



# Modelowanie Systemów Informacyjnych (MSI)

dr inż. Mariusz Trzaska  
[mtrzaska@pjawstk.edu.pl](mailto:mtrzaska@pjawstk.edu.pl)

## Wykład 14

Użyteczność (*usability*) graficznych interfejsów użytkownika

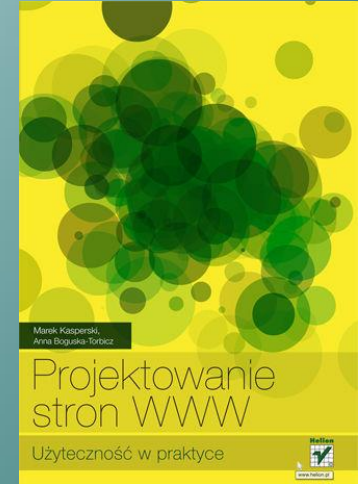
<http://www.mtrzaska.com>

# Zagadnienia

- Wprowadzenie
- Użyteczność (*usability*)
- Kształtowanie użyteczności
- Testy użyteczności
- Użyteczność, a biznes
- Lista kontrolna GUI
- Prawo *Fittsa*
- Podejścia do implementacji GUI
- Podsumowanie

# Materiały dodatkowe

- o M. Kasperski, A. Boguska-Torbicz: Projektowanie stron WWW. Użyteczność w praktyce. ISBN: 978-83-246-1291-8. Helion.
- o Steve Krug: Nie kaź mi myśleć! ISBN: 978-83-246-9274-3



# Graficzne Interfejsy Użytkownika

- **Bardzo ważny** temat związany z produkcją oprogramowania.
- ~~○ Niestety, zwykle niedoceniany.~~
- Dlaczego?
- Użytkownik ocenia aplikację przez pryzmat jej interfejsu.
- Zagadnienie z pogranicza:
  - Informatyki, m. in. programowania,
  - Psychologii,
  - Ergonomii,
  - ...



# Użyteczność (*usability*)

- Zdolność do zaspokajania potrzeb użytkownika przez:
  - Urządzenie,
  - Aplikację,
  - Interfejs.
- Kombinacja czynników kształtujących odczuwane doświadczenia użytkownika w pracy z produktem.

# Użyteczność (*usability*) (2)

- Poza oczywistym zapewnieniem bezpieczeństwa, do tych czynników należą m.in.:
  - Przydatność praktyczna. Czy system wykonuje zadania, które odpowiadają potrzebom użytkownika?
  - Łatwość nauki i obsługi. Jak szybko można nauczyć się obsługi systemu? Czy dla większości osób obsługa systemu jest wystarczająco łatwa?
  - Skuteczność. Czy system zapewni wynik zadania w takiej postaci, jak oczekuje tego użytkownik?

# Użyteczność (*usability*) (3)

- o Poza oczywistym zapewnieniem bezpieczeństwa, do tych czynników należą m.in. – c. d.:
  - Efektywność. Czy pożądaný wynik osiąga się przy umiarkowanym wysiłku ze strony użytkownika?
  - Zadowolenie. Czy użytkownik lubi pracować z systemem i czy rekomendowałby go innym?

# Kształtowanie użyteczności

- Częste badania opinii użytkowników podczas prac nad projektem,
- Testowanie prototypów z udziałem docelowych użytkowników,
- Obserwacja sposobu pracy użytkowników z systemem w rzeczywistych warunkach,
- Techniki ankietowe: wywiady i kwestionariusze.



# Użyteczność, a biznes

- W dzisiejszym świecie, wiele aplikacji posiada bardzo podobną funkcjonalność.
- W takim razie, jakimi kryteriami kierują się klienci?
  - Cena,
  - Łatwość wykorzystania funkcji systemu, co przekłada się na zadowolenie z użytkowania systemu.

Dobra użyteczność po prostu się opłaca!

## Użyteczność, a biznes (2)

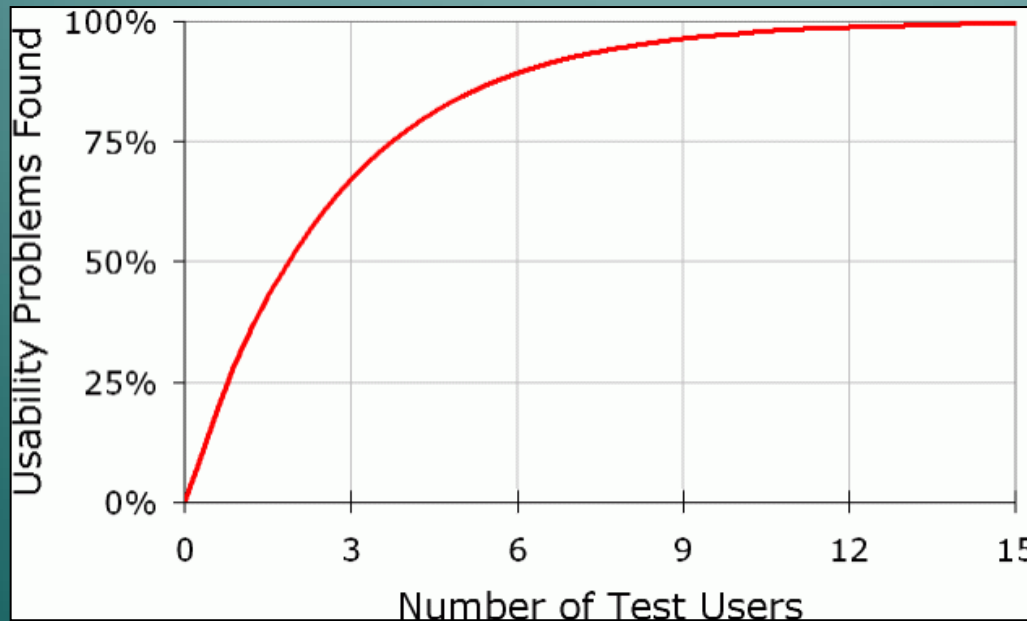
- Ostatnio nowe wersje aplikacji nie koncentrują się na dodawaniu nowych funkcji, ale na ułatwieniu dostępu (poprawie użyteczności) do już istniejących.
- Jedną z pierwszych, podkreślających nowe GUI, a nie funkcjonalności był Microsoft Office 2007.
- Aplikacje, które odniosły sukces m. in. ze względu na interfejs użytkownika.

- Testy z wykorzystaniem użytkowników:
  - Reprezentatywność testujących
  - Realizacja konkretnych (typowych) zadań
  - Obserwowanie:
    - Co użytkownicy robią?
    - W których miejscach mają trudności?
    - Rejestrowanie, np. kamera, specjalne programy.
  - Testujący powinni być zdani tylko na siebie - całkowity zakaz:
    - udzielania rad,
    - pomagania.

# Testy użyteczności (2)

Usability tests

- o Liczba testujących
  - Inaczej niż w badaniach statystycznych, nie zawsze wymagana jest duża liczba osób,
  - Zwykle grupa 5-ciu osób jest w stanie wychwycić główne problemy związane z użytecznością,



Źródło: [Jakob Nielsen, Why You Only Need to Test with 5 Users](#)

<http://www.useit.com/alertbox>



# Testy użyteczności (2)

Usability tests

- Liczba testujących – *c.d.*
  - Osoby w liczniejszych grupach przeważnie identyfikują te same problemy,
  - Lepiej iteracyjnie ulepszać projekt i testować go na małych grupach.
- Ważniejsze jest obserwowanie co testujący robią niż wysłuchanie ich późniejszego komentarza (który oczywiście też może być użyteczny).

<http://www.useit.com/alertbox>

# Testy użyteczności (3)

Usability tests

- Zanim zaczniemy nowy projekt, przetestujemy jego poprzednią wersję (oczywiście gdy jakaś była):
  - Zidentyfikujemy pozytywne i negatywne elementy,
  - Rozbudujemy te pierwsze i wyeliminujemy te drugie.
- Przetestowanie rozwiązań konkurencji,
- Wykorzystanie prototypu:
  - Papierowego,
  - „Komputerowego”
- Praca iteracyjna – **każda** iteracja kończy się testami
- Sprawdzenie projektu w kontekście znanych zaleceń dotyczących użyteczności.

<http://www.useit.com/alertbox>

# Korzyści z wysokiej użyteczności

- Krótszy czas ukończenia operacji,
- Zmniejszenie liczby błędów popełnianych przez użytkownika,
- Krótszy czas nauki i mniejszy wysiłek potrzebny do opanowania obsługi produktu,
- Zaufanie do produktu i chęć jego rozbudowy,
- Trwałe zadowolenie z produktu i zamiar polecenia go innym nabywcom.

# GUI – dobre rady

- Spójność
- Przejrzystość
- Intuicyjność
- Stosowanie się do wytycznych przygotowanych dla konkretnej platformy.
- Wykorzystanie całego ekranu – właściwe skalowanie okien.
  - Bardzo ważne
  - Czasami trudne w implementacji.



# Lista kontrolna GUI

## o Funkcjonalność

- Czy zwracasz równie dużą uwagę na funkcjonalność co na wygląd aplikacji?
- Czy Twoje okno jest w 100% zgodne z założeniami?
- Czy posiada wszystkie wyznaczone funkcje?
- Czy posiada tylko i wyłącznie wyznaczone funkcje?
- Czy pomagasz użytkownikowi w unikaniu typowych błędów poprzez umiejętne zaprojektowanie okna?
- Czy Twoje okno zgodne jest z innymi oknami tej samej aplikacji? (układ kontrolek, konwencje).

# Lista kontrolna GUI (2)

## o Platforma

- Czy Twój projekt jest w pełni zgodny z wytycznymi dotyczącymi danego interfejsu/systemu operacyjnego?
  - Domyślne kontrolki, czcionki, kolorystka,
  - Odpowiednie wymiary okien (w tym minimalne),
  - Odpowiednie ułożenie elementów,
  - Zachowanie się okien oraz kontrolek,
- Odstępstwa od powyższych zaleceń mogą oznaczać błędną pracę GUI i/lub brak „systemowych” sposobów modyfikacji.

# Lista kontrolna GUI (3)

## o Okno

- Czy Twoje okno ma nadany tytuł?
  - Czy ten tytuł jest taki sam lub zbliżony do nazwy przycisku lub opcji z menu, która otworzyła to okno?
  - Okna bez lub z domyślnym tytułem są trudne do zidentyfikowania np. w pasku zadań.
- Czy w pasku tytułu obecne są odpowiednie przyciski?
  - Czy możliwe jest powiększenie (maksymalizacja) okna,
  - Jeśli tak, czy taka możliwość jest uzasadniona?
  - Czy widoczny jest przycisk kontekstowej pomocy?

<http://www.uzytecznosc.pl>

# Lista kontrolna GUI (4)

## o Okno

- Czy dobrano odpowiedni rodzaj ramki do okna?
- Czy Twoje okno jest odpowiednio umieszczone w hierarchii?
- Czy stosujesz modalność tam, gdzie jest ona wymagana?

## o Kontrolki

- Czy wykorzystujesz intuicyjne ułożenie kontrolki i dobierasz odpowiednio ich typy, zamiast w sztuczny sposób opisywać czynności i tworzyć instrukcje?

<http://www.uzytecznosc.pl>



# Lista kontrolna GUI (5)

- Kontrolki – c. d.
  - Czy kontrolki rozmieszczone są zgodnie ze sposobem, w jaki użytkownik czyta lub skanuje ekran?
  - Czy najczęściej wykorzystywane opcje są umieszczone „najwcześniej”?
  - Czy liczba kontrolek w oknie nie jest za duża?
  - Jeśli tak, czy nie lepiej usunąć niektóre rzadziej wykorzystywane lub pokazywać je dopiero po wciśnięciu odpowiedniego przycisku?

# Lista kontrolna GUI (6)

## o Kontrolki – c. d.

- Czy stosujesz kontrolki zgodnie z ich przeznaczeniem opisanym w wytycznych i wykorzystywanym w popularnych aplikacjach?
  - Wykluczające się opcje: *radio buttons*,
  - Wiele możliwości: *check box'y*.
  - Listy z możliwością jedno- lub wielokrotnego wyboru.
- Czy Twoje kontrolki reagują w standardowy sposób? Np.:
  - przyciski po wciśnięciu powinny wykonywać komendę lub otwierać okno,
  - przyciski radiowe czy *check box'y* tylko zmieniać stan.

<http://www.uzytecznosc.pl>

# Lista kontrolna GUI (7)

## o Kontrolki – c. d.

- Czy kontrolki zbliżone znaczeniowo są pogrupowane z wykorzystaniem ramek lub pokrewnych elementów interfejsu?
- Czy nie stosujesz ramek nadmiernie, aby grupować tylko pojedyncze kontrolki?
- Czy oprogramowanie odpowiednio włącza i wyłącza nieaktywne kontrolki w razie potrzeby?

## o Przyciski

- Czy w Twoim oknie dialogowym jest obecny domyślny przycisk? Czy jest on oznaczony?

<http://www.uzytecznosc.pl>

# Lista kontrolna GUI (8)

- Przyciski – c. d.
  - Czy Twoje okno da się zamknąć w standardowy sposób? Np.:
    - przycisk na pasku tytułu,
    - klawisz Esc,
    - przycisk „Anuluj”.
  - Czy jest możliwe zamknięcie okna lub wybranie „Anuluj” bez dokonywania zmian?
  - **Użytkownik powinien zawsze mieć wyjście awaryjne z każdej sytuacji.**

# Lista kontrolna GUI (9)

- Przyciski – c. d.
  - Czy przyciski „OK,” „Anuluj” i inne odnoszące się do manipulacji okienkiem są oddzielone od innych przycisków?
- Menu
  - Czy w kontrolkach z menu (*combo boxes*) jest domyślnie wybrana jakaś opcja?
  - Czy otwierająca się lista jest odpowiednio duża (długa) w pionie tak, aby oszczędzić użytkownikowi przewijania?



# Lista kontrolna GUI (10)

- Menu –c. d.
  - Czy otwierająca się lista jest odpowiednio szeroka tak, aby każda pozycja była widoczna w całości?
- Etykiety (*labels*)
  - Czy przy kontrolkach są odpowiednie opisy (*labels*) tłumaczące ich przeznaczenie (szczególnie przy listach i menu)?
  - Czy wszystkie wyświetlane dane są opisane?
  - Czy opisy są wyrównane (w stosunku do tekstu kontrolki i do innych opisów)?

<http://www.uzytecznosc.pl>

# Lista kontrolna GUI (11)

- Opisy – c. d.
  - Czy dwukropki przy opisach są konsekwentnie stosowane lub niestosowane?
  - Czy stosujesz etykiety (*tooltips*) wyjaśniające niektóre elementy interfejsu?
    - Etykiety powtarzające standardowy opis kontrolki nie mają większego sensu.
  - Czy stosujesz polskie litery? Opisy i nazwy kontrolek bez polskich liter wyglądają niechlujnie i są mniej czytelne.

# Lista kontrolna GUI (12)

## o Klawiatura

- Czy w oknie zastosowano skróty klawiszowe, dzięki którym zaawansowani użytkownicy będą mogli szybciej skorzystać z niektórych opcji?
- Czy litery wywołujące opcje nie powtarzają się?
- Czy użytkownik może przemieszczać kursor za pomocą klawiszy *Tab* i *Shift-Tab*?
- Czy kolejność przemieszczania jest zgodna z układem okienka?
- Czy można się w ten sposób dostać do wszystkich kontrolek?

# Prawo *Fittsa*

## Eksperyment

<http://www.uzytecznosc.pl/prawofittsa/index.html>

# Prawo Fittsa (2)

- Czas wybrania celu zależy od odległości i wielkości tego celu.
- Uproszczony model Prawa Fittsa:
$$T = \log_2(D/W + 1)$$
  - $T$  jest średnim czasem potrzebnym do namierzenia celu.
  - $D$  jest odległością od punktu startowego do środka celu.
  - $W$  jest szerokością celu (mierzoną wzdłuż osi ruchu).
- Możliwość dość precyzyjnego zmierzenia użyteczności różnych elementów wskazujących.



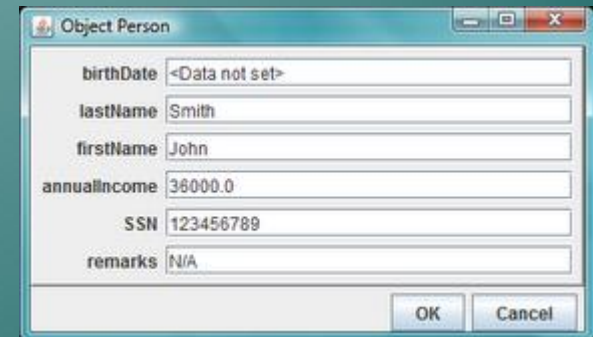
# Implementacja GUI

- Ręczna implementacja GUI jest:
  - Czasochłonna i skomplikowana,
  - Błędogenna.
- Warto używać wizualnych edytorów, które pozwalają utworzyć GUI z gotowych komponentów i podłączyć do nich zdarzenia.
- Mimo zastosowania edytora, czasami może wystąpić konieczność wprowadzania ręcznych modyfikacji.
- Podejście deklaratywne

# Implementacja GUI (2)

## o Podejście deklaratywne

- Oparte na adnotacjach, plikach konfiguracyjnych, itp.
- Wykorzystujące dedykowany język – zwykle typu DSL (*Domain Specific Language*)
  - Komunikacja przez `String`'i głównego języka programowania,
  - Modyfikacja kompilatora,
  - Specyficzna konstrukcja API udająca składnię języka:



| Field        | Value          |
|--------------|----------------|
| birthDate    | <Data not set> |
| lastName     | Smith          |
| firstName    | John           |
| annualIncome | 36000.0        |
| SSN          | 123456789      |
| remarks      | N/A            |

```
JFrame frame1 = create.frame.usingOnly(person);
```

- Przykładowy DSL: <http://code.google.com/p/gcl-dsl/>

# Implementacja GUI (3)

- Większość współczesnych języków programowania bazuje na bibliotekach zawierających gotowe komponenty sterowane zdarzeniami.
  - Java
    - AWT,
    - Swing,
    - SWT (Eclipse),
    - Java FX.
  - MS .NET (C#, C++, VB)
    - Windows Forms,
    - Windows Presentation Foundation.

# Implementacja GUI (4)

- Zestaw komponentów służących do budowania interfejsu jest dość podobny we wszystkich językach.
- Różne są (były?) manager'y rozkładu
  - Wydaje się, że najlepszy jest „kotwicowy” wykorzystywany w MS .NET oraz ostatnio również w Java.
- Dogłębne zrozumienie zasad projektowania GUI oraz jego implementacji jest dość pracochłonne i wymaga przynajmniej kilku wykładów oraz dużo ćwiczeń.

# Podsumowanie

- Graficzny Interfejs Użytkownika (*Graphical User Interface*) jest bardzo ważnym składnikiem większości aplikacji.
- Równocześnie jest jednym z bardziej niedocenianych.
- Jego poprawne zaprojektowanie nie jest sprawą prostą, ale w oparciu o istniejącą wiedzę, jak najbardziej wykonalną.
- Implementacja GUI, w niektórych przypadkach może być dość trudna (nawet w oparciu o wizualne edytory).