (schematy blokowe, diagramy UMLa, itd.).

Metoda symulacji interfejsu, zwana ***Czarnoksiężnik z Oz***, polega na zorganizowaniu eksperymentu, w którym użytkownicy odgrywają role, udając realizację swoich zadań w jednym pomieszczeniu, a odpowiedzi oprogramowania generuje operator („Czarnoksiężnik"), będący w innym pomieszczeniu. Do komunikacji pomiędzy użytkownikami a operatorem służy prosty informatyczny system komunikacyjny. Użytkownicy nie wiedzą, że komunikują się z operatorem, a nie systemem informatycznym. Całość eksperymentu musi być oczywiście odpowiednio przygotowana, pod kątem technicznym (komputery i oprogramowanie komunikacyjne, sprzęt i oprogramowanie rejestrujące działania ludzi), organizacyjnym (odpowiedni ludzie) oraz merytorycznym (dobór zadań do wykonania, scenariusze działań, dane). Po symulacji, którą obserwuje zespół zbierania rezultatów, następuje obróbka zebranych danych i wyciągane są wnioski dotyczące dialogu człowiek-komputer. Sesji symulacyjnych może być wiele. Metodą symulacji interfejsu stosuje się zwykle w celu początkowego ustalenia wymagań w stosunku do interfejsu.

Metoda **prototypowania** polega na opracowaniu prototypu interfejsu projektowanego oprogramowania lub jego części. Prototyp jest oceniany przez użytkowników (proces UT z UCD). Zwykle ocena jest zorganizowania w postaci badań.

Pod względem realistyczności prototypy dzielą się na:

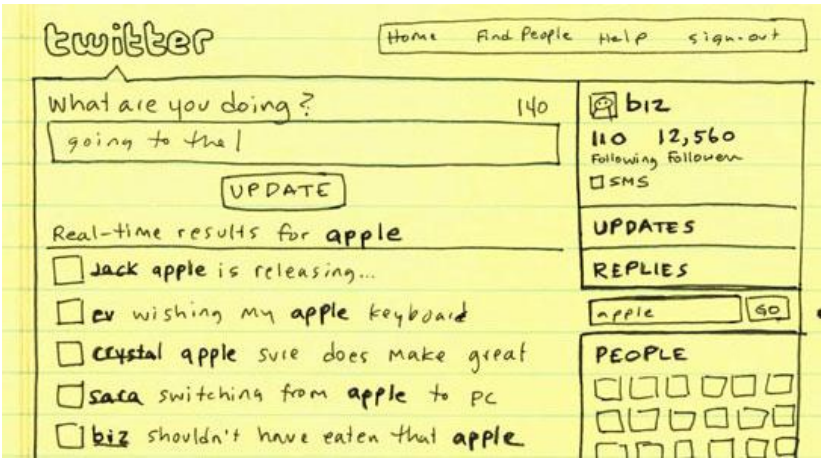
- małej dokładności (ang. *Low-Fidelity Prototype*),
- wysokiej dokładności (ang. *High-Fidelity Prototype*).

Prototypy różnią się techniką wykonania. W zależności od dokładności i techniki wykonania prototypy dzielą się na (uporządkowanie w kolejności wzrostu dokładności, realistyczności):

- szkice (ang. *Sketches*),
- makiety (ang. *Wireframes*),
- atrapy (ang. *Mockups*),
- działające (dynamiczne) prototypy.

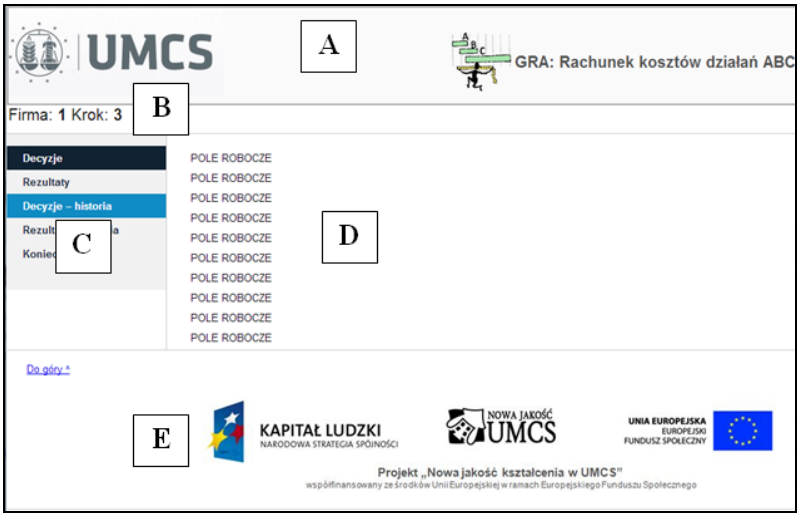
Szkice wykonuje się ręcznie na papierze lub tablicy w trakcie sesji projektowania, realizowanej zwykle z udziałem użytkowników – rys. 3.11. Zestaw takich szkiców służyć może do ręcznej symulacji działania interfejsu. Użytkownik może wskazywać na elementy interfejsu, a projektanci mogą symulować odpowiedź systemu wskazując użytkownikowi kolejne szkice.

Szkice używane są na początkowych etapach projektowania oprogramowania, głównie w celu wypracowania wielu możliwych rozwiązań i wyboru jednego z nich. Charakteryzują się prostotą, niskim kosztem realizacji oraz możliwością tworzenia dowolnych rozwiązań, w tym takich, dla których nie ma wspomagających narzędzi komputerowych.



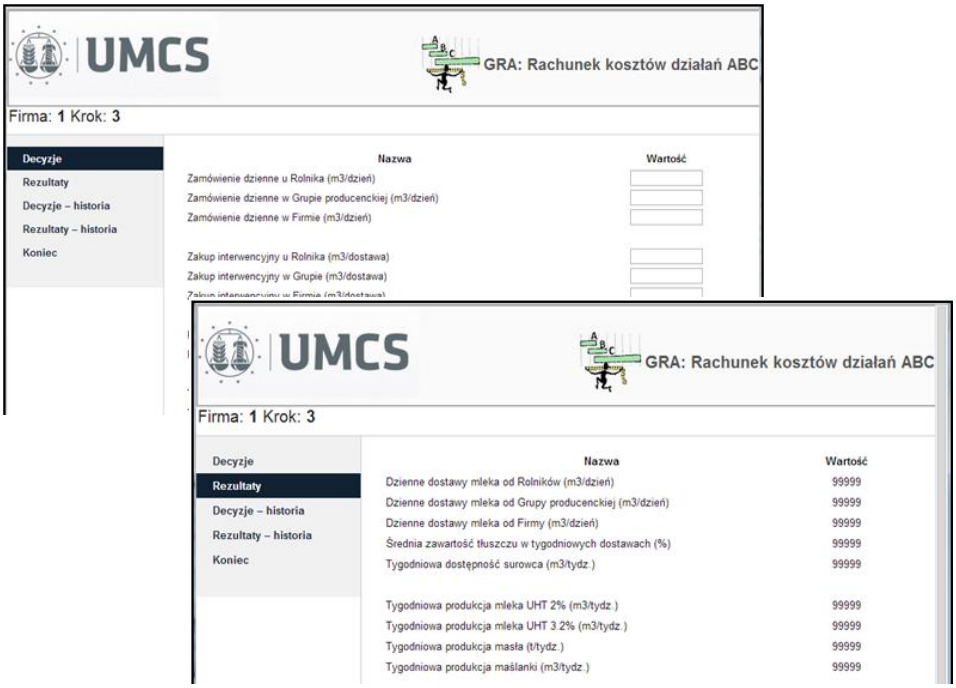
Rys. 3.11. Jeden z pierwszych ręcznych szkiców strony Twitter
Źródło: http://wphow.org/case-study-the-meaning-of-success-on-twitter/

Makiety to szkice-przewodniki, sugerujące strukturę ekranów (rys. 3.12) i nawigacji pomiędzy nimi. Stosuje się je w celu ujednolicenia interfejsu na wczesnych etapach projektowania. Są często graficznymi elementami standardów i zaleceń tworzenia interfejsów.



Rys. 3.12. Przykład makiety aplikacji webowej z oznaczonymi obszarami
Źródło: opracowanie własne

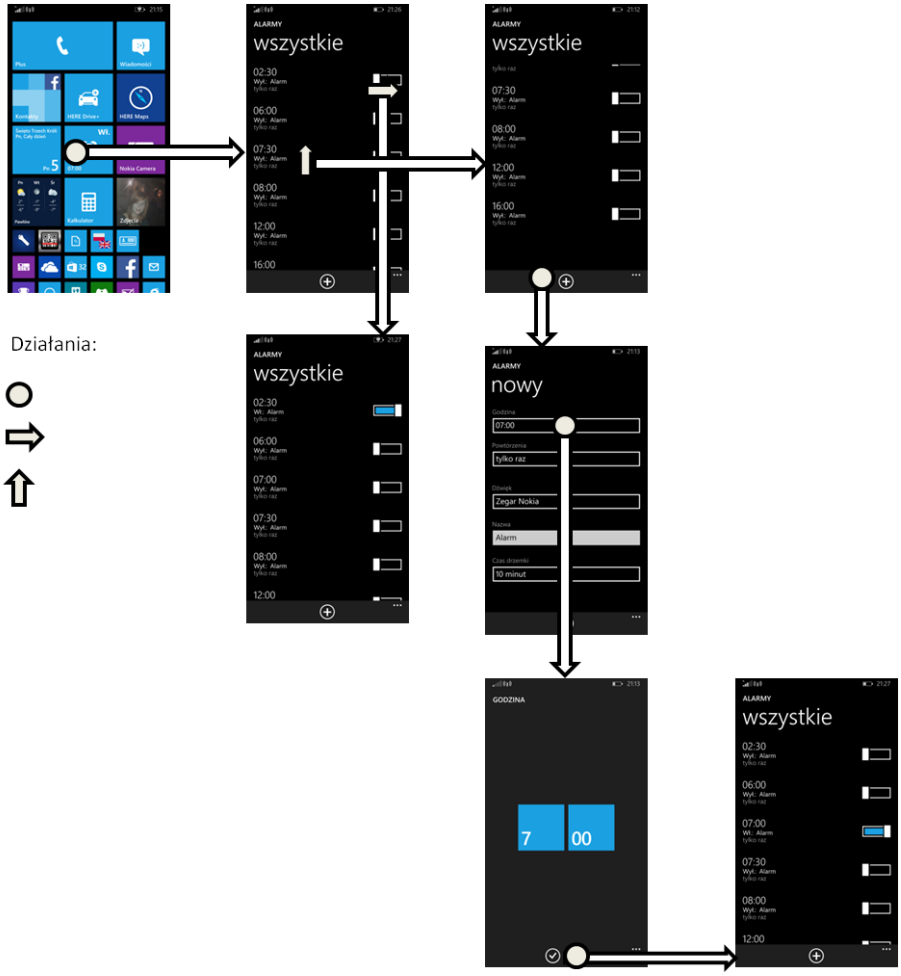
Atrapy (**mockupy**) to kompletne komputerowe modele interfejsu, które często są interaktywne rys. 3.13. Służą do projektowania, oceny i demonstrowania rozwiązań, a nawet do realizacji szkoleń użytkowników. Wykonuje się je przy pomocy oprogramowania wspomagającego. Może to być uniwersalne oprogramowanie, służące do prezentacji – np. Visio czy też PowerPoint, które to posiadają elementy aktywnej nawigacji po prezentacji. Stosowane są też specjalistyczne programy do tworzenia modeli interfejsu i symulowania pracy z nim.



Rys. 3.13. Przykład atrapy aplikacji webowej z działającym układem menu
Źródło: opracowanie własne

Działające prototypy to programy (lub dokumenty html), które wiernie przedstawiają projektowany interfejs. Powstają one z wykorzystaniem specjalnych narzędzi (środowisk do programowania wizualnego, edytorów kodu html itp.) i są w pełni działającymi aplikacjami, pozbawionymi tylko pewnych elementów technicznych, niewidocznych zwykle dla użytkownika. Mogą nie posiadać procedur uwierzytelniania, obsługi błędów czy też implementacji skomplikowanych algorytmów czy zapytań.

Storyboards są to rysunki zawierające zaprojektowane ekrany i związki pomiędzy nimi (w postaci strzałek od źródła do ekranu-rezultatu). Nazwa pochodzi od serii obrazków, w których reżyser w scenariuszu przedstawia oddzielne sceny przyszłego filmu. Są one ilustracją scenariuszy pracy z oprogramowaniem. Mogą zawierać informacje o zdarzeniach, które są przyczynami przejść. Z teoretycznego punktu widzenia są diagramami stanu interfejsu oprogramowania, w których stany systemu są klatkami (ekranami). Przykładowy storyboard przedstawia rys. 3.14.



Rys. 3.14. Storyboard dla aplikacji mobilnej
Źródło: opracowanie własne

Wykorzystanie **standardów** i **zaleceń** wspomaga prawidłowe projektowanie interfejsu od samego początku, pozwalając zaoszczędzić czas i środki na tworzenie i testowanie prototypów. Poza tym standardy i zalecenia są opracowywane (optymalizują interfejs) dla konkretnych środowisk i zawierają w sobie doświadczenie i wiedzę zdobytą w wielu projektach. Stosowanie ich ma jeszcze jedno bardzo ważne znaczenie: użytkownicy dostają ujednolicony interfejs w wielu różnych programach. Są zatem do niego już przyzwyczajeni, co ogranicza zakres szkoleń i zwiększa efektywność pracy z oprogramowaniem.

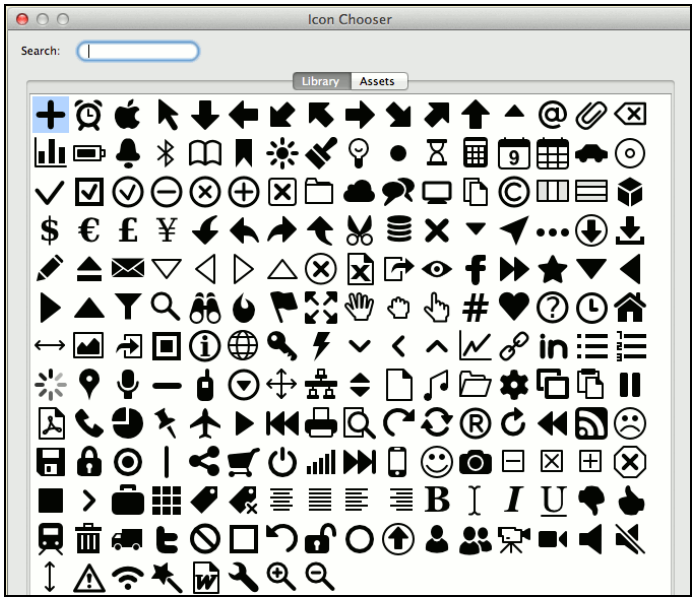
Do najbardziej znanych standardów, związanych z ergonomią interfejsów komputerowych należą:

- ISO 9241 – zestaw standardów pod wspólną nazwą ***Ergonomia interakcji człowieka i systemu***, który składa się z następujących części:
    - 110 – Zasady dialogu,
    - 210 – Projektowanie ukierunkowane na człowieka w przypadku systemów interaktywnych.
- W3C Standards – zestaw standardów Fundacji W3C, zawierający między innymi standardy (pełny zestaw dostępny na: www.w3.org/standards):
    - Accessible Rich Internet Applications (WAI-ARIA) 1.0,
    - Web Content Accessibility Guidelines (WCAG) 2.0,
    - Core Accessibility API Mappings 1.1,
    - User Agent Accessibility Guidelines (UAAG) 2.0,
    - Media Accessibility User Requirements,
    - Requirements for IndieUI: Events 1.0 and IndieUI: User Context 1.0,
    - Role Attribute 1.0.

Poza standardami, opracowywane i udostępniane są zalecenia producentów systemów operacyjnych czy sprzętu, a dotyczące zasad projektowania interfejsów. Jako przykłady można wskazać następujące:

- User Experience Design Guidelines for Windows Phone,
- iOS Human Interface Guidelines,
- Samsung Smart TV UX Guideline,
- Android Design Principles,
- IBM Application and Web design,
- GNOME Human Interface Guidelines,
- UX Design Standards for vSphere Web Client,
- BBS Global Experience Language

i wiele innych, w tym np. zbiory gotowych, standardowych ikon do zastosowania w aplikacjach (rys. 3.15).



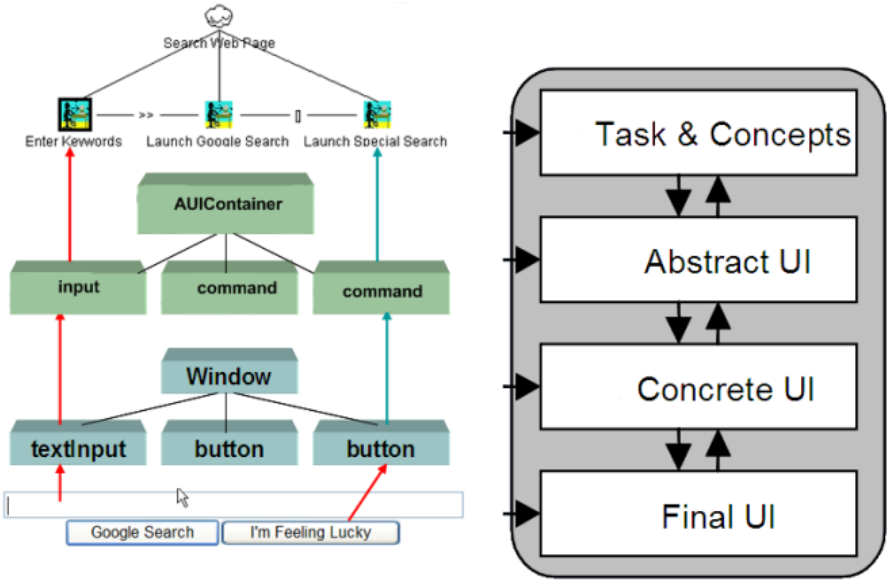
Rys. 3.15. Zbiór standardowych ikon do wykorzystania w projekcie interfejsu (przykład)
Źródło: http://wireframesketcher.com

Opracowano cały szereg **formalnych języków modelowania interfejsu**, odwzorowujących zarówno jego struktury, jak i dynamikę (czyli związki pomiędzy strukturami-ekranami). Języki te zwykle wykorzystywane są w metodzie projektowania sterowanej modelami (ang. *Model-Based UI Design*). W metodzie tej, na podstawie analizy zadań i koncepcji aplikacji, tworzy się kolejne modele [14]:

- abstrakcyjny model interfejsu (ang. *Abstract UI Model*),
- konkretny model (ang. *Concrete UI Model*),
- końcowy interfejs (ang. *Final UI*).

Modele te są w większości przypadków tworzone na podstawie modeli niskiego poziomu z wykorzystaniem różnych technik transformacji i dodatkowych danych, wspomagających transformację między poziomową.

Związki pomiędzy modelami, a realnymi elementami projektowymi przedstawia rys. 3.16.



Rys. 3.16. Modele interfejsu użytkownika i ich implementacja
Źródło: [14]

Do opisu interfejsu użytkownika wykorzystywane są między innymi następujące formalne języki [14]:

- UIML – User-Interface Markup Language,
- UsiXML – USer Interface eXtensible Markup Language,
- XAML – eXtensible Application Markup Language,
- XUL – eXtensible User-interface Language,

W praktyce projektowania interfejsów podejście sterowane modelami jest bardzo rzadko stosowane.

3.4. Narzędzia wspomagające projektowanie interfejsu oprogramowania

W rozdziale tym przedstawione są tylko nieliczne, wybrane narzędzia komputerowego wspomagania projektowania interfejsu (niekiedy w bardzo wielkim skrócie, sprowadzonym tylko do nazwy). Jest to uwarunkowane dużą ich liczbą i celami tej książki. Opisane zostały dokładniej (jak to było tylko możliwe) tylko te najbardziej znane lub najczęściej stosowane w danej metodzie. Opis jedynie przybliża możliwości oprogramowania wspomagającego pracę projektanta ergonomicznych interfejsów.

Sortowanie kart

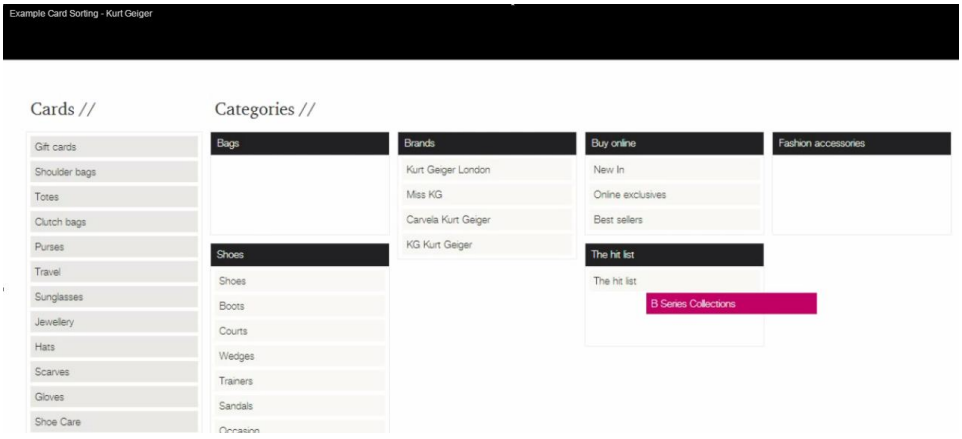
Bardzo duża liczba narzędzi wspomagających stosowanie techniki sortowania kart umożliwia współdziałanie uczestników przy pomocy Internetu. Przykładowymi aplikacjami są:

- User Conversion card sorting (http://www.userconversion.com) – rys. 3.16,
- OptimalSort (http://www.optimalworkshop.com/optimalsort.htm),
- WebSort (http://websort.net/),
- ConceptCodify (http://conceptcodify.com/),
- UsabilitiTest Card-Sorting (http://www.usabilitest.com/CardSorting),
- UserZoom (http://www.userzoom.com/card-sorting/).

Aplikacje internetowe umożliwiają:

- przygotowanie badań metodą sortowania kart,
- zaproszenie i skolekcjonowanie uczestników,
- zebranie danych,
- analizę i wizualizację.

Dostępne są także narzędzia komputerowe w postaci programów desktopowych (np. xSort – http://www.xsortapp.com/ lub UXSort – https://sites.google.com/a/ uxsort.com/uxsort/), serwerów www (np. WebCAT – http://zing.ncsl.nist.gov/ WebTools/WebCAT/overview.html) oraz wtyczek do przeglądarek (np. uzCardSort – Uzilla.mozdev.org/cardsort.html).

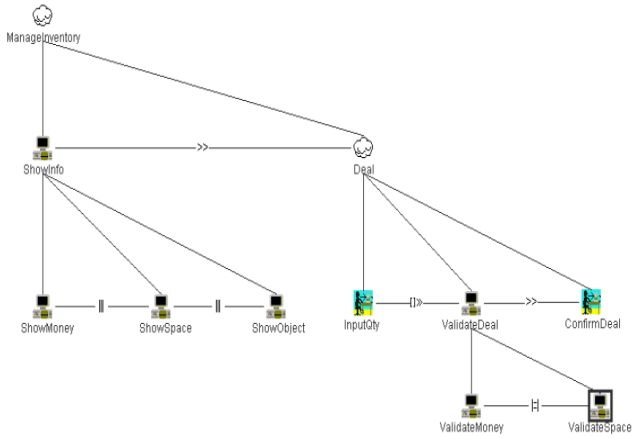


Rys. 3.16. User Conversion card sorting – przykład użycia narzędzia
Źródło: http://www.userconversion.com/Info/Test/card-sorting

Planowanie zadań

Planowanie zadań polega na budowie modelu zadań. Wspomaga go między innymi oprogramowanie wspomagające technikę budowy diagramów drzew zadań (ang. *Concur Task Trees, CTT*). Oprogramowanie to wspomaga budowę hierarchicznych struktur, odwzorowujących strukturę zadań, z możliwością odwzorowania procesów realizowanych równolegle. Związki pomiędzy zadaniami są modelowane przy pomocy różnych operatorów, posiadających specyficzne oznaczenia (np. [], |||, |=|, >> itp.) i określających różne znaczenie, począwszy od wyboru, poprzez synchronizację (w różnych trybach), a skończywszy na takich jak dostęp sekwencyjny.

Oprogramowanie wspomagające o nazwie CCTE (ang. *Concur Task Tree Environment*), dostępne na stronie http://giove.isti.cnr.it/ctte.html, umożliwia tworzenie diagramów drzew zadań w trybie wizualnym (rys. 3.17).



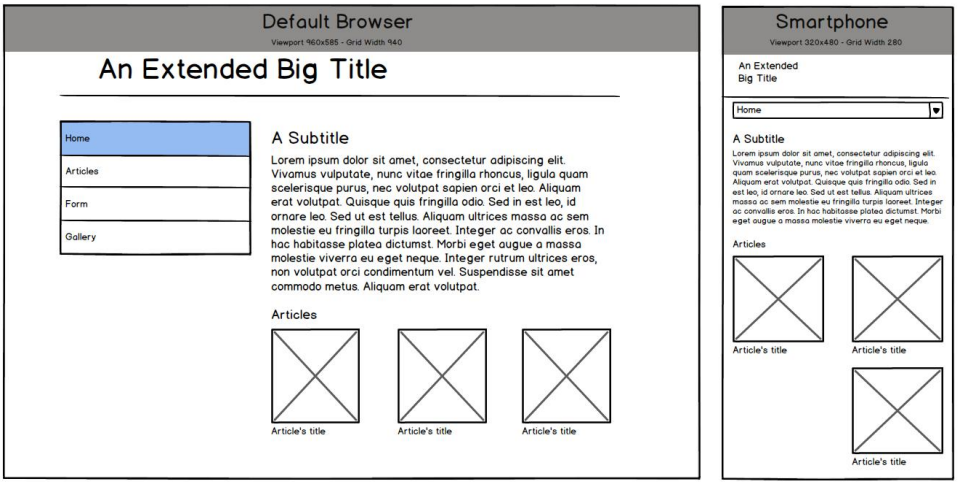
Rys. 3.17. Przykład diagramu drzewa zadań
Źródło: http://giove.isti.cnr.it/ctte.html

Tworzenie mockupów

Wiele różnych programów umożliwia tworzenie atrap, czyli mockupów interfejsu oprogramowania. Do najbardziej znanych można wskazać następujące systemy:

- Balsamiq Mockups (www.balsamiq.com) – rys. 3.18,
- HotGloo (www.hotgloo.com),
- MockFlow (mockflow.com),
- Mocking Bird (gomockingbird.com/mockingbird),
- Moqups (moqups.com),
- Proto.io (http://proto.io/),

a także ogólne, i dość uniwersalne programy do tworzenia diagramów i schematów, takie jak MS Visio.



(a) (b)

Rys. 3.18. Przykład mockaupu zrealizowanego w systemie Balsamiq równolegle dla interfejsu webowego (a) i mobilnego (b)
Źródło: [19]

3.5. Podsumowanie

Ogólny schemat projektowania interfejsu może wykorzystywać klasyczny model kaskadowy, ale znacznie efektywniej jest korzystać z modeli cyklicznych, takich jak iteracyjny, przyrostowy czy spiralny. Ze względu na czas i koszty stosuje się dwuetapowe projektowanie z wykorzystaniem prototypów papierowych (faza pierwsza) i wykonywalnych (druga). W trakcie projektowania powinny być brane cechy ergonomicznego interfejsu, takie jak: przystępność, przewidywalność, estetyka, dostępność, przejrzystość, konfigurowalność, spójność, kontrolowalność, efektywność i wydajność, standaryzacja, tolerancyjność na błędy i intuicyjność. By uzyskać tą ostatnią cechę, a także by z przyspieszyć prace nad konstrukcją interfejsu, stosuje sie projektowanie ukierunkowane na użytkownika. Proces ten jest objęty standardem. Podejście zwinne jest podstawą metody Lean UX.

Analiza potrzeb użytkownika jest procesem poznawania i ewidencjonowania potrzeb użytkownika. Wykorzystuje się w niej bardzo różnorodne metody pozyskania informacji od użytkownika. Do specyficznych dla UCA należy technika JAD i sortowania kart. W trakcie UCA stosuje się inne

specyficzne pojęcia, a mianowicie: profile i persony oraz architektura informacji. Architektura informacji to metadane, słownictwo i taksonomia informacji. Projekt architektury informacji zwiększa jakość interfejsu poprzez wzrost intuicyjności informacji.

Metody projektowania wykorzystują specyficzne techniki, takie jak: symulacja interfejsu, prototypowanie, storyboards, wykorzystywanie standardów i zaleceń oraz formalnych języków opisu interfejsu. Metoda symulacji interfejsu, zwana jest metodą Czarnoksiężnik z Oz. Prototypy mogą być małej oraz dużej dokładności i są szkicami, makietami, atrapami lub prototypami działającymi dynamicznie. Istnieje wiele standardów i zaleceń dotyczących poprawnego projektowania interfejsu. Część z nich jest uniwersalna, a część dotyczy interfejsów określonego typu oprogramowania (np. webowego, mobilnego czy wbudowanego).

Narzędzia wspomagające projektowanie skracają czas projektowania i czynią prototyp bardziej dokładnym. Wspomagają one realizacje poszczególnych metod i technik projektowania interfejsu.

4. Ocena jakości interfejsów oprogramowania

Ocena jakości interfejsu to trzeci z procesów UCD, zwany testowaniem przez użytkownika (UT). Proces ten dopełnia spiralny model projektowania interfejsu użytkownika we współczesnych aplikacjach. Nie zawsze jednak angażuje użytkownika – często bowiem go po prostu nie ma. Przykładem jest wytwarzanie oprogramowania masowego, dla ogółu, tj. bez konkretnego użytkownika.

4.1. Klasyfikacja metod oceny jakości interfejsu

O jakość interfejsu jego twórcy dbają w trakcie jego projektowania. Niestety, realną jakość rezultatu można sprawdzić jedynie po tym jak interfejs jest zaimplementowany.

Jakość interfejsu oprogramowania może być oceniana różnymi metodami, które można podzielić na metody [2]:

- **Automatyczne** (ang. *Automated*), tj. takie, w których procedurę oceny wykonują narzędzia komputerowe w całości lub w znacznej części.
- **Manualne** (ang. *Manual*), czyli wykonywane ręcznie przez człowieka. Faktem jest, że metody manualne mogą być wspomagane komputerowo, ale główna ich część, związana z oceną (pozyskaniem wiedzy), wykonywana jest przez człowieka – osobę oceniającą.

Metody automatycznej oceny mogą być stosowane wszędzie tam, gdzie dostępne są wzorce (ścisłe i poddające się algorytmizacji standardy), z którymi oceniane oprogramowanie może być porównane. Takie standardy i zalecenia są dość dobrze opracowane dla systemów desktopowych i webowych. Gorzej ze standardami, i w konsekwencji z oprogramowaniem automatycznie oceniającym, w obszarze aplikacji mobilnych [2]. Dobrym przykładem metod automatycznych są walidatory stron internetowych udostępniane przez Konsorcjum W3C na stronie internetowej: http://www.w3.org. Metody manualne, w zależności pochodzenia od osób oceniających, można podzielić na metody z i bez udziału użytkownika.

Inną klasyfikacją metod oceny jakości interfejsu oprogramowania jest ich podział na typy według sposobu ich realizacji. Doprowadza to do uzyskania następujących typów metod [2]:

- **Testowanie** (ang. *Testing*) – wykonywanie interakcji uczestników badania z interfejsem oprogramowania według określonych scenariuszy, obserwacja (w tym także pomiary, np. szybkości) i formułowanie wniosków na podstawie zebranych danych. Pewną odmianą tej metody jest zbieranie danych (zbieranie opinii) w trakcie normalnej eksploatacji oprogramowania. Rejestracja działań użytkowników może być użyta do wypracowywania poprawy zastosowanych rozwiązań.

- **Inspekcja** (ang. *Inspection*) – przegląd interfejsu z wykorzystaniem list kontrolnych, różnych kryteriów lub analizy heurystycznej, dokonywany przez ekspertów, w celu identyfikacji potencjalnych problemów z interfejsem oprogramowania;
- **Wywiad** (ang. *Survey*) – pozyskanie informacji o jakości (lub o problemach z nią) od użytkowników (po określonym czasie ich pracy z oprogramowaniem) przy pomocy wywiadów, ankiet, kwestionariuszy i podobnych technik. Metoda możliwa do zastosowania także w trakcie eksploatacji oprogramowania.
- **Modelowanie analityczne** (ang. *Analytical Modeling*) – tworzenie prognoz jakości interfejsu poprzez budowę i wykorzystanie modeli interakcji interfejsu z użytkownikiem. Metoda bardzo pracochłonna i w związku z tym dość rzadko stosowana.
- **Symulacja** (ang. *Simulation*) – użycie modeli komputerowych interakcji interfejsu oraz użytkownika do symulowania (naśladowania) typowych interakcji użytkownika z interfejsem, zbieranie danych z eksperymentów symulacyjnych i raportowanie ich wyników. Metoda wymaga specjalistycznego oprogramowania modelująco-symulacyjnego.

Do oceny jakości interfejsów mogą być stosowane różne metryki. Stosowanie konkretnej metryki daje bowiem możliwość porównywania wyników. Istnieje wiele systemów klasyfikacji metryk oceny jakości interfejsów oprogramowania. Jeden z nich dzieli metryki na:

- **Metryki wydajnościowe** (ang. *Performance Metrics*):
  - Sukces realizacji zadania (ang. *Task Success*) – mierzy, jak skutecznie użytkownicy są w stanie ukończyć dany zestaw zadań.
  - Czas realizacji zadania (ang. *Time on Task*) – mierzy, ile czasu jest potrzebne do wykonania zadania.
  - Stopa błędów (ang. *Errors*) – określa liczbę błędów popełnionych w trakcie realizacji zadania.
  - Przyswajalność (ang. *Learnability*) – mierzy zmiany w produktywności użytkowników w trakcie pracy (zaznajamiania się) z oprogramowaniem.
- **Metryki bazujące na problemach** (ang. *Issue-based Metrics*):
  - Częstotliwość występowania problemów (ang. *Frequency of Unique Issues*) – liczba unikalnych problemów jaka wystąpiła w trakcie sesji badawczej.
  - Częstotliwość występowania problemów na jednego użytkownika (ang. *Frequency of Unique Issues per Participant*) – liczba unikalnych problemów jaka wystąpiła w trakcie sesji badawczej przypadająca na jednego uczestnika sesji.
  - Procent uczestników doświadczających błędu (ang. *Frequency of Participants*) – procent uczestników sesji oceny jakości, którzy znaleźli dany błąd.

- Kategoryzacja błędów (ang. *Issues by Category/Task*) – pomiar liczby błędów w zagregowanych grupach/kategoriach.

- **Metryki bazujące na ocenach użytkowników** (ang. *Self-reported Metrics*):
  - Ocena po zadaniu (ang *Post-task Ratings*) – ocena realizowana po wykonaniu określonego zadania przez użytkownika z wykorzystaniem określonej skali; równocześnie mogą być oceniane różne parametry jakościowe oprogramowania.
  - Ocena po sesji badawczej (ang. *Post-session Ratings*) – uogólniona ocena oprogramowania, realizowana po sesji badań, czyli po serii zadań wykonanych przez użytkownika; równocześnie mogą być oceniane różne aspekty użytkowania oprogramowania.
  - Ocena specyficznego aspektu oprogramowania (ang. *Assessing Specific Attribute*) – ocena przez użytkownika specyficznej cechy oprogramowania (np. dostosowanie do osób niesprawnych ruchowo lub niedowidzących) lub też specyficznego elementu interfejsu oprogramowania (np. funkcja wyszukiwania, pomocy użytkownikowi czy też wspomagania sieci socjalnych).
- **Metryki behawioralne i fizjologiczne** (ang. *Behavioural and Physiological Metrics*).

Metody oceny jakości interfejsów z udziałem użytkowników najczęściej stosowane to:

- Testowanie interfejsu przez użytkowników lub przyszłych użytkowników.
- Test Kruga.
- Okulografia (ang. *Eyetracking*).
- Śledzenie działań użytkownika w interfejsie (ang. *Clicktracking*).
- Zbieranie opinii użytkowników oprogramowania (ang. *User Feedback*).

Do najczęściej stosowanych metod oceny jakości interfejsów bez udziału użytkowników zalicza się metody:

- Analiza ekspercka (ang. *Expert Review*).
- Uproszczona wędrówka poznawcza (ang. *Cognitive Walkthrough*).
- Rozwinięta wędrówka poznawcza (ang. *Pluralistic Walkthrough*).
- Ocena heurystyczna (ang. *Heuristic Evaluation*).
- Inspekcja standardów (ang. *Guidelines Inspection*).

W trakcie badań interfejsu stosowane są metryki oceny jakościowej i ilościowej. Metryki jakościowe zwykle są potem mapowane na skale ilościowe, typu: całkowicie spełnia, spełnia,…, całkowicie nie spełnia. Podczas mapowania stosowana jest zwykle 5 poziomowa skala. Możliwe jest mapowanie na skalę procentową: od 0% (całkowity brak) po 100% (całkowite spełnienie). Skala procentowa stosowana jest zwykle przy uśrednianiu odpowiedzi z wielu ankiet.

4.2. Ocena z udziałem użytkownika

Ocena z udziałem użytkownika jest eksperymentem badawczym, w którym w aktywny sposób angażuje się obecnych lub potencjalnych użytkowników oprogramowania. Do najważniejszych jego etapów należą:

- przygotowanie eksperymentu (tj. plan, scenariusze, sprzęt, oprogramowanie, miejsce itd.);
- dobór i pozyskanie grupy badawczej;
- przeprowadzenie eksperymentu z grupą badawczą;
- analiza i uogólnienie wyników eksperymentu;
- wyciągnięcie wniosków i wypracowanie rekomendacji.

Testowanie interfejsu przez użytkowników jest najczęściej stosowaną metodą oceny oprogramowania. Odpowiednio dobrana grupa użytkowników pracuje z oprogramowaniem wykonując określone scenariusze lub też bez tych scenariuszy. W trakcie lub po eksperymencie zbierane są jego rezultaty. Zwykle stosuje się tu techniki mieszane: obserwacja w trakcie działań, rejestracja działań i ich parametrów oraz ankietowanie po eksperymencie. W trakcie eksperymentu dokonuje się pomiaru lub oceny wartości cech, takich jak czas realizacji zadań, procent błędów, satysfakcja użytkownika itd. Ich wartości wyznacza się przy pomocy metryk opisanych w rozdz. 4.1. Testowanie w pierwszej kolejności powinno dotyczyć typowych sytuacji (najczęściej wykonywanych przez użytkownika zadań).

Do oceny jakości interfejsu można użyć wielu wskaźników, wartości których są wyznaczane w rezultacie testowania przez użytkowników. Mogą to być:

- czas ukończenia zadań;
- procent pomyślnie ukończonych zadań;
- procent zadań poprawnie ukończonych w przyjętym limicie czasowym;
- czas poświęcony na obsługę błędów;
- procentowa liczba błędów w ogólnej liczbie wykonanych przez użytkownika operacji;
- liczba używanych funkcji i poleceń;
- częstotliwość używania pomocy;
- czas spędzony na przeglądaniu dostępnej dokumentacji w celu wyszukania informacji dotyczącej sposobu realizacji różnych operacji lub obsługi pojawiających się błędów;
- liczba powtórzeń lub nieudanych prób użycia funkcji;
- liczba wprowadzeń w błąd użytkownika;
- liczba poprawnie i błędnie wywoływanych funkcji przez użytkownika w określonym czasie;
- liczba komend dostępnych, ale nie używanych nigdy przez użytkowników do realizacji zadań;

- liczba sytuacji, w których użytkownicy musieli poszukiwać alternatywnych sposobów rozwiązania danego problemu;
- liczba sytuacji rozproszenia uwagi użytkownika;
- liczba przypadków utraty kontroli nad systemem;
- liczba zgłoszeń frustracji użytkownika.

Test Kruga został zaproponowany do testowania interfejsów webowych. Można go jednak dość łatwo zaadaptować do testowania jakości dowolnego interfejsu.

Test Kruga polega na przedstawieniu użytkownikowi prototypu lub gotowej strony internetowej, na której badana osoba ma szybko wskazać [9]:

- Co to jest za witryna (identyfikator witryny)?
- Na jakiej jestem stronie (nazwa strony)?
- Jakie są główne kategorie?
- Jakie mam opcje do wyboru na tym poziomie struktury?
- Gdzie znajduję się w odniesieniu do całej struktury?
- W jaki sposób mogę czegoś poszukać?

Statystyka odpowiedzi (w układzie poprawnie, błędnie i nie wskazano) świadczy o jakości interfejsu, a także o poprawności każdego z jego badanych elementów.

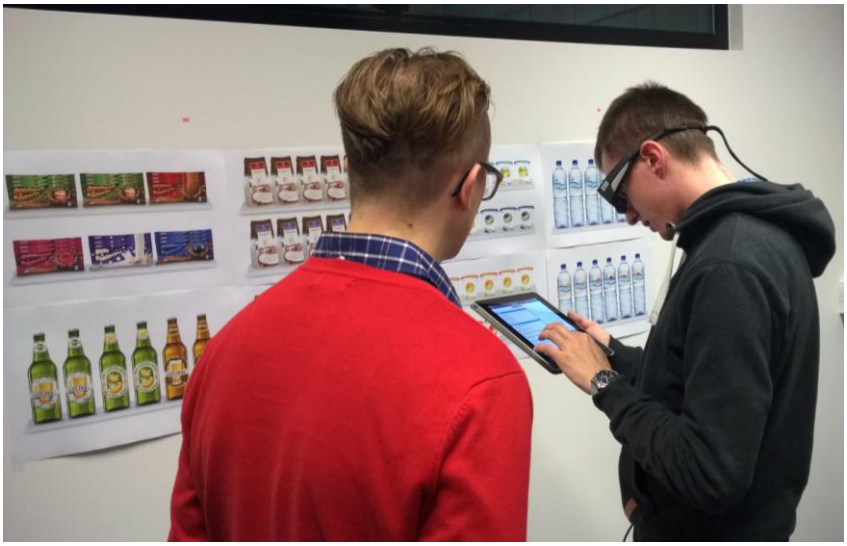
Przy pomocy testu Kruga i statystycznej obróbki rezultatów można wskazywać na to, która propozycja interfejsu dla której grupy użytkowników jest najbardziej odpowiednia.

Okulografia, czyli eyetracking, to metoda analizy działań użytkownika w trakcie pracy z oprogramowaniem poprzez śledzenie ruchów gałek ocznych człowieka w celu rejestracji punktów i czasów skupienia wzroku. Punkty te są nanoszone na tło, czyli obraz, na który patrzy użytkownik i który to jest rejestrowany równolegle ze śledzeniem gałek ocznych. W chwili obecnej w okulografii stosowane są metody bezinwazyjne polegające na oświetleniu gałki ocznej promieniami podczerwonymi (niewidocznymi dla oka) oraz rejestracja wektora (linii) patrzenia. Pozwala to wyznaczyć punkt, na którym badany koncentruje wzrok (nawet bezwiednie). Urządzenia do śledzenia (zwane okulografami lub eyetrackerami) mogą być przenośne (w postaci okularów – rys. 4.1) lub stacjonarne – monitor LCD z odpowiednim oprzyrządowaniem.

Wyniki badań okulograficznych przedstawia się w różnych formach, które ułatwiają analizę rezultatów. Do najczęściej stosowanych należą:

- mapy fiksacji (tj. koncentracji wzroku) – rys. 4.2;
- mapy cieplne – rys. 4.3;
- filmy z punktem skupienia wzroku (dla badań dynamicznych, np. działań graczy w grach komputerowych).

Dane uzyskane z badań można także poddawać obróbce bardziej wyrafinowanej niźli prosta wizualizacja. Pozwala to na przykład identyfikować obszary zainteresowania użytkownika.



Rys. 4.1. Badania interfejsu aplikacji mobilnej eyetrackerem przenośnym
Źródło: opracowanie własne (Laboratorium Analizy Ruchu i Ergonomii Interfejsów, Politechnika Lubelska)

Rys. 4.2. Mapa fiksacji wzroku
Źródło: http://www.flickr.com/

Rys. 4.3. Mapy cieplne różnych interfejsów
Źródło: http://eyetracking.me/

Zbieranie opinii użytkowników oprogramowania dotyczyć może eksploatowanych wersji oprogramowania lub też tzw. wersji beta, przekazywanej do testowania różnym grupom przyszłych użytkowników. Opinie wyrażane przez użytkowników są pozyskiwane automatycznie (np. średni czas realizacji określonego zadania) lub też metodą wywiadu. Wywiady mogą być przeprowadzane tradycyjną-papierową metodą (ang. *Paper-based Survey*) lub też elektronicznie, przy pomocy ankiet, e-maili lub też forów dyskusyjnych. Podstawowymi zaletami metody są niskie koszty, przy dużej liczbie potencjalnych respondentów, i możliwość zorganizowania badań jako procesu ciągłego. Podstawowe wady metody to nieznana i niereprezentatywna próba badawcza, niekonkretne odpowiedzi (szczególnie przy pytaniach otwartych), mylne sugestie oraz pojawiające się sprzeczności.

Dobór grupy badawczej (i jej liczebność) jest bardzo ważnym czynnikiem, jaki powinien zostać wzięty pod uwagę w trakcie badań z udziałem użytkowników. Dobór grupy badawczej powinien uwzględniać oprócz celu badań (np. badanie stopnia dostępności interfejsu aplikacji webowej dla najbardziej szerokiego grona użytkowników) również:

- doświadczenie użytkowników w pracy z danym typem interfejsu (doświadczenie manualne);
- doświadczenie użytkowników w pracy z poprzednimi wersjami oprogramowania lub podobnym (doświadczenie merytoryczne);
- zaburzenie widzenia (brak, praca w okularach, zaburzenia postrzegania barw, itp.);

- wiek i poziom wykształcenia;
- pochodzenie społeczne i kulturowe.

W zależności od celu badań dobiera się grupę osób o specyficznym wektorze cech.

Liczebność grupy badawczej przy badaniach oceniających jakość interfejsu nie musi być duża. Już przy 5 osobach wykrywane jest 80% problemów (ale przy iteracyjnie wykonywanych badaniach). Zwiększenie tej liczby do 20 pozwala wykryć praktycznie wszystkie problemy.

W trakcie badań użytkownicy poddawani są stresowi, który może doprowadzić do zafałszowania wyników. Sytuacja stresowa dla uczestników badań wynika z następujących elementów:

- tremy,
- odczucia bycia sprawdzanym czy testowanym,
- odczucia bycia porównywanym z innymi,
- współzawodnictwa z innymi,
- bycia obserwowanym i ocenianym przez mądrzejszych badaczy.

Powyższe negatywne czynniki powinny być w trakcie badań niwelowane poprzez stworzenie komfortowej sytuacji w trakcie badań, zapewnienie prywatności i dyskrecji obserwacji, uprzedniego przeszkolenia uczestników badań i wytłumaczenia im ich istoty.

4.3. Metody oceny bez udziału użytkownika

Ocena jakości interfejsu bez udziału użytkownika realizowana jest albo w sposób automatyczny przez projektantów (twórców) oprogramowania lub też z wykorzystaniem ekspertów. Badanie interfejsu z udziałem ekspertów nazywa się **analizą ekspercką**. Wymaga ono zaangażowania grona ekspertów z obszaru jakości interfejsów i odpowiedniego przygotowania. Może być przeprowadzane w różny sposób.

Uproszczona wędrówka poznawcza jest jedną z metod eksperckich. W jej trakcie ekspert realizuje scenariusz wykorzystania oprogramowania. Głównym celem tej metody jest ocena procesu uczenia się oraz poznanie płynności procesu realizacji zadań z użyciem oprogramowania, a nie jakości poszczególnych ekranów interfejsu. Podczas realizacji scenariusza ekspert stara się znaleźć odpowiedzi na następujące pytania:

- Czy użytkownik będzie wiedział, w jaki sposób osiągnąć zamierzony efekt?
- Czy użytkownik zauważy elementy interfejsu pomocne w realizacji zadania?
- Czy użytkownik będzie potrafił powiązać elementy interfejsu z akcjami, które musi wykonać?
- Czy użytkownik będzie informowany o stanie realizacji zadania (czy otrzyma odpowiednią informację zwrotną od systemu)?

Odpowiedzi na te pytania są oceniane przez eksperta w skali od 1 do 5, gdzie 1 oznacza bardzo łatwo, a 5 – bardzo trudno. Analiza taka jest testem oprogramowania, realizowanym przez eksperta jako użytkownika.

Rozwinięta wędrówka poznawcza jest rozszerzeniem uproszczonej wędrówki poznawczej, polegającym na dołączeniu użytkowników, programistów i innych członków zespołu projektowego do zespołu ekspertów analizującego jakość oprogramowania. W konsekwencji powstaje szeroki zespół ekspertów oceniających oprogramowanie w trybie analogicznym do uproszczonej wędrówki poznawczej.

Ocena heurystyczna polega na tym, że eksperci oceniają oprogramowanie wykorzystując standardowy zestaw zasad dobrego interfejsu (tzw. heurystyki). W ocenie udział bierze od jednego do trzech ekspertów, którzy wyszukują i odnotowują wszelkie niezgodności z tymi zasadami. W trakcie oceny eksperci powinni brać również pod uwagę kontekst użycia oprogramowania. Ocenę przeprowadza się w następujących etapach:

- Planowanie działań, obejmujące wybór heurystyki i kontekstu użycia oprogramowania oraz zaznajomienie z nimi ekspertów, a także opracowanie zadań do wykonania przez ekspertów.
- Realizacja badań, w trakcie której eksperci niezależnie od siebie wykonują zadania i odnotowują wszystkie odchylenia od zasad podanych w heurystyce. Poszczególnym problemom przypisywane są liczby oceniające ich istotność.
- Analiza wyników polega na scaleniu list opracowanych przez ekspertów i oszacowaniu stopy wykrytych problemów. Poza tym przygotowuje się raport z badań wskazujący na istotność poszczególnych problemów.

Pomimo, że pojedynczy ekspert znajduje ok. 35% problemów, to już przeprowadzenie takiego badania na wczesnym etapie budowy prototypu pozwala na wypracowanie informacji zwrotnej dla projektantów. Pozwala to zmniejszyć liczbę problemów przed testowaniem z udziałem użytkowników, a więc zmniejszyć jego koszty.

Inspekcja standardów, zwana niekiedy inspekcją z wykorzystaniem list kontrolnych, polega na przeprowadzeniu formalnej kontroli interfejsu na zgodność z wymaganiami. Wykorzystuje się przy tym specjalne kwestionariusze ocen – listy kontrolne. Typowy kwestionariusz zawiera kilka, kilkanaście monotematycznych sekcji, które odpowiadają wymaganiom głównym, po kilka, kilkanaście pytań w każdej. Każde pytanie dotyczy pojedynczego wymagania z listy, przy czym pytania muszą dotyczyć wszystkich pozycji. Odpowiedź na nie może być binarna (Tak lub Nie, Spełnia lub Nie) lub też może oceniać poziom spełnienia wymagań, zwykle w 4. punktowej skali (od 2 – całkowicie nie spełnia, aż po 5 – całkowicie spełnia), lub też innej wybranej.

Inspekcję przeprowadza się w następujących etapach:

- Wybór odpowiedniej listy kontrolnej.
- Przegląd wstępny oprogramowania, jego funkcji i sposobu obsługi.
- Inspekcja oprogramowania.
- Opracowanie wyników.
- Przygotowanie raportu.

W zależności od sposobu oceny wyniki mogą być przedstawiane jako:

- Procentowy poziom spełnienia wymagań głównych.
- Uśredniony poziom spełnienia wymagań (w wybranej skali) głównych i całkowity dla całej listy.

Poza tym raport powinien zawierać szczegółową listę niespełnionych wymagań, wraz z oceną ich istotności oraz propozycjami poprawy.

4.4. Techniki i narzędzia wspomagające

Heurystyki Nielsena-Molicha [17] zostały opracowane na podstawie badań statystycznych, dotyczących prawidłowej interakcji człowiek-maszyna o możliwie najszerszym spektrum zastosowań w projektach informatycznych. Lista dziesięciu heurystyk wskazuje na podstawowe zasady konstrukcji dobrego i ergonomicznego interfejsu. Są to:

1. ***Widoczny status systemu.***
   System powinien zawsze informować użytkownika o swoim stanie z użyciem stosownych i zrozumiałych elementów systemu. Przy czym, ważne jest by oprogramowanie informowało odpowiednio szybko, bez zbędnych opóźnień.
2. ***Zgodność pomiędzy systemem a rzeczywistością.***
   System powinien być zrozumiały dla użytkownika nie tylko poprzez wersję językową, ale także poprzez dobór odpowiedniej terminologii. Wszystkie informacje powinny być podawane w logicznym, naturalnym porządku. Zrozumiałe powinny być również konwencje multimedialne (np. metafory graficzne).
3. ***Użytkownik musi mieć kontrolę i swobodę działań.***
   Zarówno w przypadku nawigacji w oprogramowaniu, jak i w przypadku wyboru błędnej opcji, użytkownik musi mieć zapewnioną prostą możliwość powrotu do poprzedniego położenia. Ważne jest by ta „ucieczka" nie wymagała zbyt długiego dialogu, a opcja ją umożliwiająca powinna być jasno oznaczona i łatwo dostępna.
4. ***Zachowanie jednakowych konwencji w obrębie serwisu.***
   Wszystkie słowa, symbole i elementy graficzne powinny być stosowane w jednakowy sposób w obrębie całego oprogramowania. Użytkownik nie powinien być zaskakiwany nietypowymi elementami (zarówno graficznymi jak i behawioralnymi) dialogu. Najlepiej jest w tym przypadku używać konwencji platformy na jakiej ma działać oprogramowanie.

5. ***Zapobieganie błędom.***
   Dialog z użytkownikiem i jego poszczególne elementy powinien być tak zrealizowany by zapobiegać błędom. Twórcy powinni być ukierunkowani na ochronę użytkownika i aplikacji przed możliwością popełnienia błędów, a nie ich obsługę.
6. ***Rozpoznawanie a nie zapamiętywanie.***
   Interfejs powinien być tak zaprojektowany by użytkownicy nie musieli pamiętać informacji przechodząc z jednej do drugiej części aplikacji. Wszystkie potrzebne w danej sytuacji informacje (np. wprowadzone dane) i instrukcje powinny być widoczne na ekranie tak, aby nie obciążać pamięci krótkookresowej użytkownika.
7. ***Elastyczność i efektywność.***
   Doświadczeni użytkownicy powinni mieć możliwość przyśpieszonego dostępu do używanych funkcji. Powinni mieć także możliwość wyboru najbardziej im pasującego sposobu wykonywania typowych zadań. Wybór ten powinien być dokonywany z wielu możliwych metod.
8. ***Estetyka i minimalizm interfejsu.***
   Interfejs nie powinien zawierać elementów zbędnych w danym momencie, utrudniających zrozumienie i przyswojenie informacji. Jednocześnie interfejs powinien mieścić się w kanonach estetyki obowiązujących powszechnie u użytkownika.
9. ***Właściwa obsługa błędów.***
   Wszystkie komunikaty powinny być napisane w sposób prosty i zwięzły, aby ułatwić użytkownikowi ich zrozumienie. Powinny one również wskazywać na typ problemu i podawać sposób jego rozwiązania. Nie powinno się posługiwać kodami błędów.
10. ***Pomoc i dokumentacja.***
    Oprogramowanie powinno być tak wykonane, by mogło być użytkowane bez sięgania do dodatkowej dokumentacji (interfejs powinien być samo-wyjaśniający). Tym niemniej oprogramowanie powinno zapewnić pomoc i dokumentację w zakresie niezbędnych do zrealizowania zadań przez użytkownika. Dostęp do tych informacji powinien być prosty i intuicyjny oraz niezajmujący więcej czasu niż jest to konieczne.

Bazując na heurystyce Nielsena-Molicha powstały różne inne heurystyki. Jedną z nich jest [13] lista LUT (tab. 1.1) w połączeniu [11] ze skalą ocen ekspertów (tab. 1.2) i opracowana na jej podstawie metryka oceny jakości interfejsu – punkty WUP (ang. *Web Usability Points*).

Punkty WUP stanowią uśrednioną po wszystkich obszarach metrykę jakości aplikacji webowej i są wyznaczane przy pomocy formuły [11]:

$$WUP = \frac{1}{n_a}\sum_{i=1}^{n_a}\frac{1}{s_i}\sum_{j=1}^{s_i}\frac{1}{q_{ij}}\sum_{k}^{q_{ij}} p_{ijk} \tag{4.1}$$

gdzie:

n_a – liczba obszarów (tab. 4.1),

s_i – liczba podobszarów w obszarze i,

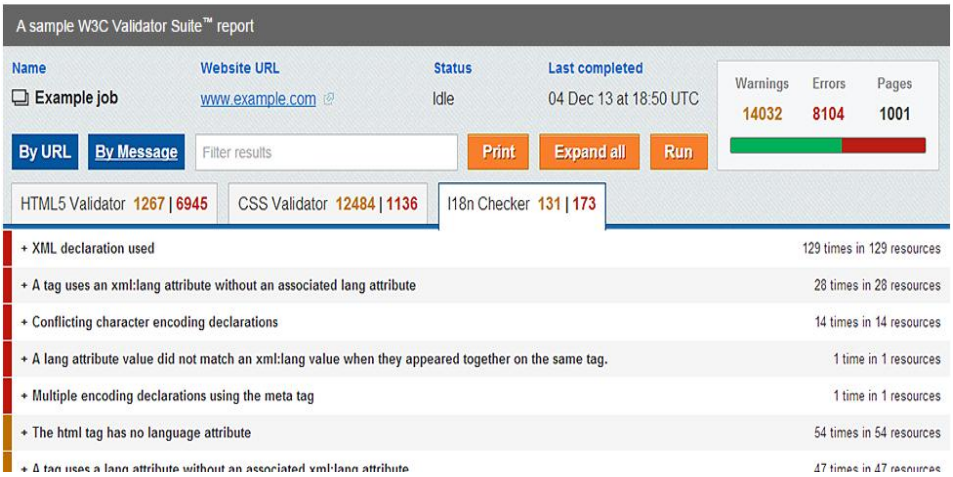
q_{ij} – liczba pytań w obszarze i oraz podobszarze j,

p_{ijk} – ocena (w punktach wg tab. 1.2) pytana o numerze k w podobszarze j obszaru i.

Wartość punktów WUP zmienia się od 1 do 5. Większa wartość oznacza lepszy interfejs.

Narzędzia komputerowego wspomagania oceny jakości interfejsów można podzielić na dwie grupy:

- narzędzia automatycznej kontroli interfejsu (przykładem jest walidator W3C – rys. 4.4);
- narzędzia wspomagające wykorzystanie list kontrolnych i heurystyk – przykład przedstawia rys. 4.5.



Rys. 4.4. Przykład raportu walidatora W3C
Źródło: https://validator-suite.w3.org/

Tab. 4.1. Lista LUT do oceny jakości interfejsu z pytaniami i ich klasyfikacją

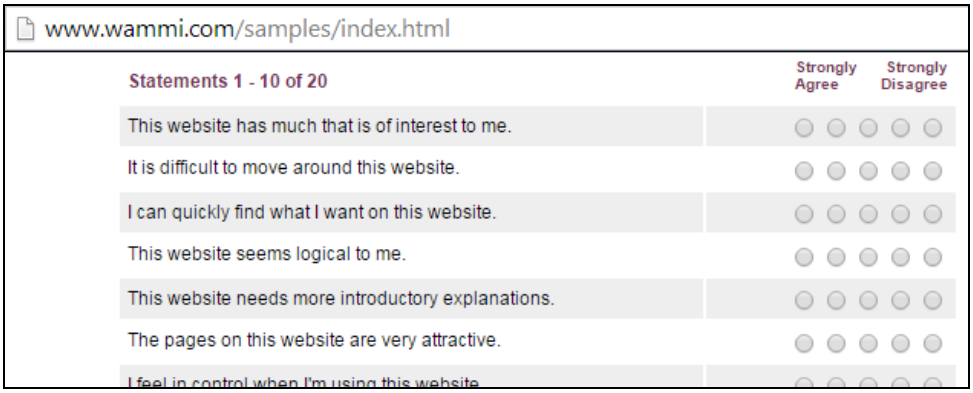
Obszar	Podobszar	Pytanie
Nawigacja i struktura	Łatwość nawigowania	Czy dostęp do wszystkich sekcji aplikacji jest łatwy i intuicyjny?
		Czy dostęp do wszystkich funkcji aplikacji jest łatwy i intuicyjny?
	Hierarchia inform.	Czy hierarchia informacji nie jest zbyt głęboka?
	Struktura informacji	Czy struktura informacji jest przemyślana?
		Czy struktura informacji jest spójna?
		Czy struktura informacji jest zrozumiała dla użytkownika?
	Elementy ekranu	Czy wspierają proces nawigacji?
Komunikaty, *feedback*, pomoc dla użytkownika	Komunikaty (ogólne)	Czy dostarczają wystarczająco dużo informacji zwrotnych dotyczących stanu operacji wykonywanych przez użytkownika?
	Komun. o błędach	Czy zawierają podpowiedzi dotyczące rozwiązania problemu?
	Informacje zwrotne i pomoc	Czy pojawiają się w miejscach, gdzie mogą być potrzebne?
		Czy treść pomocy jest dostępna dla przeciętnego użytkownika?
		Czy treść pomocy jest zrozumiała dla przeciętnego użytkownika?
		Czy prezentowane podpowiedzi bądź rozwiązania problemów są możliwe do wykonania przez zwykłego użytkownika?
Interfejs aplikacji	*Layout*	Czy *layout* jest czytelny?
		Czy *layout* jest dostosowany do różnych rozdzielczości?
		Czy *layout* jest dostosowany do urządzeń mobilnych?
		Czy układ graficzny jest spójny?
		Czy *layout* wspiera realizację zadań?
	Dobór barw	Czy kontrast pomiędzy tekstem a tłem jest odpowiedni?
		Czy dobór barw umożliwia skorzystanie z aplikacji przez osoby z zaburzeniami widzenia barw?
		Czy dobór barw umożliwia skorzystanie z aplikacji przy wykorzystaniu różnego rodzaju wyświetlaczy?
Treść podstron	Teksty	Czy są zrozumiałe dla użytkownika?
	Nazewnictwo	Czy używane w aplikacji nazewnictwo jest spójne?
		Czy używane w aplikacji nazewnictwo jest zrozumiałe?
	Etykiety	Czy używane w interfejsie etykiety dostarczają wystarczająco dużo informacji?
		Czy elementy interfejsu posiadają niezbędne etykiety?
Wprowadzanie danych	Formularze	Czy posiadają czytelny projekt?
		Czy umożliwiają wprowadzenie niezbędnych danych?
		Czy są dostosowane do urządzeń mobilnych?
	Dane	Czy zwykły użytkownik nie ma trudności z wprowadzeniem danych do formularza?
		Czy formularze posiadają elementy podpowiedzi dotyczące wprowadzanych danych (m.in. formatu, zakresu)?
		Czy formularze posiadają elementy walidujące wprowadzone dane?

Źródło: [13]

Tab. 4.2. Skala ocen dla pytań z listy LUT

Ocena	Opis
1	Wystąpiły krytyczne problemy dotyczące użyteczności, uniemożliwiające korzystanie z aplikacji bądź zniechęcające do korzystania z niej
2	Napotkano poważne problemy dotyczące użyteczności mogące uniemożliwić większości użytkowników realizacje zadań
3	Wystąpiły drobne problemy związane z użytecznością, które pojedynczo nie stanowią utrudnienia dla większości użytkowników, jednak ich nagromadzenie może wpłynąć na jakość pracy użytkownika
4	Zidentyfikowano pojedyncze drobne problemy związane z użytecznością mogące obniżyć jakość pracy z aplikacją (np. słaba czytelność tekstu)
5	Nie stwierdzono problemów związanych z użytecznością ani mających wpływ na jakość pracy użytkownika

Źródło: [11]



Rys. 4.5. Przykład elektronicznej listy kontrolnej
Źródło: http://www.wammi.com/samples/index.html

4.5. Podsumowanie

Klasyfikacja metod oceny jakości interfejsu wyróżnia metody automatyczne, tj. realizowane przez oprogramowanie, i metody manualne – realizowane ręcznie przez człowieka z możliwym wspomaganie przez oprogramowanie. Inna klasyfikacja obejmuje następujące typy metod: testowanie, inspekcja, wywiad, modelowanie analityczne i symulacja. Ocena jakości związana jest z pomiarem, który może wykorzystywać różne metryki. Można je pogrupować w metryki wydajnościowe, bazujące na problemach, bazujące na ocenach użytkownika oraz metryki behawioralne i fizjologiczne. W ocenie jakości interfejsu stosuje sie specyficzne metody z udziałem użytkownika, takie jak: testowanie interfejsu

przez użytkowników lub przyszłych użytkowników, test Kruga, okulografia, śledzenie działań użytkownika w interfejsie czy też zbieranie opinii użytkowników oprogramowania. Część metod nie wymaga udziału użytkownika. Należą do nich: analiza ekspercka, uproszczona wędrówka poznawcza, rozwinięta wędrówka poznawcza, ocena heurystyczna oraz inspekcja standardów.

Ocena jakości interfejsu z udziałem użytkownika, niezależnie od wybranej metodyki, wymaga realizacji następujących działań: przygotowanie eksperymentu, dobór i pozyskanie grupy badawczej, przeprowadzenie eksperymentu z grupą badawczą, analiza i uogólnienie wyników eksperymentu, wyciągnięcie wniosków i wypracowanie rekomendacji.

Ocena jakości interfejsu z bez udziału użytkownika wymaga zaangażowania ekspertów oraz ścisłego stosowania wybranej metody, zaleceń, standardów, list kontrolnych, analizy statystycznej itp.

Techniki wspomagające ocenę jakości interfejsu oprogramowania to przede wszystkim różnego typu heurystyki jakości oprogramowania, listy (np. LUT) oraz metryki (np. WUP).

Narzędzia wspomagające ocenę jakości interfejsu oprogramowania są zwykle różnego typu automatycznymi walidatorami, sprawdzającymi zgodność oprogramowania ze standardami lub zaleceniami. Jest to również oprogramowanie wspomagające akwizycję danych i obliczenia w metodach manualnych.

Bibliografia

1. Barnard P, Duke D, Byrne R, Davidson I., *Differentiation in cognitive and emotional meanings: An evolutionary analysis*, Cognition & Emotion *[online]*, 2007, vol. 21(6), s. 1155–1183.
2. Borys M., Plechawska-Wójcik M., *Badanie użyteczności oraz dostępności interfejsu w aplikacjach mobilnych*. Nierówności społeczne a wzrost gospodarczy, vol. 35, 2013, s. 63–77.
3. Galitz W. O. *The essential guide to user interface design*, Wiley Publishing, Inc., 2007.
4. *GUI usability and accessibility: exchanging knowledge and experiences [online]*, [dostęp 26 września 2014]. Dostępny w World Wide Web: http://www.usability-accessibility.org/project/project-results
5. *Arthur Abraham's Humane User Interface talk, 2 November 2004 [online]*, [dostęp: 20 września 2014]. Dostępny w World Wide Web: http://chandlerproject.org/Journal/HumaneUserInterface20041102
6. ISO/IEC 25010:*2011 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models*, ISO, Geneva, 2011.
7. *ISO/IEC 9126:2001 Software engineering – Product Quality. Part 1: Quality model*, ISO, Geneva, 2001.
8. Jacob R., Shaer O., Girouard A., Hirshfield L., Horn M., Solovey E., Zigelbaum J., *Reality-Based interaction: A framework for post-WIMP interfaces*, Conference On Human Factors In Computing Systems – Proceedings, 26th Annual CHI Conference on Human Factors in Computing Systems, Conference Proceedings, 2008, s. 201–210.
9. Krug S., *Nie każ mi myśleć!; O życiowym podejściu do funkcjonalności stron internetowych*. Helion, 2006.
10. Lipiec M., *Lean UX [online]*, [dostęp 20 września 2014]. Dostępny w World Wide Web: http://uxdesign.pl/lean-ux/
11. Miłosz M., Borys M., Laskowski M., *Memorability Experiment vs. Expert Method in Websites Usability Evaluation*. ICEIS 2013 – Proceedings of the 15th International Conference on Enterprise Information Systems, Angers, France, 4–7 July 2013. vol. 3, s. 176–182.
12. Miłosz M., Lujan-Mora S., *Economical Aspect of Website Memorability*, Actual Problems of Economics, No 4 (142), 2013, s. 405–414.
13. Miłosz M., Plechawska-Wójcik M., Borys M., Laskowski M., *Quality improvement of ERP system GUI using expert method: a case study*. HSI 2013, 6th International Conference on Human System Interaction, 6–8 June 2013, Sopot, s. 145–152.

14. *Model-Based UI XG Final Report [online]*, [dostęp 20 września 2014]. Dostępny w World Wide Web: http://www.w3.org/2005/ Incubator/model-based-ui/XGR-mbui-20100504/
15. *Natural user* interface *[online]*, [dostęp 20 września 2014]. Dostępny w World Wide Web: http://en.wikipedia.org/wiki/Natural_user_interface
16. Nielsen J., *10 Usability Heuristics for User Interface Design [online]*, [dostęp 20 września 2014]. Dostępny w World Wide Web: http://www.nngroup.com/articles/ten-usability-heuristics/
17. Nielsen J., *Card Sorting: How Many Users to Test [online]*, NN/g, 2004 [dostęp 20 września]. Dostępny w World Wide Web: http://www.nngroup.com/articles/card-sorting-how-many-users-to-test/
18. Piotrowska P., *ASCII Art*, Praca magisterska, ASP Gdańsk, 2009.
19. Plechawska-Wójcik M., Lujan_Mora S., Wójcik Ł., *Assessment of user experience with responsive web applications using expert method and cognitive walkthrough: a case study.* ICEIS 2013 – Proceedings of the 15th International Conference on Enterprise Information Systems, Angers, France, 4–7 July 2013, s. 60–67.
20. *PN-EN ISO 9241-210:2011 Ergonomia interakcji człowieka i systemu – Część 210: Projektowanie ukierunkowane na człowieka w przypadku systemów interaktywnych*. PKN, 2011.
21. Rasmussen, J., *Skills, rules, knowledge; signals, signs, and symbols, and other distinctions in human performance models*. IEEE Transactions on Systems, Man and Cybernetics, 1983, vol. 13, s. 257–266.
22. Sommerville I., *Software Engineering*, 9th edition, Addison-Wesley, 2011.
23. *Technologie internetowe – od teorii do praktyki*, pod red. Marka Miłosza, Polskie Towarzystwo Informatyczne, Warszawa, 2008.
24. Wachs J., Kolsch M., Stern H., Edan Y., *Vision-Based Hand-Gesture Applications*, Communications of the ACM, February 2011 No. 54(2), s. 60–71.
25. *Wikipedia. Wolna encyklopedia [online]*, [dostęp 20 września 2014]. Dostępny w World Wide Web: http://www.wikipedia.org
26. Wróblewska M., *Ergonomia. Skrypt dla studentów*, Opole, Politechnika Opolska, 2004.

Literatura dodatkowa

27. Arnowitz J., Arent M., Berger N., *Effective Prototyping for Software Makers*. Morgan Kaufmann, 2006.
28. Ballard B., *Designing the Mobile User Experience*. Wiley, 2007.
29. Chen F., *Designing Human Interface in Speech Technology*. Springer, 2005.
30. Garrett J.J., *The Elements of User Experience: User-Centered Design for the Web*. Peachpit Press, 2002.
31. Helander M., *A Guide to Human Factors and Ergonomics*. CRC Press, 2005.

32. Jones M., Marsden G., *Mobile Interaction Design*. Wiley, 2006.
33. Karwowski W., *Handbook of Standards and Guidelines in Ergonomics and Human Factors*. CRC Press, 2005.
34. Love S., *Understanding Mobile Human-Computer Interaction*. Butterworth-Heinemann, 2005.
35. Moggridge B., *Designing Interactions*. The MIT Press, 2007.
36. Morville P., Rosenfeld L., *Information Architecture for the World Wide Web: Designing Large-Scale Web Sites*. O'Reilly Media, 2006.
37. Mullet K., Sano D., *Designing Visual Interfaces: Communication Oriented Techniques*. Prentice Hall, 1994.
38. Snyder C., *Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces (Interactive Technologies).* Morgan Kaufmann, 2003.
39. Tidwell J., *Projektowanie interfejsów. Sprawdzone wzorce projektowe*. Helion, 2012.