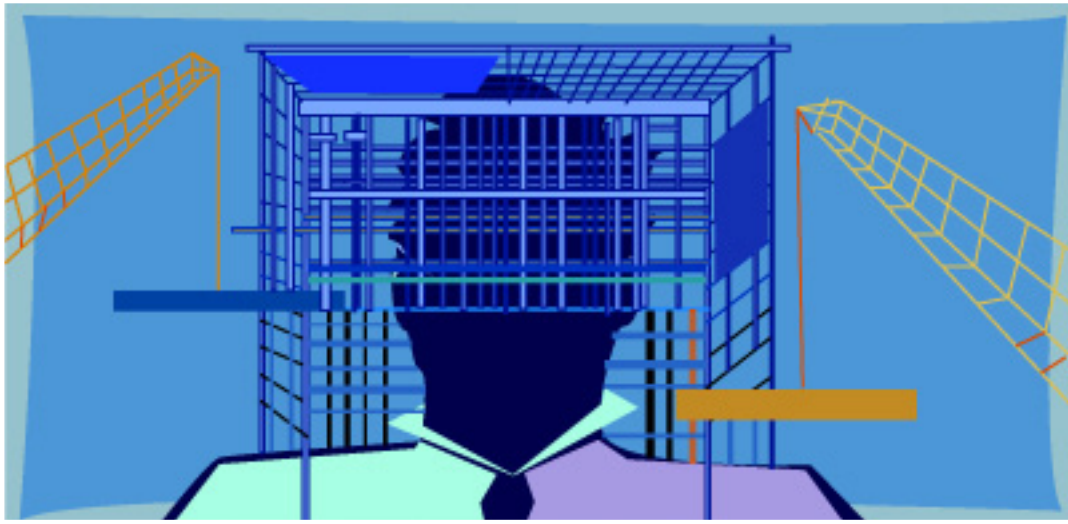
Projektowanie zorientowane na użytkownika



Mirosław Prywata

INNOWACYJNA GOSPODARKA
NARODOWA STRATEGIA SPÓJNOŚCI

PARP

WEB.GOV.PL
WSPIERAMY E-BIZNES

UNIA EUROPEJSKA
EUROPEJSKI FUNDUSZ
ROZWOJU REGIONALNEGO

Autor:
Mirosław Prywata
Infovide-Matrix

Wydawca:

Polska Agencja Rozwoju Przedsiębiorczości (PARP)
ul. Pańska 81/83
00-834 Warszawa

www.parp.gov.pl

Skład:
Małgorzata Gałązka
Infovide-Matrix

Wydanie I

Publikacja bezpłatna

Publikacja powstała w ramach projektu „Uruchomienie wielofunkcyjnej platformy komunikacji internetowej wspierającej realizację działań 8.1 i 8.2 Programu Operacyjnego Innowacyjna Gospodarka", realizowanego przez Polską Agencję Rozwoju Przedsiębiorczości, współfinansowanego ze środków Unii Europejskiej w ramach Europejskiego Funduszu Rozwoju Regionalnego.

Wspieramy e-biznes www.web.gov.pl

Spis treści

Wprowadzenie

Korzystając z Internetu, używając różnych programów komputerowych często odnosimy wrażenie, że w szerokim spektrum dostępnych rozwiązań są takie, z których korzysta nam się lepiej, w których wykonujemy nasze działania szybciej, łatwiej uzyskujemy informacje jakich potrzebujemy i są też takie, które uważamy za trudne w użyciu, hermetyczne i niezrozumiałe – mimo że jedne i drugie realizują podobne funkcje. Posługując się przykładem z innego obszaru: każdego telefonu komórkowego można użyć do nawiązania połączenia, jednak nie wszystkie są równie łatwe w obsłudze. Kwestie te były przedmiotem zainteresowania osób tworzących aplikacje od kiedy tylko pojawiły się pierwsze komputery, sposób podejścia ewoluował w czasie.

Tworzenie oprogramowania to skomplikowany proces. Podobnie jak w innych dziedzinach programy powstają w oparciu o projekt, przy czym różne sposoby i metody wytwarzania oprogramowania podchodzą w odmienny sposób do tego, jak i kiedy taki projekt powinien powstać. W samym projektowaniu aplikacji mówimy o tworzeniu zbioru wymagań. Są to ujęte zwykle w formie opisów warunki, które stawiamy aplikacji. Projektanci i twórcy systemów dzielą je na dwie podstawowe kategorie – takie, które mówią o tym, co program ma robić, jakie funkcje realizować (tzw. wymagania funkcjonalne) oraz pozostałe (tzw. wymagania pozafunkcjonalne). I właśnie wśród tych drugich istnieje duży obszar związany nie z tym, co aplikacje robią, tylko jak łatwo użytkownik może z nich korzystać – określany zwykle jako użyteczność (ang. usability).

Dobre zdefiniowanie i zrealizowanie wymagań związanych z użytecznością może być kluczowe w osiągnięciu sukcesu. Konsekwencje niewystarczającej użyteczności mogą być różne. Przykładowo jeśli aplikacja ma wspierać sprzedaż produktów to nieużyteczny interfejs może spowodować, że klienci będą porzucać stronę internetową zanim dokonają zakupów lub nie znajdując szybko informacji których szukają. W efekcie sprzedaż realizowana poprzez projektowaną aplikację będzie znacząca mniejsza, a klienci zwrócą się do konkurencji. W przypadku aplikacji, której używał będzie personel firmy do bieżących działań operacyjnych trudności z obsługą aplikacji będą powodowały, że pracownicy będą w stanie obsłużyć mniej spraw i zajmie im to więcej czasu. Wpłynie to na koszty i jakość obsługi klientów.

Jednym ze sposobów podejścia do zagadnienia użyteczności przy tworzeniu aplikacji jest projektowanie zorientowane na użytkownika.

Podstawowe pojęcia

Z projektowaniem aplikacji zorientowanych na użytkownika oraz użytecznością aplikacji związane jest wiele powszechnie używanych terminów. Poniżej kilka podstawowych pojęć z tego obszaru.

Projektowanie zorientowane na użytkownika (User-centered design, UCD) – podejście do projektowania, w którym proces projektowania opiera się na informacji o osobach, które będą używali produktu [1].

Użyteczność (usability) – termin, używany tutaj w kontekście oprogramowania, określający w sposób jakościowy łatwość posługiwania się oprogramowaniem przez użytkownik.

Projektowanie interakcji (interaction design) – dyscyplina zajmująca się projektowaniem interakcji człowieka z urządzeniami (a w szczególności oprogramowaniem).

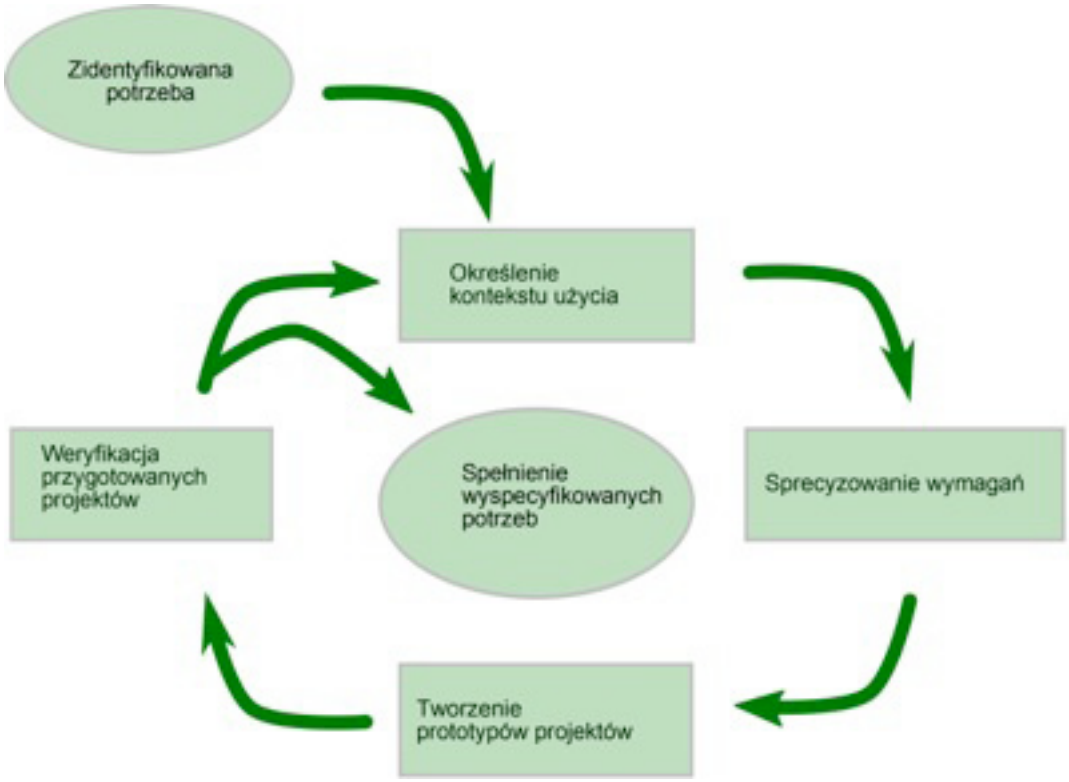
Interakcja człowiek- komputer (Human-Computer Interaction, HCI) – dziedzina zajmująca się badaniem interakcji pomiędzy człowiekiem a komputerem. Zawiera aspekty wielu różnych dyscyplin jak nauki komputerowe, informatyka, psychologia, socjologia, psychologia społecznej, psychologia poznawcza, projektowanie. Stanowi jeden z aspektów projektowania interakcji.

Projektowanie aplikacji zorientowanej na użytkownika

Tworzenie aplikacji zorientowanej na użytkownika wymaga zaplanowania i myślenia o użyteczności aplikacji w całym procesie. W szczególności siła ciężkości przenoszona jest tutaj z produktu na użytkownika. Wynika to ustawienia odpowiednich relacji: aplikacja jest dla użytkownika a nie użytkownik dla aplikacji. Odpowiednim momentem na to, by myśleć o aspektach użytkowych jest etap projektowania aplikacji. To wtedy zapadają kluczowe decyzje, które później – przy tworzeniu aplikacji oraz podczas testowania – trudno jest już poprawić. Dlatego istotnym elementem jest przeprowadzenie odpowiedniej analizy potrzeb i możliwości użytkownika.

1 Na podstawie materiałów Usability Professionals's Association http://www.usabilityprofessionals.org/usability_resources/about_usability/what_is_ucd.html

Ogólne podejście do projektowania zorientowanego na użytkownika obejmuje standard ISO 13407:1999[2]. Definiuje on w szczególności co należy zrobić, jednak nie precyzuje jak to ma zostać wykonane w konkretnym przypadku (pamiętajmy, że w ogólności projektowanie zorientowane na użytkownika dotyczy nie tylko aplikacji ale dowolnych produktów). Podstawowe założenia to koncentracja na użytkowniku i kontekście w jakim użytkownik używa produktów oraz iteracyjny proces projektowania i wytwarzania produktu. Proces ten jest schematycznie przedstawiony na rysunku 1.



Rysunek 1. Proces projektowania zorientowanego na użytkownika

Wejściem do procesu jest zidentyfikowana potrzeba, leżąca u podstaw tworzenia aplikacji. Rozpoczyna to iteracyjne tworzenie projektu. Proces ten składa się z czterech kroków

1. Określenie kontekstu użycia.

2. Sprecyzowanie wymagań.

3. Tworzenie prototypów projektów.

4. Weryfikacja przygotowanych projektów.

Kroki powtarzane są iteracyjnie aż do momentu, gdy aplikacja spełnia wyspecyfikowane wcześniej potrzeby. Iteracyjny proces wytwórczy zakłada, że wraz z kolejnym cyklem wytwarzany produkt jest określany z większą dokładnością aż do osiągnięcia wymaganego poziomu.

Faza określania kontekstu użycia programu polega na odpowiedzi na kilka podstawowych pytań, które przedstawione są poniżej.

1. **KTO**? Kim są użytkownicy? Jaką posiadają wiedzę? Jaki jest ich potencjał w zakresie nauki obsługi aplikacji? Odpowiedzi na te pytania są podstawą do charakterystyki użytkownika aplikacji.

2. **PO CO**? Jakie są oczekiwania użytkownika wobec aplikacji, czyli co chcą użytkownicy zrobić wykorzystując aplikację? Aspekt ten ma odpowiedzieć na pytania o cele biznesowe wykorzystania aplikacji przez użytkownika.

3. **SKĄD**? Z jakiego środowiska pochodzą użytkownicy, jakie mają doświadczenia (a jakich doświadczeń

2 ISO 13407:1999 "Human-centred design processes for interactive systems"

nie posiadają)? Duże znaczenie na projektowanie aplikacji zorientowanych na użytkowników ma dotychczasowe doświadczenie, pozwala określić, czego możemy oczekiwać od użytkowników w zakresie możliwości obsługi aplikacji.

4. **GDZIE**? Jaki jest kontekst użycia aplikacji przez użytkowników? Jest to kolejny istotny aspekt – sytuacja, w której użytkownicy korzystają z aplikacji.

5. **CO**? Jakie operacje ma wykonywać użytkownik, a co powinna wykonywać aplikacja? Ostatnie pytanie pozwala określić podział pomiędzy czynnościami wykonywanymi przez użytkownika oraz funkcjami, które realizuje aplikacja.

Zebrany materiał pozwala na określenie wymagań na podstawie stworzonego właśnie profilu użytkownika. Następnie należy wykonać projekty/prototypy aplikacji. Nie będą to jeszcze w pełni funkcjonalne prototypy, należy się tutaj skupić na przedstawieniu podstawowych możliwości, które można poddać dalszej pracy. W pierwszych iteracjach można przygotowywać projekty np. w formie graficznej (na papierze) i posługiwać się nimi podczas fazy weryfikacyjnej – wykonując czynności na papierowych makietach. Dużo zależy od specyfiki aplikacji i możliwości łatwego przygotowania testowego interfejsu w kilku wersjach.

Mając przygotowany zbiór propozycji przystępujemy do ich weryfikacji (ewaluacji), przy czym dokonujemy tego przy współudziale docelowych użytkowników aplikacji. Istotne jest abyśmy nie dokonywali oceny w oparciu o nasze wyobrażenia, co myślą i jak działają użytkownicy. Powinniśmy wykonać prawdziwe testy z ich udziałem, dając im do wykonania zadania, które normalnie będą wykonywać w aplikacją. W ten sposób jesteśmy w stanie ocenić projekty i przygotować się do kolejnej iteracji. Ze względu na to, że uczestnictwo we wszystkich iteracjach użytkowników może być czasochłonne i kosztowne zalecane jest skorzystanie z doświadczeń użytkowników z projektami w kluczowych iteracjach, wtedy, gdy podejmujemy najważniejsze decyzje dotyczące aplikacji. Decyzje te podejmowane są niejako przez samych użytkowników – bo wpływ na nie ma sposób, w jaki użytkownicy wykonują działania, jakie normalnie będą wykonywać na aplikacji.

Krok ewaluacji propozycji kończy iterację. Są one powtarzane do momentu, w którym będzie można stwierdzić, że projekt aplikacji osiągnął poziom odpowiadający naszym pierwotnym założeniom. Jednocześnie trzeba zwrócić uwagę na to, że planując projekt musimy także zaplanować liczbę iteracji w projektowaniu – brak określenia iteracji (wraz z celami, które powinny osiągać) prowadzi do nieprzewidywalności działań projektowych. Nie oznacza to, że liczba iteracji w wyniku prac projektowych nie może się zmienić – może ulec zmianie, jednak zmiana ta powinna mieć przyczynę, która będzie stanowiła podstawę do decyzji o skróceniu bądź wydłużeniu procesu przygotowania projektu.

Widać, że proces tworzenia aplikacji zorientowanych na użytkownika jest z jednej strony prosty z drugiej może odbiegać od standardowego podejścia, w którym po stworzeniu aplikacji dokonujemy jej testowania. Bywa tak, że usunięcie wtedy dostrzeżonych błędów użyteczności wymagałoby przebudowy całej aplikacji i wiązałoby się z dużymi kosztami.

Dobre praktyki

Dobre praktyki wytwarzania aplikacji zorientowanych na użytkownika wspierane są przez metodyki określające procesy wytwórcze, w których elementy pozwalają na odpowiednie projektowanie, wytwarzanie i testowanie aplikacji.

Wśród dobrych praktyk znajdujemy także normy ISO związane z użytecznością aplikacji. Normami tymi są

ISO 13407:1999 [3] - norma zawierająca wytyczne dotyczące działań przeprowadzanych podczas cyklu tworzenia interaktywnych systemów informatycznych przy zastosowaniu metody projektowania zorientowanego na użytkownika. Norma dotyczy projektowania systemów interaktywnych.

ISO/TR 16982:2002 [4]– norma przeznaczona dla kierowników projektów. Zawiera informacje na temat metod, które mogą zostać użyte do projektowania i testowania systemów z uwzględnieniem aspektów związanych użytecznością.

ISO-9241 [5]– wieloczęściowy standard odnoszący się do wielu obszarów interakcji użytkownika i systemu. Standard podzielony jest na 28 części, przy czym ISO jest w trakcie zmian, które mają pozwolić na znacznie obszerniejszy zakres standardu.

3 ISO 13407:1999 "Human-centred design processes for interactive systems"
4 ISO/TR 16982:2002 „Ergonomics of human-system interaction - Usability methods supporting human-centered design"
5 ISO-9241 "Ergonomics of Human System Interaction"

Główne błędy

Podczas tworzenia aplikacji zorientowanych na użytkownika możliwe jest popełnienie wielu błędów. Mogą one odbić się niekorzystnie na projekcie i użyteczności samej aplikacji. W tej części e-booka przedstawione zostaną przykłady takich błędów. Podzielone zostały na dwie grupy – błędy w samym procesie projektowania zorientowanego na użytkownika[6] oraz błędy w samym interfejsie użytkownika oraz sposobie działania aplikacji[7].

Podstawowe błędy w procesie wytwarzania aplikacji zorientowanych na użytkownika to:

- Pominięcie w procesie tworzenia badania potrzeb i kontekstu klienta. Błąd ten jest całkowitym zaprzeczeniem projektowania zorientowanego na klienta.
- Podejmowanie decyzji w oparciu o wymyślony (a nie realny) profil użytkownika. Błąd taki jest popełniany, gdy zamiast przeprowadzić prawdziwe badania sami specyfikujemy potrzeby i kontekst klienta. Bywa też tak, że grupa docelowa jest heterogeniczna (różne sposoby działania z aplikacją, różne wykonywane czynności) – w takiej sytuacji typowym błędem jest tworzenie profilu użytkownika jako średniej. W ten sposób określamy wymagania, które nie będą odpowiednie dla żadnej z podgrup.
- Przeprowadzanie złego rodzaju badań konsumenckich. W wyniku nieodpowiedniego doboru rodzaju badań otrzymamy wynik, który nie będzie określał faktycznej użyteczności produktu. Często przeprowadza się tzw. badania fokusowe, jednak w przypadku oprogramowania niekoniecznie odpowiedzą one na stawiane pytania. Lepszym wyjściem może być poproszenie osób biorących udział w badaniu o wykonanie tych czynności, które powinni normalnie wykonywać podczas pracy z aplikacją. Nagrywając ich działania można poddać analizie czas wykonywania czynności oraz zidentyfikować miejsca, nad którymi należy popracować.
- Niezaangażowanie przedstawicieli interesariuszy w przeprowadzanie testów użyteczności – nie mamy wtedy szans na uzyskanie prawdziwych zastrzeżeń do produktów. Trzeba przy tym nadmienić, że nie oznacza to, że wszystkie iteracje muszą odbywać się z zaangażowaniem interesariuszy – pamiętajmy jednak o zaangażowaniu ich w kluczowych momentach tworzenia oprogramowania, tzn. tam, gdzie podejmowane są najważniejsze decyzje.
- Brak określenia priorytetów do zastrzeżeń zgłoszonych podczas badania prototypów projektów. Bez priorytetów projekt będzie koncentrował się na rzeczach mało istotnych z biznesowego punktu widzenia.
- Brak związku badań klienta z celami biznesowymi, co może nastąpić, gdy nie mamy w projekcie określonego uzasadnienia biznesowego leżącego u podstaw realizacji projektu lub gdy to uzasadnienie nie jest właściwie komunikowane.

Poza błędami w procesie tworzenia aplikacji należy również wystrzegać się błędów w samej technologii tworzonych aplikacji. Poniższa lista zawiera przykładowe błędy, na które natykamy się w Internecie, które wpływają w istotny sposób na użyteczność, a w konsekwencji na to, jak odbieramy przeglądane strony internetowe i aplikacje internetowe.

1. Wyszukiwanie na stronie wymagające od użytkownika dokładnego wpisania poszukiwanego terminu.

2. Umieszczanie informacji jako pliki PDF (Portable Document File). Pliki PDF mają zupełnie inny format oraz przeznaczenie. Większość plików PDF przeznaczona jest do drukowania na drukarce w układzie pionowym (kartka A4 lub format letter) podczas, gdy ekran monitora ma układ poziomy.

3. Prezentowanie linków do stron odwiedzonych i nieodwiedzonych tym samym kolorem powoduje zgubienie się użytkownika, gdyż musi zapamiętywać, które obszary odwiedził lub liczyć się z tym, że niektóre treści obejrzy kilkukrotnie.

4. Prezentowanie treści w długich blokach tekstu (jak w książce) bez wyróżnień, śródtytułów, akapitów, sekcji, etc. Takie długie bloki tekstu są bardzo trudne do czytania na ekranie monitora. Brak jakichkolwiek przerw i wyróżnień powoduje prozaiczne problemy, jak trudności z przewijaniem ekranu czy znużenie czytanym tekstem.

5. Używanie czcionki zadanej z góry wielkości uniemożliwiając użytkownikowi zmianę w konfiguracji przeglądarki. W szczególności dotyczy to wstawiania animacji flash, które nie skalują się z wielkością ekranu. Właściwe jest używanie określenia wielkości względnej pozwalając użytkownikowi/przeglądarce na przeskalowanie całości automatycznie.

6. Tytuły stron które są mało związane z treścią przez co wyszukiwarki internetowe przypisują im inne konotacje. Pamiętajmy, że używanie wyszukiwarek jest jednym z częstszych sposobów w jaki użytkownicy docierają do stron.

---

6 Na podstawie http://www.interactiondesignblog.com/tag/usability-mistakes/
7 Na podstawie listy przygotowanej przez Jacoba Nielsena – Top Ten Mistakes in Web Design, 10 najczęstrzych błędów w projektach aplikacji internetowych (aktualizacja 2007)

7. Używanie form prezentacji wykorzystywanych przez reklamy. Ze względu na to, że człowiek ma możliwości selektywnego przyswajania informacji, bardzo szybko uczy się ignorować wszystko, co wygląda jak reklama (np. banery, wyskakujące okna, ruchome elementy).

8. Nieużywanie powszechnie stosowanych konwencji (jak np. układ strony, kolory linków, wygląd przycisków) w projektowaniu stron. Użytkownicy spodziewają się, że aplikacje i strony internetowe będą działać przewidywalnie, przez co mają poczucie, że mają kontrolę nad aplikacją.

9. Otwieranie nowych okien zamiast przechodzenia do nowych stron w tym samym oknie. Otwierane nowe okna kojarzą się reklamami i programami typu spyware itp. Stawiając użytkownika w niekomfortowej sytuacji, gdzie traci on kontrolę nad własnym komputerem.

10. Brak odpowiedzi na prawdziwe pytania użytkownika (dotyczy to zwłaszcza tzw. FAQ, czyli Często Zadawanych Pytań – z ang. Frequently Asked Questions), tzn. odpowiadanie na pytania, które chcielibyśmy aby użytkownik zadał, zamiast pytań, które najczęściej zadają użytkownicy.

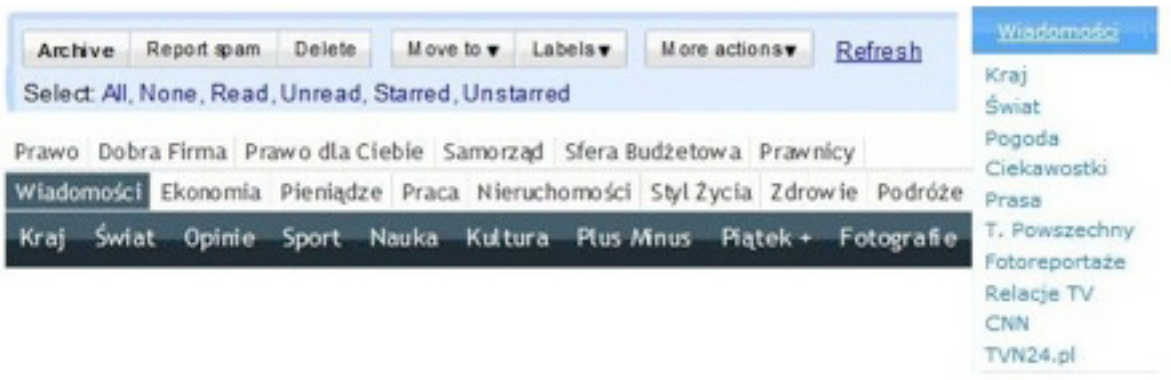
Widać więc, że w projektowaniu zorientowanych na użytkownika należy wystrzegać się błędów zarówno na poziomie odpowiedniego ustawienia procesu projektowania jak i w elementach technicznych, które są obecne w projekcie.

Przykłady elementów aplikacji internetowych

Po przedstawieniu błędów spróbujemy wskazać kilka elementów, które można określić jako użyteczne. Nie będą prezentowane pełne aplikacje. Przedstawione zostaną części aplikacji, które obecnie bywają standardowym wyposażeniem aplikacji internetowych.

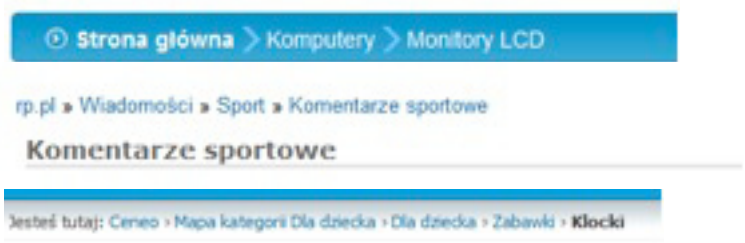
Poniżej przedstawione są przykłady dobrych praktyk poprawiające użyteczność.

- Wyszukiwarki, w których uwzględniona jest fleksja oraz możliwość popełnienia błędów w pisowni przez użytkownika. Dzięki takiemu rozwiązaniu istnieje większa szansa, że użytkownik znajdzie informację, której szuka, i nie straci zaufania do wyszukiwarki. Dobrze zaprojektowany system wyszukiwania stanowi ważny element aplikacji internetowych.
- Aplikacje, w których reklamy stanowią profilowany dodatek a nie główną treść wyświetlaną każdemu użytkownikowi jako np. wyskakujące okienka.
- Dostosowanie formy przekazu do grupy docelowej oraz doświadczenia w obsłudze komputerów użytkownika. Równie ważna obok treści jest forma – dotyczy to nie tylko samego sposobu prezentacji treści ale także używanie fachowego słownictwa i specjalistycznych terminów. Im prostszymi słowami będziemy wyjaśniać użytkownikom, co mają zrobić tym większa szansa, że zrobią to dobrze, a w przypadku dobrze sformułowanego komunikatu odpowiednio zareagują.
- Automatyczne uzupełnianie pól na podstawie danych, które dotychczas użytkownik wpisywał lub na podstawie posiadanych danych. Podpowiedzi ułatwiają, a przede wszystkim przyspieszają częste wpisywanie podobnych danych. Jeśli aplikacja służy do obsługi poczty elektronicznej, to jest duża szansa, że pisząc nowego maila będziemy go adresować do osoby, która jest u nas w książce adresowej – dlatego adres można podpowiedzieć, gdy użytkownik zacznie wpisywać pierwsze litery.
- Używanie powszechnie przyjętych standardów wykorzystywanych w aplikacjach internetowych, np. tabulacje oraz rozwijane menu (patrz Rysunek 2). Ułatwiają one poruszanie się po rozbudowanych serwisach, zwłaszcza tam, gdzie kategorii czy działów serwisu jest dużo. Należy przy tym unikać wielopoziomowych menu, bo są one trudne w obsłudze.



Rysunek 2. Przykłady menu opartych o tabulacje i rozwijane listy (przykłady pochodzą ze stron: http://gmail.com, http://www.rp.pl, http://www.onet.pl)

- Używanie w aplikacjach internetowych tzw. ścieżek dostępu (ang. breadcrumb) do nawigacji. Jest to rodzaj paska, w którym wyświetlane jest bieżące, hierarchiczne położenie użytkownika w serwisie względem głównej strony głównej. Oparte jest o analogię do katalogów i podkatalogów i umożliwia nawigowanie do wyższych poziomów w serwisie (patrz przykładowe breadcrumb - Rysunek 3). Breadcrumb ma dwie podstawowe zalety – użytkownik wie, gdzie się znajduje (gdzie znajduje się hierarchii serwisu strona, którą aktualnie wyświetla) oraz ma możliwość łatwego przejścia do dowolnego poziomu wyżej a w ten sposób ułatwia nawigację po serwisie.



Rysunek 3. Przykłady breadcrumb, do rozdzielania różnych poziomów serwisów używane są znaki lub elementy graficzne (http://www.skapiec.pl, http://www.rp.pl, http://www.ceneo.pl)

Bieżące trendy

Jeśli spojrzymy na aplikacje końca XX wieku oraz dzisiejsze programy komputerowe, to widać pomiędzy nimi dużą różnicę, która tylko częściowo wynika z rozwoju technologicznego i możliwości. Dotyczy to zwłaszcza serwisów internetowych, które w latach dziewięćdziesiątych ograniczały się do statycznych stron. Niektóre technologie pozostały niszowe i nie zdobyły internetowego rynku. Patrząc na podstawowe trendy obecne dzisiaj można wyróżnić dwa podstawowe aspekty

- Interaktywność Internetu – rozwój wszelkiego rodzaju serwisów, w których użytkownik ma szansę funkcjonować, dodawać własne treści, komunikować się z innymi. O ile w latach 90. użytkownicy tworzyli tzw. strony domowe – miejsca przechowujące statyczne treści, jak opisy i zdjęcia, o tyle teraz rolę tą odgrywają serwisy społecznościowe, blogi, komunikatory, które pozwalają nie tylko na prezentację własnych treści, ale także interakcję z innymi użytkownikami jak ocenę, komentowanie, wymianę opinii.
- Technologie mobilne, które pozwalają korzystać nie tyle z Internetu, co z pewnych szczególnych usług internetowych. Usługi te pozwalają np. na natychmiastową komunikację z serwisami społecznościowymi. Jeśli do tego dołączymy usługi geolokalizacji to zamiast topologii Internetu mamy usługi bazujące na naszym rzeczywistym położeniu. Przykładem jest wprowadzony przez Apple iPhone, którego możliwości są ściśle związane z dostępem do Internetu.

Oba te trendy rozwijają się równolegle, wykorzystują nawzajem i wspierają. Dlatego w najbliższym czasie należy spodziewać się dalszej integracji w tych obszarach na co wskazuje np. nowy produkt firmy Google – Buzz.google.com. Nie wiadomo jeszcze jaki będzie odbiór przez użytkowników, natomiast serwis integruje dane geolokalizacji i dostęp z telefonów komórkowych z uczestnictwem w portalach społecznościowych oraz pocztą elektroniczną.

Ile to kosztuje

Jedną z kwestii pozostaje koszt, który musimy ponieść podczas wytwarzania aplikacji na testy użyteczności. Odpowiednio zaplanowany proces projektowania zorientowanego na użytkownika nie musi być bardzo drogi. Jeśli przygotowujemy dużą aplikację, gdzie koszty są rzędu milionów złotych, to warto poświęcić kilkanaście czy nawet kilkadziesiąt tysięcy złotych na odpowiednie testy. Uchronią nas przed znacznie droższymi, bo mogącymi iść w setki tysięcy złotych, zmianami w ostatnich etapach tworzenia aplikacji. Pamiętajmy, że same testy użyteczności nie są celem samym w sobie. Przy małych projektach zamiast przygotowywać drogie i długotrwałe testy możemy spróbować konsultować nasze pomysły z profesjonalistami z obszaru użyteczności, by korzystać z dobrych praktyk i uniknąć najgorszych błędów, które w konsekwencji mogą być bardzo kosztowne.

Podsumowanie

Użyteczność aplikacji jest ważnym aspektem tworzenia oprogramowania. Wśród wielu metod zapewnienia użyteczności jest projektowanie zorientowane na użytkownika – łączy ono w sobie elementy odpowiedniego profilowania użytkownika – odkrywania potrzeb i możliwości z włączeniem przedstawionych działań w proces wytwórczy. Aby móc wykonać takie działania powinniśmy się do nich przygotować na etapie planowania tworzenia aplikacji poprzez włączenie odpowiednich zadań do harmonogramu oraz przyjęcia podejścia, które cechuje nastawienia na użytkownika. Tekst nie wyczerpuje tego obszernego tematu – osoby zainteresowane mogą poszerzyć wiedzę korzystając z załączonego spisu literatury

związanej z użytecznością aplikacji.

Źródła

The Fable of the USER-CENTRED DESIGNER - David Travis, książka dostępna w Internecie, możliwość zakupu wersji papierowej, http://www.userfocus.co.uk/fable/index.html

http://www.usabilityprofessionals.org/ - strona Usability Profesionals's Association

http://www.upassoc.org/upa_projects/body_of_knowledge/bok.html - kompendium wiedzy nt. użyteczności (na stronach UPA)

http://www.interaction-design.org/ - strona internetowa grupująca informacje nt. projektowania interakcji człowiek-komputer.

http://www.ixda.org/ - strona internetowa IxDA stowarzyszenia projektowania interakcji

http://www.designinginteractions.com/interviews - zestaw wywiadów z czołowymi postaciami zajmującymi się projektowaniem interakcji.

http://www.useit.com/ - strona Jacoba Nielsena zawierająca informacje i odnośniki związane z użytecznością aplikacji.

Polecana literatura

Tom Stafford, Matt Webb - 100 sposobów na zgłębienie tajemnic umysłu
Mark Pearrow - Funkcjonalność stron internetowych
Steve Krug - Nie każ mi myśleć. O życiowym podejściu do funkcjonalności stron internetowych
Jakob Nielsen, Hoa Loranger - Optymalizacja funkcjonalności serwisów internetowych
James Kalbach - Projektowanie nawigacji strony WWW. Optymalizacja funkcjonalności witryny
Jeffrey Zeldman - Projektowanie serwisów WWW. Standardy sieciowe.
Paweł Frankowski, Arvind Juneja - Serwisy społecznościowe. Budowa, administracja i moderacja
Joshua Porter - Serwisy społecznościowe. Projektowanie
June Cohen - Serwisy WWW. Projektowanie, tworzenie i zarządzanie
Mark Breier - Szybcy i mądrzy
Tomasz Karwatka - Usability w e-biznesie. Co kieruje Twoim klientem?
Alan Cooper - Wariaci rządzą domem wariatów
Amy Shuen - Web 2.0. Przewodnik po strategiach
Joel Sklar - Zasady tworzenia stron WWW
Russ Unger, Carolyn Chandler - A Project Guide to UX Design: For user experience designers in the field or in the making
Alan Cooper, Robert Reimann, David Cronin - About Face 3: The Essentials of Interaction Design
Peter Morville - Ambient Findability: What We Find Changes Who We Become
Dan Brown - Communicating Design: Developing Web Site Documentation for Design and Planning
Jeremy Sydik - Design Accessible Web Sites
Bill Moggridge - Designing interactions
Jenifer Tidwell - Designing Interfaces - Patterns for Effective Interaction Design
Bill Scott, Theresa Neil - Designing Web Interfaces: Principles and Patterns for Rich Interactions
Jonathan Arnowitz, Michael Arent, Nevin Berger - Effective Prototyping for Software Makers
Jeff Johnson - GUI Bloopers 2.0, Second Edition: Common User Interface Design Don'ts and Dos
Jeffrey Rubin, Dana Chisnell, Jared Spool - Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests
O'Reilly Peter Morville - Information Architecture for the World Wide Web, Third Edition
Janice (Ginny) Redish - Letting Go of the Words: Writing Web Content that Works
Tom Tullis - Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics
Carolyn Snyder - Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces
Luke Wroblewski - Site-Seeing: A Visual Approach to Web Usability
Bill Buxton - Sketching User Experiences: Getting the Design Right and the Right Design
Gene Smith - Tagging: People-powered Metadata for the Social Web
Douglas van Duyne, James Landay, Jason Hong - The design of sites
Jesse James Garrett - The Elements of User Experience: User-Centered Design for the Web (Voices That Matter)
Tom Brinck, Darren Gergle, Scott D. Wood - Usability for the Web: Designing Web Sites that Work
James Governor, Dion Hinchcliffe,, Duane Nickull - Web 2.0 Architectures
Jennifer Niederst - Web Design in a Nutshell: A Desktop Quick Reference
Ashley Friedlein - Web Project Management: Delivering Successful Commercial Web Sites
Kelly Goto, Emily Cotler - Web ReDesign 2.0: Workflow that Works