Marta Bartnicka: koordynatorka tłumaczeń i lokalizacji — od roku w Dolby Laboratories, a wcześniej w IBM, gdzie przeszła ścieżkę od tłumacza i testera poprzez project management po kierowanie Translation Center. Inżynier informatyk. Kilka razy w roku prowadzi – we współpracy z firmą Localize.pl – szkolenia z lokalizacji oprogramowania i z zastosowania tłumaczeń maszynowych w praktyce. Współautorka książki „Programiści i tłumacze". Hobby w kolejności alfabetycznej: biegówki, córki, ogrodnictwo, Tatry.
Ewa Dacko: tłumaczka specjalizująca się w IT i nowych technologiach, inżynier od zarządzania, specjalistka od marketingu, grafik i projektant. Pasjonatka UX i zastosowania tej koncepcji w każdym aspekcie życia. Szkoli tłumaczy w dziedzinie UX, a projektantów w dziedzinie lokalizacji. Nie twierdzi może, że UX jest odpowiedzią na wszystkie problemy, ale gorąco wierzy, że pomaga rozwiązać wiele z nich. Prywatnie instruktorka żeglarstwa, sternik morski i mama nastolatki. Ma dwa koty.

KONTAKT: http://localization.pl

O czym **nie będziemy** mówić

t:f

Większości ludzi (nawet/zwłaszcza z IT) pojęcie „lokalizacja” kojarzy się — i zasadniczo słusznie — z lokalizacją geograficzną, fizycznym miejscem, geocachingiem, GPS-em, nawigacją, mapami.
Tematem prezentacji jest jednak „inna” lokalizacja: lokalizacja produktów (nie tylko) cyfrowych: aplikacji, stron, usług itp.

O czym **będziemy** mówić

t:f

Lokalizacja:

dostosowanie obiektu do innego środowiska kulturowego

- zachowanie najistotniejszych cech obiektu
- zastąpienie cech nieprzekładalnych odpowiednikami zrozumiałymi dla odbiorcy
- uwzględnienie wymagań technicznych i prawnych

ALFABET
LICZBY, DATY
NORMY
PRZEPISY

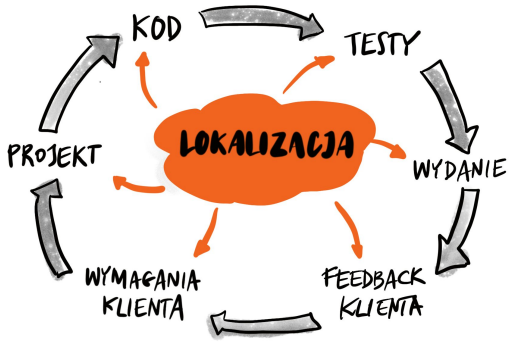
Tak rozumiana lokalizacja to **dostosowanie czegoś (aplikacji, produktu, usługi) do innego środowiska kulturowego**.
Ważne są tu trzy aspekty:

1. zachowanie najistotniejszych cech tego czegoś (aplikacja do obróbki zdjęć pozostaje aplikacją do obróbki zdjęć, samochód pozostaje samochodem)
2. zastąpienie cech nieprzekładalnych (obcych, niezrozumiałych, kojarzących się inaczej) odpowiednikami zrozumiałymi dla odbiorcy
3. uwzględnienie wymagań technicznych i prawnych (alfabet, sposób zapisu liczb i dat, waluty, strefa czasowa, normy, przepisy)

Przykłady lokalizacji: nie tylko aplikacja z przetłumaczonym UI, ale też różne wersje serwisu WWW (grafika, kolorystyka, zdjęcia użytkowników), samochód z kierownicą po prawej stronie, informacje o składzie produktów albo przetwarzaniu danych osobowych (RODO), program do fakturowania zgodny z przepisami i systemem podatkowym danego kraju (np. płatności online w Korei Południowej, gdzie przepis z 1999 r. wymagał stosowania wyłącznie IE z ActiveX; źródło: *Leading Quality*, Ronald Cummings-John i Owais Peer)

Po co lokalizować i **kto** to robi?

t:f



Po co lokalizować?

Żeby **sprzedać** lub **umożliwić używanie**

Kto to robi?

Projektanci + development + tłumacze + testerzy

- **Cel lokalizacji:** sprzedać produkt większej liczbie osób albo umożliwić używanie go większemu gronu użytkowników (najczęściej po prostu: więcej zarobić).
- Lokalizacja wpływa na **każdy etap tworzenia produktu** i trzeba ją brać pod uwagę od samego początku (czyli ustalania już w momencie wymagań i projektowania).
- W proces lokalizacji zaangażowani są **wszyscy uczestnicy tego procesu:** projektanci, programiści, tłumacze (których specjalizacja zależy od złożoności produktu), testerzy (i wszyscy muszą się komunikować — to warunek konieczny sukcesu!)

test:fest
Wrocław Testing Conference 2020

5 przykazań

przygotowania produktu

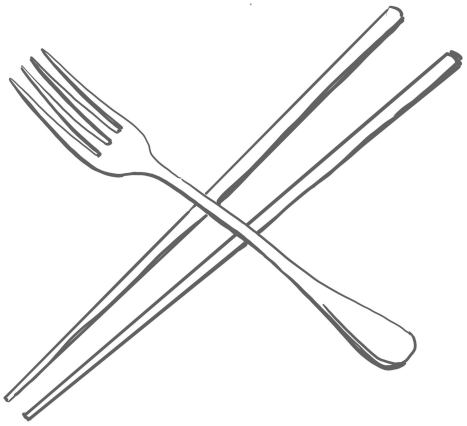
Lokalizację trzeba brać pod uwagę na każdym etapie pracy nad produktem. Nawet najlepiej przygotowane i przeprowadzone testy niewiele pomogą, jeśli produkt jest nieprzemyślany i zrobiony źle, dlatego pierwszych 5 przykazań dotyczy przygotowania produktu (a kolejnych 5 — testowania lokalizacji).

1. Uwzględnij różnice UX i funkcjonalne

t:f

Kluczowe pytania

- **Co** chcemy zrobić?
- Dla **kogo**?
- **Jak** (i dlaczego)?



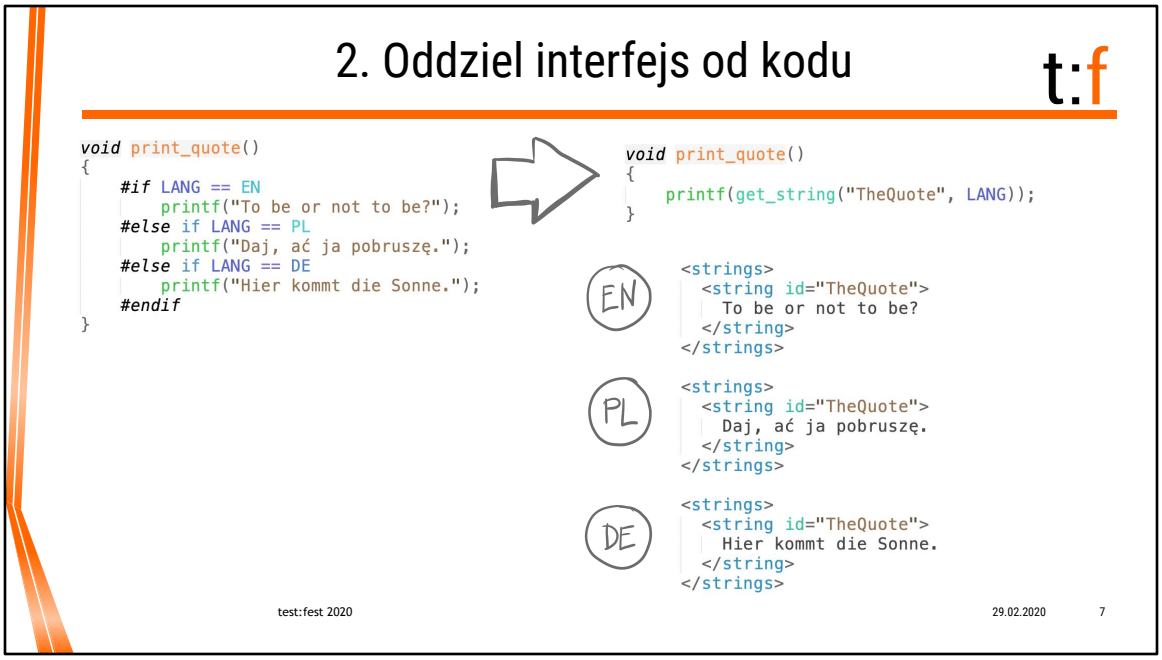
Kluczowe pytania, na które trzeba tu sobie odpowiedzieć: **CO** chcemy zrobić? **DLA KOGO**? **JAK**?
Trzeba wziąć pod uwagę **nie tylko** to, że nasz odbiorca mówi innym językiem (nie wystarczy tłumaczenie). Różnice mogą być ogromne (kulturowe, obyczajowe, prawne…) i trzeba je uwzględnić już przy projektowaniu wymagań.
Jeśli na tym etapie coś pójdzie źle albo coś się zaniedba, nie ma szans, żeby kolejne etapy się zakończyły powodzeniem.

Jak to zbadać?

- Analiza przepisów, specyfikacji...
- Metody UX-owe: badania potrzeb użytkowników, **lokalne persony**.

Wyniki tych badań i analiz mogą nas bardzo zaskoczyć: może się okazać, że trzeba zrobić **zupełnie inną wersję aplikacji**... albo że ogóle lokalizowanie danego produktu nie ma sensu.

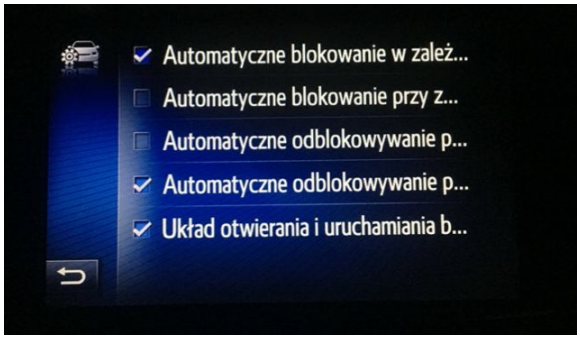
PRZYKŁAD: Aplikacja słabo sprzedawała się w Indonezji; potencjalni użytkownicy „odpadali" przy rejestracji na etapie podania nazwiska. Współpraca z lokalnymi testerami pozwoliła wyjaśnić, że około 40% Indonezyjczyków nie ma nazwiska, a mniej wyrobieni technicznie nie wpadną na wpisanie w obowiązkowe pole byle czego. Dopiero usunięcie wymogu podania nazwiska pozwoliło na dotarcie do rynku z 262 milionami osób. (źródło: *Leading Quality*, Ronald Cummings-John i Owais Peer).



Póki interfejs użytkownika pisany jest pod jeden język, zwykle kusi, żeby teksty na UI generować prosto z kodu (a w każdym razie te teksty, które się dopisuje w ostatniej chwili). Kiedy dodaje się 1-2 języki, to też kusi, żeby to zaszyć w kodzie („bo na pewno nie będzie więcej”).
Lepiej od razu trzymać zasoby UI osobno, a wszelkie teksty i przetwarzanie zależne od języka wydzielać.

Źródło przykładu: *User Interface Good for Every Place*, Paweł Pluta (Espotel/Etteplan), October 3, 2014

3. Weź pod uwagę inny układ tekstu



Przykład **tekstu niemieszczącego się na ekranie**: interfejs samochodu (Toyota RAV4)

PROBLEM 1: teksty przetłumaczone mogą być dłuższe niż oryginał (w tłumaczeniu z angielskiego na polski: średnio 30%).
PROBLEM 2: po przetłumaczeniu informacja kluczowa (różne sposoby blokowania drzwi w samochodzie) znalazła się na końcu.
SKUTEK: interfejs jest nie tylko nieestetyczny, ale też **nie spełnia swojej funkcji** - nie wiadomo, jak blokować/odblokować drzwi w samochodzie. Na co dzień niewygodne — w sytuacji awaryjnej niebezpieczne.
PRZYCZYNA: nie uwzględniono, że tłumaczenie może być dłuższe; **nie było testów (albo były niewłaściwe).**

4. Rozumnie używaj konkatenacji i zmiennych

t:f

Tego nie tłumaczył robot! Ekran powstał przez wstawienie tłumaczenia “Cucumber” (zapewne była lista różnych innych upraw) i stworzenie liczby mnogiej przez dodanie “s”. O, tak: {0}s :) Stanowczo nie kleimy słów z kawałków. Działa w angielskim i chińskim, w pozostałych NIE DZIAŁA.

5. Kontekst daj tłumaczowi!

t:f

{0} inflicts {1} to your resources and decreases your HP by {2} for every shop you visit.

{Event.Saint.Nicholas.Day} inflicts {Wallet.damage.points} to your resources and decreases your HP by {HP.points} for every shop you visit.

10

Zdanie w pierwszej wersji trzeba tłumaczyć w ciemno. Dopiero drugi wariant daje tłumaczowi szansę „opakować” zmienne w poprawne zdanie.
Jeśli nie możemy udostępnić całej aplikacji ani zrzutów ekranów oryginału - zadbajmy, żeby do tłumacza trafiły **komentarze albo przynajmniej znaczące nazwy zmiennych** (są na to różne techniki zależne od narzędzia, w jakim pisze się UI / przechowuje się zasoby).
Jeśli praca odbywa się w trybie continuous delivery i część interfejsu została już przetłumaczona, udostępnijmy również te tłumaczenia.

Jeśli ktoś sugeruje, że przygotowanie UI do lokalizacji wygląda tak:

- wrzucić do CSV/Excela same teksty bez komentarzy i identyfikatorów,
- wyciąć duplikaty (po co płacić dwa razu za tłumaczenie tego samego),
- posortować alfabetycznie,
- wysłać do tłumaczy,

...to najgorszy możliwy scenariusz :(

Źródło przykładu: konkurs dla tłumaczy gier, Gabriela Janiszewska, grudzień 2019

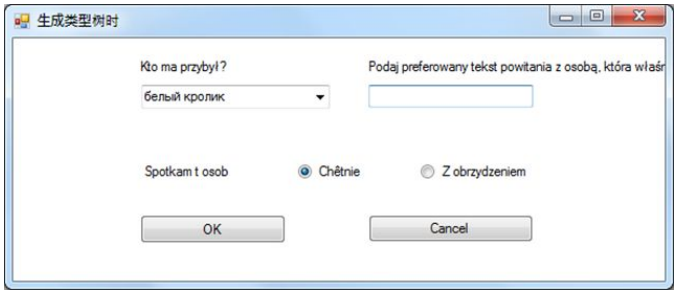
test:fest

Wrocław Testing Conference 2020

5 przykazań

testowania produktu

6. Zaplanuj testy



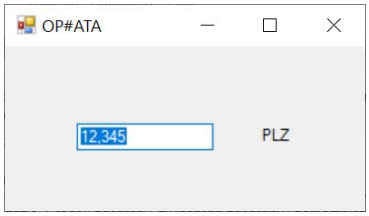
Przykład:

- Na tym ekranie możemy sprawdzić, czy etykiety są we właściwym języku, mają znaki narodowe i czy się mieszczą (oraz czy są w miarę sensowne, o ile znamy polski).
- Nie sprawdzimy, czy można wprowadzać dane po polsku. Nie sprawdzimy, czy opcje robią to, co jest napisane, że robią.
- Poprawki do tłumaczenia trzeba sprawdzić w następnej iteracji.

Ogólnie:

- Testy funkcjonalne lokalizacji powinny się wydarzyć przed testami tłumaczenia.
- Plan testów będzie zależał od tego, jaka jest aplikacja: web UI bez wprowadzania danych — mniej testowania; na specyficznych urządzeniach z dużą ilością wprowadzania danych przez użytkowników — więcej testowania.
- Dobre praktyki wymieniane w książce *Leading Quality*: testowanie na miejscu w rzeczywistych warunkach, wykorzystanie lokalnych person/użytkowników, współpraca z partnerami zewnętrznymi.

7. Rozróżnij błędy funkcjonalne i tłumaczeniowe

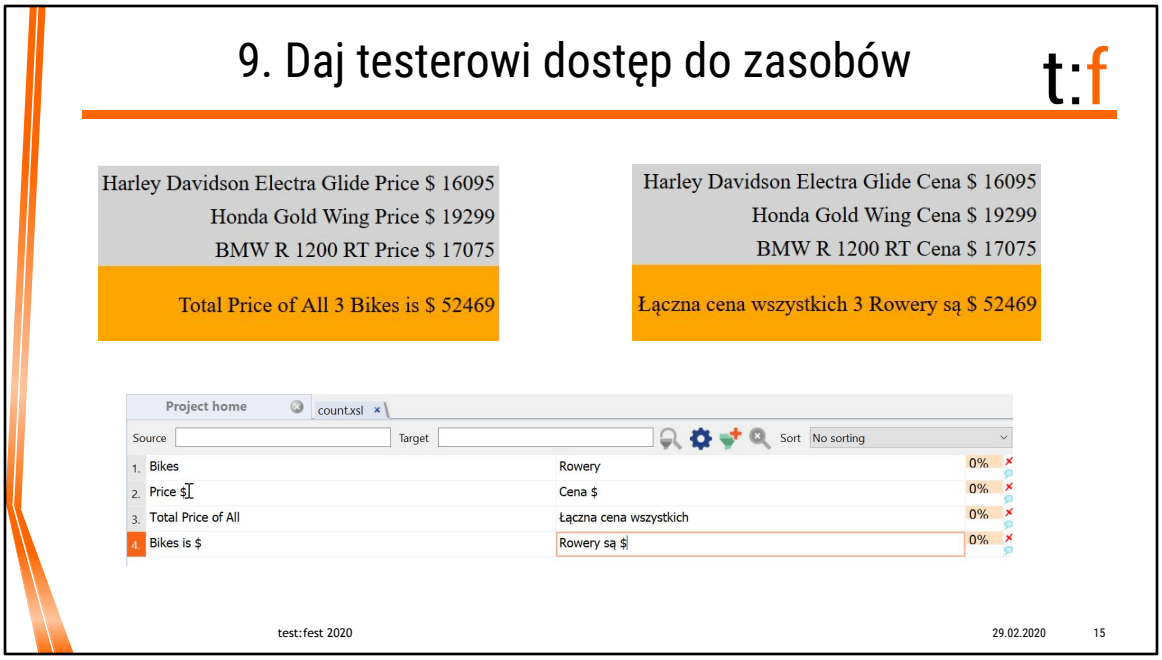


W raporcie z testów konieczne jest rozróżnienie błędów funkcjonalnych (format kwoty) od tłumaczeniowych (PLZ zamiast PLN). To jest istotne, bo kto inny będzie je poprawiał (funkcjonalne developer, tłumaczeniowe i językowe — tłumacz).

8. Native is king

t:f

- Przy testowaniu zlokalizowanego interfejsu osoba niewładająca danym językiem może niewiele więcej niż automat. Tylko native speaker (rodzimy użytkownik języka) zauważy niewykrywalne literówki, użycie słów niezgodnie z kontekstem albo uzusem („tak się po prostu nie mówi”).
- Tester lokalizacji wybrany wg klucza „bo znał(a) polski” potrzebuje **dobrego scenariusza**, bo niekoniecznie zna produkt.



Przykład:

- Tester widzi brzydką, niezbyt zrozumiałą wersję polską z zaszytym dolarem. Wersja angielska jest OK, więc zagląda do źródła.
- W źródle widać, że tekst jest a) klejony z kawałków b) bez informacji o kontekście, a c) znak dolara jest zaszyty na sztywno.

Tester nie musi poprawiać błędów tłumaczenia (choć jest taki model pracy), ale powinien widzieć, co było w oryginale i jakie są ograniczenia w tłumaczeniu (np. tekst w kawałkach). Na tej podstawie może dać sensowne zalecenia tłumaczowi.
Jeśli czas pozwala (a rzadko kiedy pozwala na tym etapie), tester może dać zalecenie przeprojektowania tekstów.

Przykład:
3 komunikaty — 3 różne tłumaczenia słowa „expired”. Powinno to zostać uspójnione na etapie tłumaczenia, ale nie zostało (praca podzielona na wielu tłumaczy, brak uzgodnienia słownika, dotłumaczanie w kolejnych wersjach bez przekazania słownika z wcześniejszych).
Tester nie musi decydować, jaki termin jest poprawny, ale zgłasza go do uspójnienia przez tłumaczy.

Tak samo należy **zadbać o zgodność UI z UA** (dokumentacją, pomocą) — genialny pomysł, że interfejs i pomoc tłumaczą dwa różne zespoły, trzyma się twardo na rynku od ćwierć wieku :(

test:fest

Wrocław Testing Conference 2020

Przykłady

że da się to zrobić dobrze

Lokalizacja jest jak kanalizacja

t:f

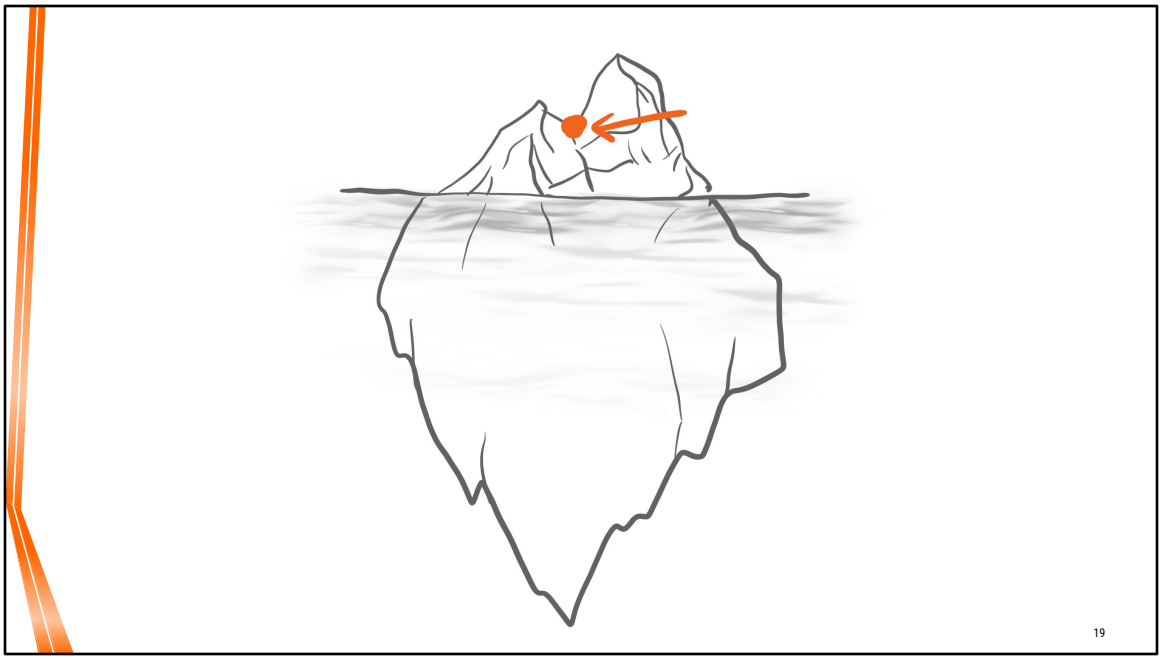
Evernote

test:fest 2020 29.02.2020 18

Lokalizacja jest jak kanalizacja — nikt o niej nie mówi, dopóki działa. Sukcesy lokalizacji uważa się za coś oczywistego, publikuje i nagłaśnia się znacznie więcej ciekawych przykładów porażek.

Przykłady udanej lokalizacji:

1. **Biletomaty Wrocław**
   - więcej ludzi kupuje bilety, mniej pasażerów na gapę
   - skutki społeczne i wizerunkowe (co 10 wrocławianin jest Ukraińcem, b. dużo turystów)
2. **Evernote** — ponad 4 mln użytkowników w Chinach w ciągu roku od premiery
   - nie tylko przetłumaczony interfejs, ale też:
   - inna nazwa na chiński rynek (Yinxiang Biji)
   - inne funkcje (np. synchronizacja tylko po wi-fi ze względu na drogą transmisję w 3G, integracja z WeChat)
   - dużo testów z lokalnymi użytkownikami



Do zapamiętania

t:f

localization.pl

Najważniejsze rzeczy do zapamiętania:

1. Lokalizacja to nie geocaching i nie tylko tłumaczenie.
2. Trzeba ją uwzględniać w całym procesie tworzenia oprogramowania, od samego początku.
3. W testy warto włączyć osoby znające docelowy język i kulturę.
4. Więcej informacji, linki do szkoleń itp. można znaleźć na stronie **localization.pl**


test:fest
Wrocław Testing Conference 2020
DOLBY

-