This paper should be cited as: Bączyk, P., & Szóstek, Ł. (2011). Co warto wiedzieć projektując interfejsy użytkownika w dużej firmie. Proceedings of the Conference: Interfejs użytkownika - Kansei w praktyce, Warszawa 2011. Warsaw: Wydawnictwo PJWSTK.

Co warto wiedzieć projektując interfejsy użytkownika w dużej firmie

Piotr Bączyk
Orange Labs Poland
ul. Obrzeżna 7
02-691 Warszawa, Polska
Piotr.Baczyk@telekomunikacja.pl

Łukasz Szóstek
Orange Labs Poland
ul. Obrzeżna 7
02-691 Warszawa, Polska
Lukasz.Szostek@telekomunikacja.pl

Interfejs użytkownika – Kansei w praktyce 2011
ISBN 978-83-89244-99-4

Abstract
In business practice, the final shape of user interface (UI) is determined by many factors from outside of traditional designer's domain. In this paper we present the best practices we have come across that should be taken into account when running projects where user interface is considered a key factor. We describe these practices in the context of two commercial projects we have been leading. The first one, b-Link, was developed at Orange Labs Poland in cooperation with Technical University of Lodz. The other one, called Calypso, was developed in The Netherlands and USA between Philips Research Labs, Sony and Intertrust.

Keywords
user interface, user interface design, best practices, role of UI designer, project management, PMI, agile programming

Wstęp

Wiele się mówi o umiejętnościach i technikach pracy projektanta w czasie projektowania interfejsu człowiek-maszyna. Niewiele zaś mówi się o otoczeniu, w jakim zachodzi tworzenie ostatecznego kształtu tego interfejsu. A przecież trudno zaprzeczyć, że zarówno twórca jak i otoczenie, w jakim on pracuje, wzajemnie na siebie oddziałują. W naszym referacie skupimy się właśnie na wpływie organizacji na ostateczny kształt interfejsu. Dla

zwiększenia przejrzystości tekstu, pisząc „interfejs” będziemy mieli na myśli interfejs użytkownika. Wzmianki o innych interfejsach (programowych czy sprzętowych) będziemy jednoznacznie opisywać.

Oddziaływanie organizacji na przebieg projektów najlepiej będzie przeanalizować na konkretnym przykładzie. Sięgniemy po dwa przykłady projektów, które prowadziliśmy, a które naszym zdaniem będą wystarczająco reprezentatywne dla pokazania, jakim wpływom może podlegać interfejs użytkownika (user interface—UI) powstający w dużych firmach. Pierwszy z nich b-Link zakończył się wdrożeniem komercyjnym, a drugi Calypso został zatrzymany na etapie prototypu. Dla klarownego przedstawienia kluczowych momentów dla budowy UI dalsze rozdziały będą opatrzone nazwami faz projektu występującymi najczęściej w projektach realizowanych w dużych firmach.

Zacznijmy od krótkiego przedstawienia celów obu projektów.

b-Link to projekt, który miał na celu wdrożenie programu umożliwiającego korzystanie z komputera i zasobów sieci Internet osobom o dużym stopniu niepełnosprawności narządów ruchu. W szczególności oprogramowanie miało umożliwiać korzystanie osobom niepełnosprawnym z komputera bez konieczności zakupu specjalistycznych urządzeń (rysunek 1). Równocześnie założono, że Orange Labs Poland (OLP) udostępni to oprogramowanie za darmo na licencji Open Source, tak, aby każdy mógł z niego korzystać, a także umożliwić swobodny rozwój aplikacji przez społeczność internetową[7]. Cel projektu został sformułowany w porozumieniu z działami public relations Telekomunikacji Polskiej (TP) i przyjęty do realizacji według metodologii PMI [2][3]. W projekcie pracowało 7 osób, tj. 4 programistów, architekt-programista, grafik i kierownik projektu (Piotr Bączyk).

Rysunek 1. Ogólna zasada działania programu b-Link

Projekt Calypso miał na celu skonstruowanie przenośnego odtwarzacza video implementującego system zarządzania prawami (DRM - Digital Rights Management) Marlin [6]. Wynikiem projektu miało być działające urządzenie gotowe do wdrożenia pilotowego na małą skalę: kilkadziesiąt tytułów dla kilkuset użytkowników. Powyższy cel projektu został sformułowany na najwyższym stopniu zarządzania firmy Philips i przekazany do realizacji w Philips Research Laboratories. Implementacją zajmował się zespół 6 osób w Holandii składający się z 4 programistów, architekta oraz kierownika projektu (Łukasz Szóstek) wspomagany przez zespół 4 programistów z firmy Intertrust realizujących część funkcjonalności DRM i jednego projektanta interfejsów z Philips Design. Grupa w Holandii pracowała z użyciem metod

[7] http://b-link.sourceforge.net

"agile development" [7], w szczególności "extreme programming"[8].

Pomysł na projekt

Etap krystalizowania się pomysłu na przyszły projekt jest najmniej formalnym etapem w cyklu życia projektu a często może się odbywać poza strukturą firmy. Nie powinno to dziwić, gdyż na tym etapie zależy nam szczególnie na nieskrępowanym generowaniu pomysłów, spośród których później możemy dokonać wyboru przyszłego celu.

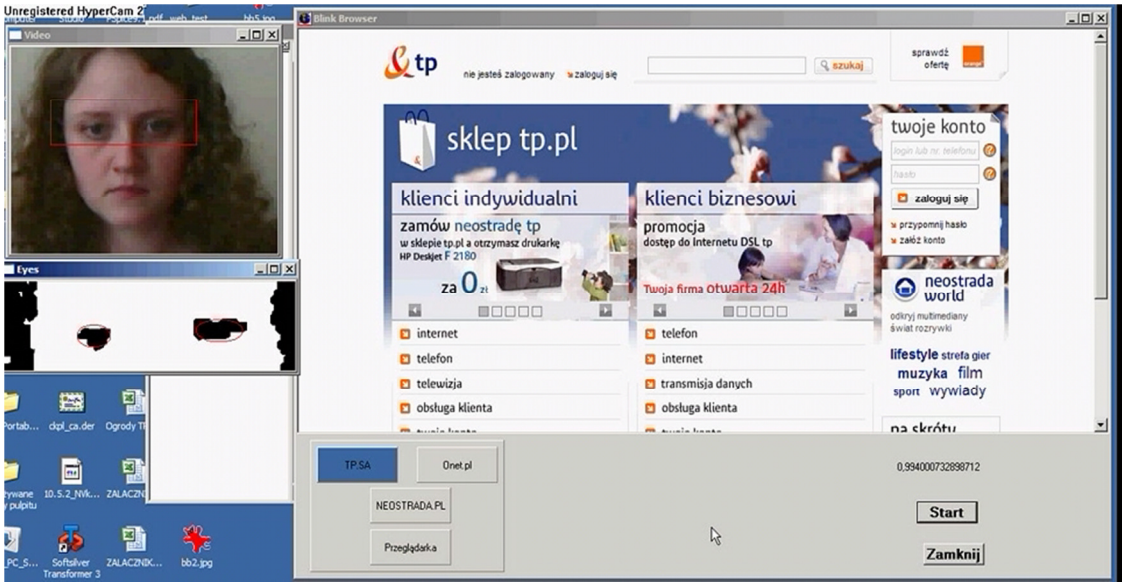
Jeśli wokół tego etapu w dużych firmach tworzy się jakieś procedury, to dotyczą one przeważnie motywowania osób do tworzenia pomysłów oraz obiektywnej oceny przydatności dla firmy wygenerowanych pomysłów. Jak już powiedzieliśmy, etap ten charakteryzuje największa kreatywność. W przeważającej większości projektów, które mają pozostać nie tylko prototypem technologicznym, ta kreatywność przekłada się w największym stopniu na budowę UI.

W przypadku projektu b-Link, etap formułowania celu projektu był najdłuższym etapem prac. Rozpoczął się on od wykonanego przez OLP przeglądu prac z zakresu elektroniki medycznej prowadzonych przez Politechnikę Łódzką. W toku przeglądu zainteresowano się kilkoma pomysłami z, pośród których jeden dotyczył prac nad śledzeniem szybkością mrugania powiekami w zależności od stanu psychofizycznego człowieka w ramach doktoratu prowadzonego na tej uczelni [1]. Na tym etapie w Ogrodach Innowacji[8] zaprezentowany został wczesny prototyp UI pozwalający sterować kursorem myszy za pomocą mrugania powiekami. Nie przypadkiem na tak wczesnym etapie powstał pierwszy, wtedy jeszcze niedoskonały i niekompletny, pomysł na UI – dziś moglibyśmy o nim powiedzieć „jakiś UI" (rysunek 2) – pozwolił on na bardzo wczesnym etapie realizacji przedsięwzięcia sprawdzić i zademonstrować pomysł na przyszły produkt.

Projekt Calypso powstawał inaczej. Sposób, w jaki zadanie zostało zlecone zespołowi, pozostawiał stosunkowo mało miejsca na inwencję w zakresie funkcjonalności. Odtwarzacz miał być przedmiotem wdrożenia pilotowego i należało trzymać się koncepcji i funkcjonalności, której spodziewają się po takim urządzeniu użytkownicy, natomiast wiadomo było, że wyzwaniem będzie obsługa funkcjonalności DRM.

Funkcjonalność DRM, czyli ograniczania dostępu do treści (w naszym przypadku filmów), jest sama w sobie trudna do implementacji w przejrzysty dla użytkownika sposób. Dopóki dany użytkownik ma prawa, których wymaga dostawca treści, funkcjonalność ta jest wprawdzie niewidoczna. W momencie, gdy użytkownik tych praw nie posiada, należy mu pokazać, dlaczego dany film nie może być odtworzony czy skopiowany, jak również co użytkownik powinien zrobić, aby takie prawa nabyć. Sytuację dodatkowo komplikował fakt, że Marlin, wychodząc naprzeciw problemom zgłaszanym przez użytkowników został zaprojektowany jako dużo bardziej elastyczny niż tradycyjne systemy DRM, jak na przykład Windows Media czy iTunes. Marlin umożliwia definiowanie grup ludzi (na przykład rodziny) oraz przypisywanie urządzeń zarówno do pojedynczych osób jak i do grup.

[8] Ogrody Innowacji to specjalne miejsce które Orange Labs Poland przeznaczyło na prezentacje nowych, innowacyjnych pomysłów.



Rysunek 2. Bardzo wczesna implementacja pomysłu rozwiniętego w projekcie b-Link

Z kolei treści chronione przez system Marlin mogą również być przypisywane zarówno do ludzi, do grup ludzi (np. rodziny, współlokatorów) jak i do urządzeń i ich grup.

Takie założenia systemu zostały zdefiniowane przez firmy zajmujące się dystrybucją filmów (głównie z Hollywood), aby umożliwić realizację różnych modeli biznesowych nie ograniczając się możliwościami technologii DRM. Niestety, z punktu widzenia użytkownika takie podejście bardzo komplikowało interakcję z urządzeniem w szczególności jego konfigurację. Musieliśmy zapewnić użytkownikowi możliwość „odwzorowania" w systemie zależności pomiędzy różnymi osobami należącymi do grupy oraz ich urządzeniami tak jak wymagał tego Marlin. Ponadto musiało być jasne dla użytkownika jakich praw wymaga zamawiany czy zakupiony już przez niego film (np. ten film może być oglądany na każdym urządzeniu należącym do członka rodziny użytkownika).

Drugim ograniczeniem dla projektu interfejsu była forma urządzenia, na którym projekt miał być zaimplementowany. Na etapie organizacji nie było jeszcze wiadomo, jak będzie wyglądać – i czy w ogóle istnieje – urządzenie, którego moglibyśmy użyć.

Podsumowując:

- Prototypy UI mogą, a często powinny, powstawać na bardzo wczesnym etapie projektu, ponieważ sprzyjają otrzymaniu szybkiej informacji zwrotnej zarówno od

użytkownika, jak i osób decydujących o być albo nie być danego projektu.

- Jednocześnie na etapie definiowania projektu często jest jeszcze dużo niewiadomych, które mogą spowodować, że żadne wiążące decyzje dotyczące interfejsu użytkownika nie mogą być podjęte. Nie oznacza to jednak, że nie powinno się rozważać różnych opcji, identyfikować potencjalnych problemów itp.
- Nawet najbardziej uproszczony prototyp UI pozwala zademonstrować najbardziej skomplikowaną technologię w działaniu, zanim podejmiemy decyzje o wydaniu większej sumy pieniędzy na jej pełną implementację.

Formalne rozpoczęcie projektu

Formalne rozpoczęcie projektu jest momentem, w którym muszą zostać ustalone trzy podstawowe parametry projektu: jego zakres merytoryczny, czas realizacji oraz budżet. Określenie tych wymiarów projektu może przyjmować różne formy i zależy między innymi od tego, czy budowane rozwiązanie ma trafić po zakończeniu projektu od razu na rynek czy ma pozostać prototypem.

Dla projektu b-Link formalnym rozpoczęciem było podpisanie karty projektu [3], tj. dokumentu formalnego zawierającego krótki opis celu projektu, zakres projektu zdefiniowany przez kluczowe funkcjonalności, listę osób wraz z rolami pełnionymi w projekcie oraz czas rozpoczęcia i zakończenia projektu, a także przyznany budżet. W karcie tej zostało też określone, jaka ma być forma organizacji projektu. W przypadku b-Link była to metodologia Project Management Institute (PMI) [3].

Z punktu widzenia UI ważny zapis znalazł się w części opisującej kluczowe funkcjonalności, zobowiązując projekt do udostępnienia aplikacji na otwartej licencji GPL [4]. Licencja ta ogranicza zakres zewnętrznych bibliotek programistycznych wykorzystanych w projekcie do objętych tą samą licencją, co miało istotny wpływ na prace deweloperskie i projektowe. Równocześnie wszystko, co zostało stworzone w projekcie, miało być udostępnione za darmo użytkownikom i społeczności internetowej wraz z kodem źródłowym.

Projekt Calypso startował mniej formalnie. Projekt został uruchomiony bezpośrednio przez (i na życzenie) wysoko postawionego dyrektora, więc nie wpisywał się w standardowe procedury pozyskiwania budżetu, a kierownik i architekt projektu został zaangażowany już w wyżej opisanej fazie organizacji. Za punkt startu można by ewentualnie przyjąć skompletowanie zespołu i rozpoczęcie przez niego pracy, ale formalnie taki moment trudno zidentyfikować.

Podsumowując:

- Etap inicjacji projektu może pozostać bez wpływu na pracę projektanta UI, o ile zapisy z tego dokumentu zostaną przeanalizowane na etapie tworzenia wymagań funkcjonalnych i prawnych.

Wymagania funkcjonalne

Etap wymagań funkcjonalnych jest bardzo ważny z punktu widzenia projektanta UI, ponieważ to właśnie tutaj określane są cechy, jakimi ma się charakteryzować produkt dla przyszłego użytkownika. W praktyce oznacza to, że to właśnie na tym etapie powstaje pierwsza wersja interfejsu użytkownika – lub kolejna, bardziej dojrzała, jeśli wcześniej powstał pre-prototyp lub choćby prezentacja opisująca jego potencjalny przyszły kształt.

Przywykliśmy myśleć, że pracę związaną z projektowaniem interfejsu użytkownika wykonuje tylko jego projektant, tymczasem nasze doświadczenia pokazują, że współtworzy go cały zespół projektowy, włączając w to również sponsora. O sposobie pracy całego zespołu projektowego dla realizacji tego konkretnego zadania wiele już powiedziano. Jednak w praktyce metodę pracy grupy na tym etapie trzeba dostosować do takich czynników jak:

- wiedza merytoryczna poszczególnych członków zespołu,
- zaangażowanie wszystkich członków zespołu,
- dystans pomiędzy poszczególnymi członkami zespołu
- jakości opisu idei/pomysłu jaki „otrzymaliśmy" od zleceniodawcy.

W przypadku b-Linka na tym etapie zastosowano metodę mieszaną tj. burzy mózgów przeplatanej coaching-iem. Wprawdzie zespół dysponował wystarczającą wiedzą merytoryczną oraz był dostatecznie zaangażowany aby cel ten realizować bez udziału kierownika projektu (coacha) jednak zespół pracował po raz pierwszy dla dużej firmy w której wiedza o interesariuszach i koordynacja były kluczowymi czynnikami do osiągnięcia celu. Zdecydowano, że wymagania funkcjonalne będą zbierane w oparciu o metodę burzy mózgów, a porządkowane na spotkaniach z udziałem coach-a. Forma, jaką przyjęły wymagania funkcjonalne dla projektu b-Link, była zgodna z rekomendacjami metodyki PMI [3] tj. przyjęły formę listy ponumerowanych hierarchicznie wymagań.

W przypadku projektu Calypso formalnie nie stworzono dokumentu opisującego wymagania funkcjonalne. Teoretycznie metody agile nie stawiają żadnych wymagań względem dokumentacji, zwłaszcza przed rozpoczęciem implementacji. W naszym jednak przypadku architekt w przeciwieństwie do zespołu dysponował dużą wiedzą z zakresu działania i budowy systemu Marlin. Dla zaoszczędzenia czasu, jaki zespół musiałby włożyć w poznanie podstaw Marlina architekt jeszcze przed implementacją zbudował działający na symulatorze model odzwierciedlający docelowe działanie prototypu. Takie nietypowe podejście okazało się, z kilku powodów, bardzo skutecznym narzędziem specyfikacji funkcjonalnej a także komunikacji z zespołem. Po pierwsze, zbudowanie działającego modelu wymaga od jego twórcy dogłębnego przemyślenia rozwiązań technicznych niemożliwych do przewidzenia w inny sposób. Po drugie, z obserwacji takiego modelu zespół doświadczonych programistów może, często podświadomie, wyciągnąć wiele wniosków sugerujących sposób myślenia i podejścia projektowego architekta. W końcu, działający model zawiera w sobie taką ilość szczegółowych informacji, że spisanie ich w dokumencie jest nierealne – a jeśli już, to taki dokument byłby bardzo długi i mniej czytelny niż model.

Ciekawą obserwacją był fakt, że pomimo, że programiści dostali do wglądu działający model, który mogli analizować w dowolny sposób, podstawowym środkiem komunikacji między architektem a programistami stał się wykres MSC (message sequence chart) wygenerowany z modelu.

Z punktu widzenia projektanta interfejsu etap powstawania tego modelu był kluczowy. Na tym etapie zostały podjęte wszystkie fundamentalne decyzje dotyczące architektury rozwiązania, a więc również o tym, co może być prezentowane na interfejsie. Niestety po kilku miesiącach pracy nad projektem wystąpiły niespodziewane okoliczności, które zmusiły nas do zmiany tych decyzji, ale o tym piszemy dalej.

Podsumowując:

- Etap zbierania wymagań jest najważniejszym z punktu widzenia prototypowania UI. Na tym etapie zostają podjęte fundamentalne decyzje o tym jak będzie wyglądał produkt i wszystkie zmiany wprowadzane później spotkają się z dużym oporem zespołu oraz obarczone będą większym kosztem wprowadzenia.
- Na etapie wymagań funkcjonalnych UI projektowany jest nie tylko przez projektanta interfejsu, ale również przez wszystkich członków zespołu projektowego niezależnie od metodyki prowadzenia projektu.

Wymagania techniczne

Tradycyjnie każde wymaganie funkcjonalne powinno znaleźć swoje, przynajmniej jedno wymaganie techniczne. Jest to oczywiście teoria. W praktyce jeśli jednemu wymaganiu funkcjonalnemu odpowiadają dwa lub trzy wymagania techniczne, wtedy możemy czuć się „spokojni" co do właściwego zrozumienia potrzeb klienta. W przypadku mniejszej ich liczby niż 2 można się spodziewać problemów na późniejszych etapach projektu. Uogólnienie to stosuje się również do projektowania UI.

Najważniejsze dla projektu UI na tym etapie jest, aby właściwie zidentyfikować środowisko, w jakim daną technologię w tym UI będziemy tworzyć, tak, aby wybrana technologia nie stałą się barierą dla funkcjonalności, jakie chcemy zaimplementować.

Dla projektu b-Link wymagania takie powstały jako relacja: jedno wymaganie funkcjonalne do wielu wymagań technicznych. Utrzymywanie relacji pomiędzy wymaganiami technicznymi a funkcjonalnymi (również w obszarze UI) pozawala na kontrolę zakresu zmian koniecznych w przypadku zmiany jednego z wymagań. Rozwiązanie takie przyjęto zarówno stosując metodykę PMI [3] jak i w oparciu o doświadczenia kierownika projektu gdyż pozwala w stosunkowo łatwy sposób panować nad zmianami w projekcie w szczególności nad tym jego obszarem, na którym zależy zamawiającemu.

Zespół Calypso, realizując metodę extreme programming nie był zobowiązany do ścisłego utrzymywania dokumentacji projektowej. Jednocześnie wymagania funkcjonalne były całkowicie zależne od czynników zewnętrznych, mianowicie od wyboru platformy sprzętowej. W tamtym czasie trudno było znaleźć przenośną platformę umożliwiającą odtwarzanie video skompresowanego i sformatowanego zgodnie z wymaganiami Marlina i pozostawiającego wystarczająco dużo mocy obliczeniowej do działania systemu DRM, który opiera się na szyfrowaniu treści. W końcu udało się pozyskać z firmy AMD prototyp ich nowopowstającej platformy i to właśnie wymagania tej platformy narzuciły nam ogólne wymagania techniczne, jak wybór języków programowania, sposób integracji z UI oraz sam kształt interfejsu użytkownika. Bardziej szczegółowe decyzje techniczne były podejmowane na bieżąco, w miarę jak pojawiały się w toku implementacji.

Podsumowując:

- Budowa wymagań technicznych, z punktu widzenia UI, jest pierwszą i najprostszą metodą sprawdzenia, jak dobrze zrozumieliśmy wymagania funkcjonalne, które stawia przed nami zamawiający
- Z drugiej strony, jeśli wymagania techniczne są narzucone przez wybraną technologię, mogą być one dużym ograniczeniem dla wyobraźni projektanta – ale jednocześnie i wyzwaniem dla jego pomysłowości (jak zbudować możliwie najlepszy interfejs przy danych ograniczeniach technicznych).
- Podczas określania wymagań technicznych dobrą praktyką jest takie ich sformułowanie, aby nie ustanawiały niepotrzebnych ograniczeń dla budowy UI. Zarówno w przypadku produktów jak i prototypów, w których UI jest kluczowym czynnikiem sukcesu, tenże interfejs będzie zawsze pierwszą, a często najważniejszą częścią systemu z którą zetkną się oceniający (odpowiednio nasi klienci lub zwierzchnicy).

Implementacja i integracja

Najczęściej na tym właśnie etapie po raz pierwszy i w bardzo dobitny sposób przekonujemy się, ile warta była nasza dotychczasowa praca w naszym przedsięwzięciu – projekcie. I to właśnie w czasie trwania implementacji może dojść do najbardziej zaskakujących zmian w projekcie.

Przykłady takich najbardziej znaczących zmian w ramach omawianych projektów, mających bezpośredni lub pośredni wpływ na budowę UI, przedstawimy w poniższych kategoriach. Analizując projekty pod kątem tych kategorii należy pamiętać o tym, że każdej aktywności – czy to programisty, czy projektanta UI – zmierzającej dla implementacji poszczególnych funkcjonalności, niezbędny jest czas, a wraz z upływem tego czasu zmienia się otoczenie, w jakim osoby je tworzą.

Ograniczenia prawne
Wybierając technologię należy pamiętać o jej ograniczeniach prawnych.

Do ograniczeń tych należy podchodzić analizując różne aspekty produkcji oprogramowania (model dystrybucji, publikacje kodu, dokumentowanie) jak i licencjonowania praw patentowych w zakresie rozwiązań sprzętowych.

W przypadku projektu b-Link wybór licencji docelowej tj. GPL skutkowało wyborem prawie wszystkich narzędzi i komponentów, z jakich miało składać się oprogramowanie b-Link również na tej samej licencji. Licencja GPL jest nazywana „wirusową" i skutkuje tworzeniem oprogramowania całkowicie otwartego, bez prawa powrotu do licencji komercyjnych lub o zamkniętym kodzie. Cóż więc zaskakującego wydarzyło się w projekcie b-Link, co nie zostało przewidziane w obszarze ograniczeń prawnych, a przysporzyło więcej pracy? Po pierwsze, instalator oprogramowania na licencji GPL musiał być udostępniony również na tej samej licencji. Po drugie, konfiguracje komputerów użytkowników okazały się na tyle archaiczne w zakresie technologii „.net", że należało je przed uruchamianiem właściwego oprogramowania uporządkować, instalując odpowiednie wersje ServicePack do systemu Windows. Ponieważ jednak ServicePack nie jest oprogramowaniem rozpowszechnianym z kodem otwartym, a na licencji komercyjnej, pojawiło się nowe wymaganie dla instalatora oprogramowania b-Link – ServicePack nie mógł być jego integralną częścią, a parametrem, który należało

sprawdzić, a następnie na komputerze użytkownika i za jego zgodą pobrać via Internet i uruchomić. W efekcie UI instalatora w niektórych przypadkach jest bardziej skomplikowane, a czas pracy instalatora jest uzależniony od szybkości łącza internetowego, jakim dysponuje użytkownik.

W Calypso sytuacja prawna była odwrotna. Nasze oprogramowanie nie miało prawa korzystać z ani jednej linii kodu, o którym nie było wiadomo ze stuprocentową pewnością, że nie jest to kod na licencji GPL. Ani Philips ani Intertrust, jako firmy bardzo mocno opierające się na patentach i licencjach komercyjnych, nie mogły pozwolić sobie nawet na podejrzenie, że same takie licencje łamią. Jeśli skorzystanie z kodu o nieznanej licencji lub licencji typu GPL byłoby jedynym wyjściem, projekt zostałby zamknięty. Z drugiej strony obie firmy były właścicielami wielu patentów chroniących system Marlin i należało jak najlepiej pokazać jego możliwości.

Podsumowując:

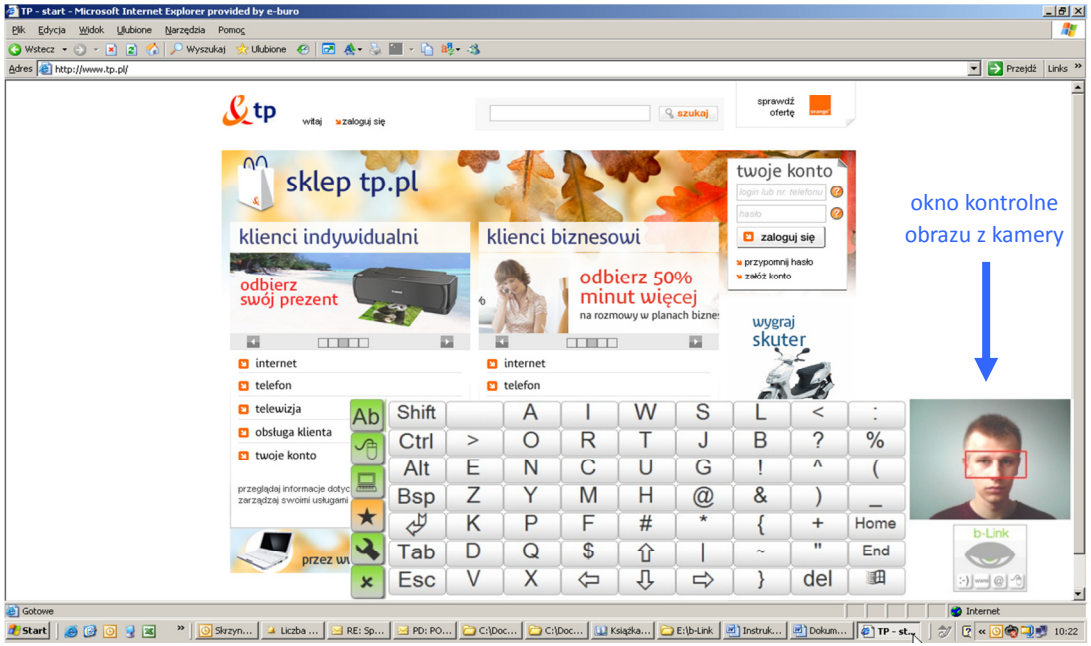
- Ograniczenia prawne projektu mają bardzo duży wpływ na wybór narzędzi, środowisk programistycznych i bibliotek, których można użyć a więc potencjalnie również na wygląd i działanie UI.
- Dodatkowo, przy komercyjnym wdrażaniu nowatorskich systemów, zwłaszcza na rynku amerykańskim, sprawdzenie sytuacji patentowej jest z jednej strony nieodzowne, a z drugiej strony nieznalezienie blokujących nasze rozwiązanie patentów nie gwarantuje w 100 procentach, że taki patent nie istnieje.

Wpływ wybranych technologii na możliwości użytkowe produktu
Decydując się na pewne konkretne rozwiązanie technologiczne rzadko zdajemy sobie, do końca, sprawę z faktycznych ograniczeń dla naszego projektu w tym również dla UI.

W projekcie b-Link została wykorzystana biblioteka OpenCV [5]. Wstępna wersja UI pracowała w oparciu o wersję 1.0 tej biblioteki, jednak w trakcie prac projektowych pojawiła się wersja 1.1, która miała być szybsza od wersji 1.0, ale wraz większą szybkością przetwarzania obrazu wprowadzała kilka nowych błędów w jej pracy. Błędy te zostały oczywiście usunięte kosztem rozszerzenia kodu aplikacji b-Link; w innym przypadku powodowałyby one zawieszanie się UI.

Omawiając temat wersjonowania bibliotek, należy wspomnieć o tak zwanych nieudokumentowanych funkcjonalnościach API. Pisząc program, czy też projektując funkcjonalności UI, który ma być dalej utrzymywana, w żadnym wypadku nie należy wykorzystywać tzw. back-door obecnych w większości API. Nie jest to oczywiście praca projektanta UI, dlatego nie przytaczamy tutaj przykładów, jednak jest to bardzo ważny aspekt pracy architekta oprogramowania, który niejako jest „dziedziczony” przez designera.

Innym bardzo poważnym zagrożeniem obok zmieniających się w trakcie budowy UI bibliotek, jest wydajność samego algorytmu versus możliwości technologicznie komputerów, jakich będzie używała docelowa grupa użytkowników.



Rysunek 3. UI programu b-Link, okno kontrolne obrazu z kamery

W przypadku b-Link kluczową rolę pełniła biblioteka OpenCV, na której algorytmie pracy oparte zostało funkcjonowanie UI. Pierwsza wersja UI zakładała śledzenie twarzy i oczu w czasie rzeczywistym, jednak takie podejście przy komputerach jednoprocesorowych powodowało 100% obciążenie CPU i brak możliwości pracy z innymi aplikacjami, a maszyny z dwurdzeniowym CPU miały obciążenia na poziomie 95-98% z podobnym skutkiem dla aplikacji. Problem musiano rozwiązać przez zmianę prawie 80% kodu programu przechodząc z modelu poszukiwania twarzy i oczu w czasie rzeczywistym na model pracy ze predefiniowanym wzorcem twarzy i oczu użytkownika aktualnie używającego program. Poprawiło to wydajność komputerów z jednym CPU zmniejszając obciążenie do poziomu 90%. Na dalszym etapie zmodyfikowano kod tak, aby zredukować ilość obrazu, z jaką pracuje algorytm. Zmiana ta dała poprawę wydajności o 5%, co nadal okazało się niewystarczające dla komputerów jednoprocesorowych. Dalsza analiza problemu pokazała, że jednym z bardziej zasobożernych elementów UI jest pokazywanie ruchomego obrazu użytkownika w oknie kontrolnym (patrz rysunek 3). Wszystkie te zabiegi mające na celu poprawę wydajności skutkowały następującymi zmianami w interfejsie człowiek-maszyna:

- wyodrębnieniem dwóch faz w użytkowaniu programu: 1) ustalenie wzorca oczu - kalibracja, 2) śledzenie wzorca oczu - praca z programem
- zmianą w sposobie rozumienia UI, nieruchomy obraz w oknie kontrolnym powoduje naturalne pytanie u użytkownika „czy program już pracuje?”

W projekcie Calypso wybór, a właściwie brak możliwości wyboru, platformy sprzętowej miał kolosalny wpływ na projekt interfejsu użytkownika. Prototypowa platforma firmy AMD, która była jedyną spełniającą nasze wymagania sprzętowe została równolegle z naszym projektem zakupiona przez firmę Jobo w celu wprowadzenia na rynek urządzenia pozwalającego profesjonalnym fotografom na gromadzenie i podstawową obróbkę ich fotografii bez potrzeby używania komputera. Jedyną możliwością pozyskania przez nas urządzenia dla naszych użytkowników było zaakceptowanie obudowy, jaką zaprojektowała i produkowała firma Jobo. Na szczęście aplikacja zaplanowana przez Jobo była na tyle bliska naszej, że nie było fundamentalnych problemów, jednak przy projektowaniu interfejsu, jak również interakcji z naszym urządzeniem musieliśmy trzymać się założeń poczynionych przez projektantów Jobo – włącznie z przypisaniem przyciskom funkcjonalności zgodnej z ich etykietami wytłoczonymi na obudowie. Wygląd urządzenia Jobo przestawiony jest na rysunku 4.

Podsumowując:

- Wybór technologii, podobnie jak wcześniej opisane ograniczenia prawne, w dużym stopniu determinuje zarówno wygląd i działanie UI.

Rysunek 4. platforma sprzętowa użyta w projekcie Calypso

Instrukcja użytkownika

Projektując interfejs UI musimy sobie zdawać sprawę z tego, że wszystkie jego elementy powinny ze sobą współgrać. Jeśli robią to harmonijnie, wtedy rola instrukcji użytkownika jest drugo- lub trzecioplanowa. Jeśli zaś elementy te nie tworzą takiej całości bądź tworzą niespotykane dotychczas relacje – jak np. w przypadku przełomowych projektów UI – wtedy instrukcja może stać się kluczem do sukcesu danego interfejsu. Dlatego właśnie projektant powinien również brać udział w budowie instrukcji użytkownika.

Wydaje się, że przykładem takiej „nowej jakości’ w wśród UI może być interfejs programu b-Link. Przekonały nas o tym spotkania z użytkownikami, którym został dedykowany ten program, mianowicie osobom niepełnosprawnym ruchowo. Wielu z nich korzystało dotychczas tylko z interfejsów człowiek-maszyna dla osób sprawnych ruchowo i po nim „odziedziczyło” pewne wzorce interpretacyjne elementów takiego interfejsu. I właśnie w oparciu o te wzorce chciało pracować z programem, nie czytając wcześniej instrukcji

obsługi. Pomocne w tym przypadku okazało się stworzenie trzeciej kolejnej instrukcji obsługi (obok pełnej i drugiej skróconej) w postaci filmu pozwalającego w ciągu niespełna 10 min. zapoznać się z kluczowymi zasadami używania programu b-Link.

Niestety projekt Calypso został, z przyczyn leżących poza nim, zatrzymany na etapie prototypu, pomimo że spełnił wszystkie założenia projektowe. Nie byliśmy w stanie stwierdzić, czy interfejs i zaprojektowana przez nas interakcja były intuicyjne dla użytkowników. Naszą ambicją była taka implementacja, aby instrukcja obsługi nie była potrzebna ale ponieważ koncept na którym opierał się Marlin był nowatorski, nie mieliśmy pewności, że informacje zawarte w interfejsie będą wystarczające do zrozumienia jego działania.

Podsumowując:

- Z perspektywy obu projektów wynika, że dużym wyzwaniem dla projektanta UI jest znalezienie kompromisu pomiędzy ilością informacji przekazywanej początkującemu użytkownikowi przez interfejs a łatwości i szybkości interakcji jakiej wymaga użytkownik zaawansowany.
- Na jakość instrukcji obsługi wpływa zarówno jej zawartość jak i forma.

Integracja

Integracja jest bardzo istotnym elementem implementacji rozwiązania nie tylko z punktu widzenia dodawania i testowania nowej funkcjonalności; często ma też negatywny wpływ na szybkość reakcji UI. Dlatego integrację należy przeprowadzać dostatecznie często, równocześnie weryfikując jakość działania UI.

W projekcie b-Link do pewnego momentu UI i silnik aplikacji odpowiadający za detekcję i interpretacje mrugnięć powiekami były programowane niezależnie. Stała kontrola tego, co robili programiści ze swoim częściami kodu oraz praca w jednym zespole pozwoliła uniknąć jakichkolwiek problemów związanych z integracją.

Jednym z podstawowych założeń praktyk agile jest ciągła integracja kodu i bardzo częste etapy pośrednie, w których można zademonstrować działanie całego rozwiązania nawet jeśli część funkcjonalności jest tylko zasymulowana. W praktyce jest to chyba najważniejszy fundament tych technik i z niego wynikają największe oszczędności czasowe. Niestety w przypadku Calypso nie było możliwe dokładne przestrzeganie tej zasady. Część zespołu pracująca w USA implementowała swoją część funkcjonalności w oderwaniu od reszty systemu. Argumentem za takim postępowaniem był fakt, że ta funkcjonalność rzeczywiście była oderwana od reszty kodu (w tym od UI) do tego stopnia, że tworzona była w innym języku programowania. W oczekiwaniu na ten kod zespół w Europie wstawił w jego miejsce „zaślepkę” i nikt nie przeczuwał problemów. Niestety w momencie integracji okazało się, że kod zza oceanu nie zachowuje się tak jak tego spodziewał się zespół z Europy. Miało to bardzo duży wpływ na działanie całego systemu – w tym, niespodziewanie, również na UI. Okazało się, że po integracji system zaczął działać na tyle wolniej, że zachodziła obawa, że trzeba będzie przeprojektować interakcję, tj. zmienić kolejność działań zabierających więcej czasu tak, aby użytkownik nie musiał na nie czekać albo tak, żeby mogły toczyć się równolegle z innymi akcjami użytkownika. Potencjalnie mogło to mieć katastrofalny wpływ na interakcję, gdyż po zmianie kolejności

działań i stawała się ona mniej intuicyjna. Ostatecznie udało się uniknąć zmian w interfejsie ale późna integracja została okupiona dużym dodatkowym wysiłkiem.

Podsumowując:

- Z doświadczenia obu projektów mimo, że prowadzone były innymi metodami wynika, że częsta integracja poszczególnych funkcjonalności systemu oraz ciągłe utrzymywanie jego działającej wersji nadającej się do demonstracji jest kluczowa dla zmaksymalizowania szansy powodzenia projektu.

Nieprzewidziane zmiany
Często źródłem nieprzewidzianych zmian w projekcie są jego przeglądy a w czasie przeglądów najbardziej widoczną częścią demonstracji jest interfejs użytkownika. W zależności od przyjętej metodologii, przeglądów w procesie projektowym może być więcej lub mniej, ale mało który sponsor nie będzie chciał raz na jakiś czas przekonać się jak zaawansowany jest jego projekt.

Znając wagę tego komunikacji z interesariuszami, oprogramowanie b-Link było gotowe do demonstracji zlecającemu po każdej tygodniowej puli zmian wprowadzonych do kodu i UI.

W przypadku Calypso przeglądy również nie były problemem gdyż w naszym procesie przegląd wewnętrzny odbywał się co 2 tygodnie i zawsze połączony był z prezentacją działania całego systemu. W ten sposób zawsze byliśmy automatycznie gotowi na przeglądy osób spoza projektu. Niespodzianka nastąpiła podczas wizyty pewnego wysoko postawionego dyrektora, który po obejrzeniu działania naszego prototypu zapytał skąd ma wiedzieć co dzieje się w środku. Było to dla nas niespodzianką bo nigdy nie pomyśleliśmy o konieczności demonstracji interakcji zachodzących pomiędzy wewnętrznymi komponentami rozwiązania — technologia miała być ukryta przed użytkownikami. Tu okazało się jednak, że musimy zbudować nowy „demonstracyjny interfejs użytkownika" dla osoby, która wprawdzie nie będzie faktycznym użytkownikiem naszego urządzenia, ale od której zależy decyzja o kontynuacji naszych działań, a także końcowa ocena sukcesu projektu. Interfejs taki został zaprojektowany i zaimplementowany i okazał się później bardzo pomocny przy demonstracji koncepcji działania systemu.

Oczywiście w większości przypadków końcowy produkt nie służy demonstracji technologii, na której został oparty, jednak już na początku projektu, w fazie definiowania założeń funkcjonalnych, należy się zastanowić nad wszystkimi interesariuszami projektu i świadomie podjąć decyzję, czy nie należy dodać czegoś do interfejsu użytkownika pod ich kątem. O sukcesie naszego projektu decyduje bowiem całkiem spora grupa ludzi, zanim w ogóle nasz produkt będą mieli szansę zobaczyć końcowi użytkownicy. W praktyce najczęściej taka sytuacja zachodzi gdy nasz klient (czyli osoba podejmująca decyzję o zakupie) nie jest końcowym użytkownikiem naszego produktu.

Podsumowując:

- Nie można zabezpieczyć się przed nieprzewidzianymi zdarzeniami ale można zminimalizować ich wpływ na projekt. Najważniejszą, według nas, ochroną jest wspomniana w poprzednim rozdziale częsta integracja, całkowite panowanie nad kodem oraz używanym sprzętem oraz utrzymywanie kolejnych stabilnie

działających wersji systemu, które można zademonstrować interesariuszom.

Testy

Testy z użytkownikami to taki moment pracy nad projektem, a raczej z produktem tego projektu, kiedy powinniśmy dysponować już bardzo zaawansowanym lub w pełni funkcjonalnym rozwiązaniem UI, najlepiej jednak całego rozwiązania. To właśnie na tym etapie praca projektanta interfejsu jest podlega najsurowszej z ocen – tym surowszej, im mniej było prób prototypowania UI wykonanych przed jego implementacją.

Dla b-Linka faza ta oznaczała testy aplikacji u użytkownika w domu, jako że osoby niepełnosprawne ruchowo nie mają wystarczającej swobody poruszania się, aby można było takie testy zrealizować w przygotowanych do tego pomieszczeniach, gdzie np. zachowania użytkownika można zapisać na taśmie do dalszej analizy. Jednocześnie taki wybór miejsca testów ma też dobre strony, np. pozwala zaobserwować użytkownika w otoczeniu, w jakim będzie on pracował na co dzień.

Ocenie podlegał cały pakiet, tj. program instalacyjny wraz z w pełni funkcjonującą aplikacją oraz instrukcjami obsługi (krótką, długą i filmową). Problemy, które zostały zidentyfikowane to:

- Konieczność poprawienia instalatora, w którym nie przewidziano ustalania innego niż standardowo konfigurowanych przez system praw dostępu do katalogów systemowych zawierających komponenty, jakich wymagała aplikacja
- oraz rzecz bardzo ważna z punktu widzenia UI, mianowicie luka koncepcyjna w sposobie korzystania z programu przez docelowego użytkownika.

Ta luka koncepcyjna ujawniła się, gdy jeden z użytkowników był zmuszony do leżenia na boku i obserwowania ekranu monitora pod kątem 90° a nie jak wyobraził to sobie projektant UI, tj. siedząc bądź leżąc przed monitorem na plecach (zawsze w orientacji pionowej). Algorytm pracujący z obrazem z kamery zakładał zawsze tylko jedną orientację twarzy i oczu użytkownika Na rynku jest zaskakująco niewiele kamer, które mogą być zawieszone z boku monitora lub obracane tak, aby dostosować je do orientacji użytkownika. Lukę usunięto, wprowadzając dodatkowy przełącznik w linii komend, który może być dodawany do skrótu uruchamiania programu, jeśli użytkownik będzie chciał pracować w orientacji poziomej.

Jak już pisaliśmy wcześniej, projekt Calypso nie doszedł do fazy testów z użytkownikami, jednak projekt interfejsu nie opierał się jedynie na pomysłach inżynierów wchodzących w skład zespołu. Ponieważ zespół nie mógł włączyć do swojego składu projektanta na stałe, zastosowaliśmy pewien trick. Po ustaleniu, na jakiej platformie zostanie zaimplementowany nasz odtwarzacz, zespół we własnym zakresie zaprojektował i uruchomił działającą na komputerze PC makietę interfejsu. Makieta była w pełni funkcjonalna i zachowywała się tak, jak miał zachowywać się końcowy produkt. W następnym etapie został wynajęty zewnętrzny projektant, któremu pokazaliśmy urządzenie i w ciągu kilku spotkań wyjaśniliśmy, co system ma robić. Po przejściu całego cyklu projektowania we własnym zakresie projektant pokazał nam swoją koncepcję interfejsu i interakcji, którą mogliśmy porównać z naszą. Ponieważ obie koncepcje okazały się w zasadzie identyczne,

pozwoliliśmy sobie założyć, że rozwiązanie było poprawne i należy je zaimplementować.

Podsumowując:

- Złą wydaje się praktyka wykonywaniu testów z użytkownikami tylko dla UI, ponieważ naszym zdaniem taki test może nie uwzględniać zachowania się reszty rozwiązania o ile nie zostanie ono zasymulowane np. przez wprowadzenie przewidywanych opóźnień.
- Przedmiotem pracy projektanta UI jest nie tylko końcowy produkt ale również wszystko co jest niezbędne aby użytkownik mógł tego produktu użyć. W tym instalator z własnym UI, czy wspomniane już wcześniej instrukcja obsługi i programy pomocnicze.
- Testy z użytkownikami potrafią ujawnić bardzo zaskakujące luki koncepcyjne, najczęściej wynikające z przyjętych zbyt wąskich kryteriów przed projektowaniem UI.

Najlepsze praktyki dla projektantów UI

Nasze wskazówki ściśle dla projektantów UI, w odróżnieniu od dobrych praktyk w projekcie w ogóle, można streścić w trzech kategoriach.

Po pierwsze, projektant UI powinien być zaangażowany w projekt na jak najwcześniejszym etapie jego powstawania. Wczesne prototypy czy makiety UI pozwalają na lepszą komunikację z interesariuszami i sponsorami na etapie uruchamiania projektu. W fazach definiowania wymagań technicznych i funkcjonalnych projektant powinien mieć wpływ na wybór technologii, na których oparty zostanie interfejs użytkownika jak również powinien być świadomy ograniczeń, które nałożą na niego wybory innych członków projektu.

Po drugie, wdrożenia komercyjne wprowadzają wiele ograniczeń, których projektant niekoniecznie musi doświadczyć w innych warunkach. W szczególności, zarówno małe firmy jak i uczelnie są mniej narażone na skutki uregulowań prawnych. Projekt, który odnosi spektakularny sukces rynkowy będzie przedmiotem dociekań prawników zarówno niezależnych jak również pracujących dla konkurencji, których celem będzie storpedowanie naszego sukcesu lub zmuszenie nas do wykupienia różnego rodzaju licencji. Dlatego jest niesłychanie ważne aby sprawdzać na jakiej licencji udostępniane są narzędzia i biblioteki, których chcemy używać oraz sprawdzenie sytuacji patentowej.

Po trzecie, o sukcesie projektu komercyjnego decyduje całość wrażenia jakie pozostawimy na użytkowniku. Nawet najlepszy produkt nie odniesie sukcesu jeśli nie zadbamy o każdy krok, z którym zderzy się nasz klient począwszy od zapoznania się z propozycją, poprzez ewentualną instalację produktu, nauczenie się funkcjonalności, regularne używanie (również gdy coś nie wychodzi) aż do fazy użytkowania jako ekspert.

Literatura

[1] Ocena stanu psychofizycznego osoby na podstawie analizy dynamiki mrugania, mgr inż. Aleksandra Królak, 2009
[2] PMI (Project Management Institute), www.pmi.org
[3] A Guide to the Project Management Body of Knowledge, 3th Edition
[4] GPL (General Public License), www.gnu.org/licenses/gpl.html
[5] OpenCV (Open Computer Vision Library), sourceforge.net/projects/opencvlibrary

[6] Marlin Developer Community
www.marlin-community.com/technology
[7] Agile software development
en.wikipedia.org/wiki/Agile_software_development
[8] Extreme programming
www.extremeprogramming.org