

Pamiętam jak kiedyś, w studenckich czasach, prowadziłem spływ kajakowy. Mielśmy na nim stary, zardzewiały, opalony setkami ognisk grill, na którym smażyliśmy to, co udało się kupić w miejscowych sklepikach. Przed wyruszeniem w kolejny etap spływu, trzeba było optukać go w wodzie i przetrzeć piaskiem, żeby nie ubrudzić sadzą wszystkiego w kajaku. W czasie pierwszego biwaku, zwróciłem się do jednego z uczestników spływu: „Czy mógłbyś umyć grill?”. Chłopak zniknął w zaroślach nad wodą, nie było go chyba ze dwie godziny, po czym wrócił niosąc coś błyszczącego w rękach. To był nasz grill, wyszorowany do połysku...

Nie ukrywam, że zrobiło mi się głupio. Ktoś odwalił kawał ciężkiej, solidnej, i absolutnie niepotrzebnej roboty, tylko dlatego że nie przekazałem mu odpowiedniej informacji. Nie wziąłem pod uwagę, że chłopak był pierwszy raz na spływie, i być może nigdy wcześniej nie korzystał z takiego urządzenia. Dla mnie sposób „codziennej konserwacji grilla” był czymś trywialnym i oczywistym, on miał prawo niczego o tym nie wiedzieć.

Od tego spływu minęło kilkanaście lat, a jednak tamta sytuacja regularnie mi się przypomina. Przychodzi mi na myśl, kiedy widzę jak ktoś, pełen najlepszych chęci, wykonuje kawał dobrej roboty, która ostatecznie okazuje się do niczego niepotrzebna. Dlatego że od początku chodziło o coś innego, a rozbieżność powstała w momencie, kiedy jakiś współpracownik, bynajmniej nie ze złej woli, przekazał niewłaściwą informację, lub przekazał informację w niewłaściwym sposób.

Tego typu sytuacje mogą powstać wszędzie tam, gdzie praca wykonywana przez jedną osobę, opiera się na informacjach dostarczanych przez inną osobę. Klasycznym przykładem jest programista, który poprawia błędy w programie, na podstawie informacji otrzymywanych od testera. Jeśli tester pomylił się dokumentując znaleziony błąd, programista może spędzić długie godziny, szukając usterek w poprawnym fragmencie kodu. Jeśli opis okoliczności wystąpienia błędu nie będzie dostatecznie precyzyjny, programista będzie miał większy kłopot ze zlokalizowaniem usterki.

Błędy znalezione w oprogramowaniu, można dokumentować na różne sposoby, używając różnych narzędzi. Istnieje jednak kilka uniwersalnych cech, niezależnych od techniki prowadzenia dokumentacji, które decydują o jakości opisu błędu. To właśnie od nich zależy, czy programista działając na podstawie takiego opisu, „w ciągu dwóch minut doprowadzi grill do właściwego stanu”, czy też przez długi czas będzie wykonywał równie ciężką co nieefektywną pracę.

Poniżej zamieszczam listę cech, które uważam za kluczowe dla porządnego opisu błędu. Wszystkie przykłady w tekście, pochodzą z jednego z projektów, w których brałem udział.

Cechy dobrego opisu błędu.

1. Nie zawiera błędów.

Po sporządzeniu opisu błędu, zawsze należy bardzo dokładnie przeczytać to, co się napisało, i upewnić się że nie ma tam pomyłek. Pomyłki w opisie błędu mogą spowodować stratę czasu przez programistę, który starając się odnaleźć przyczynę błędu, będzie szedł fałszywym tropem.

2. Jest użyteczny nie tylko dla programistów.

Oprócz długiego opisu błędu, przeznaczonego dla programisty, należy również sporządzić krótki opis, który umożliwi szybkie zorientowanie się o co ogólnie chodzi. Warto podać również kategorię (funkcjonalność / moduł / fragment programu), której dotyczy ten błąd. Oczywiście, błędowi należy też przypisać określony priorytet. Wszystkie te informacje (krótki opis, długi opis, kategoria, priorytet) powinny się znaleźć w oddzielnych polach listy błędów. Dzięki temu, taki dokument będzie pożyteczny zarówno dla programisty tworzącego kod, jak i dla kierownika projektu, który będzie mógł szybko zorientować się w jakim stanie jest program, bez potrzeby wnikania w szczegóły techniczne.

3. Opisuje tylko jeden, konkretny błąd.

Czasem testera może korcić, żeby dla zaoszczędzenia czasu i wysiłku, zawrzeć informacje o kilku błędach w jednym opisie. Jest to zła praktyka, która przysparza problemów. Pierwszy kłopot ma programista, który np. poprawi tylko jeden z tych błędów, i nie wie czy całość ma oznaczyć jako poprawioną, czy zostawić jako otwartą, czy może umieścić jakąś adnotację. Następny problem ma tester, który zweryfikuje że nie wszystko zostało poprawione, i musi umieszczać dodatkowe wyjaśnienia, żeby było wiadomo co działa, a co nie. Dlatego każdy opis powinien dotyczyć pojedynczego, konkretnego błędu – to po prostu ułatwia życie.

4. Podaje jednoznaczny sposób reprodukcji błędu.

Opis w rodzaju *błąd taki a taki przy dodawaniu nowego klienta* nadaje się do pola *krótki opis* listy błędów, ale jako *długi opis* można go użyć tylko w przypadku, gdy ten błąd występuje bez względu na to jak wchodzimy w opcję dodawania klientów, bez względu na to jakie czynności po drodze wykonujemy, i bez względu na to jakie dane próbujemy wprowadzić. Jeśli w jakichś okolicznościach błąd nie występuje (lub występuje inny błąd), wtedy taki opis jest już niewystarczający – trzeba dokładnie podać, krok po kroku, jakie czynności należy wykonać, żeby dany błąd uzyskać.

Przykład.

Kategoria: Korelacje.

Krótki opis: Korelacje nie są obliczane w pewnej sytuacji.

Długi opis: Scenariusz prowadzący do sytuacji, w której korelacje nie są obliczane: Wybierz kryteria dla pierwszego wykresu, kliknij 'R', wybierz produkt (wykres zostaje wyświetlony), wybierz 'Obszar' tylko dla drugiego wykresu, kliknij 'R', wybierz produkt (wykres nie jest wyświetlany), wybierz 'Miarę' dla drugiego wykresu, kliknij 'R', wybierz produkt (wykres zostaje wyświetlony), kliknij 'Korelacje'. Korelacje nie zostają obliczone.

5. Dokładnie informuje, jaki to błąd.

Opis błędu powinien wystarczyć do zrozumienia jaki to błąd, bez konieczności uruchamiania programu i wykonywania reprodukcji. Powód jest oczywisty – oszczędność czasu czytającego. Dlatego informacje typu numer błędu, czy wyświetlona na ekranie wiadomość o błędzie, należy zawrzeć w opisie.

Przykład.

Kategoria: Wiele okien z wykresami.

Krótki opis: Przy zamykaniu okien w pewnej kolejności, występują błędy.

Długi opis: Wykonałem następujące czynności:
Wybrałem bazę danych.
W oknie 'wybór danych' wybrałem produkt i wyświetliłem dla niego wykres (nazwijmy okno z tym wykresem W1).
Zminimalizowałem W1, w oknie 'wybór danych' wybrałem inny produkt i wyświetliłem dla niego wykres (nazwijmy okno z tym wykresem W2).
Zminimalizowałem W2.
Zamknąłem okno 'wybór danych'.
Zamknąłem W1.
Otworzyło się puste okno 'wybór danych', z komunikatem: 'wybierz podstawowe kryteria selekcji'.
Kliknąłem 'OK'. Wystąpił błąd AdodcTP: 'Authentication failed'.
Kliknąłem 'OK'. Wystąpił błąd -2147217843: 'Automation error'.

6. Jest dokładny i precyzyjny.

Czyli taki, że programista nie musi, w trakcie poprawiania błędu, przychodzić do testera z pytaniami w rodzaju: *co konkretnie miałeś na myśli? o co chodziło ci w tym miejscu?* Ponadto, jeśli do opisu dołączone są jakieś dodatkowe elementy (np. kilka plików ze zrzutami ekranów), to w opisie, w odpowiednich miejscach powinny występować odniesienia do tych elementów. Czyli jeśli np. podajemy sposób reprodukcji błędu, i do opisu dołączamy pięć plików ze zrzutami ekranów zrobionymi w czasie reprodukcji, to przy opisie kolejnych czynności reprodukcji, podajemy nazwy odpowiadających im zrzutów. Tak żeby programista miał jasno powiedziane jaki zrzut odnosi się do jakiej sytuacji, i nie musiał się zastanawiać co do czego pasuje. Na zrzutach warto zaznaczyć (np. edytując plik graficzny pod *Paint*'em), miejsca na które należy zwrócić uwagę.

Przykład.

Kategoria: Korelacje.

Ekran: 839a, 839b, 839c, 839d

Krótki opis: Okresy czasowe są niepoprawne w pewnej sytuacji.

Długi opis: Wybrałem produkt w oknie 'wybór danych', i kliknąłem 'korelacje'. Oznaczmy początkową datę dla danych dotyczących tego produktu przez P1, a końcową przez K1 (patrz 839a). Następnie, w oknie 'korelacje', wybrałem produkt dla drugiego wykresu. Oznaczmy jego początkową datę przez P2, a końcową przez K2. Patrz 839b:
 $P1 = P2 < K2 < K1$.
Następnie wybrałem produkt dla trzeciego wykresu. Oznaczmy jego początkową datę przez P3, a końcową przez K3. Patrz 839c:
 $P1 = P2 < K2 < P3 < K3 = K1$
A zatem, koniec drugiego wykresu, poprzedza początek trzeciego wykresu.
Spójrzmy teraz na tablicę korelacji na ekranie 839d: data początku okresu korelacji jest późniejsza niż data końca tego okresu, a wartości korelacji są puste.

7. Jest pełny i kompletny.

Oszczędzanie przez testera czasu na sporządzaniu opisów błędów, skutkuje często stratą znacznie większej ilości czasu przez programistę, a następnie – przez tegoż testera w trakcie weryfikacji poprawionego błędu. Dlatego (dla zaoszczędzenia czasu własnego i współpracowników), należy opisywać błędy w sposób pełny i kompletny.

Przykład.

Kategoria: Wybieranie danych przy użyciu kryteriów selekcji.

Ekran: 885a, 885b, 885c, 885d, 885e, 885f, 885g, 885h