- Reaguj szybko. Zanim zareaguje konkurencja czy prasa.
- Pokaż, że nie tylko widzisz problem, ale również rozumiesz jego wagę i konsekwencje.
- Kanały komunikacji utrzymuj otwarte. Informuj o postępach prac. Odpowiadaj na pytania.
- Przeproś.
- Powiedz: „To się już więcej nie wydarzy" i zrób wszystko, aby już do tego nie doszło.

Nasza praca polega na szukaniu rozwiązań, które zminimalizują liczbę „uciekinierów". Powinniśmy wdrożyć analizę przyczyny „uciekiniera" i możliwości narzędziowe, procesowe i ludzkie jego powstrzymania.

Sam temat raportowania o defektach zostanie szerzej opisany w części praktycznej tej książki.

4.3.2. Błędy popełniane przez testerów

Również testerzy oprogramowania popełniają błędy. Typowy dla nich jest tzw. błąd zakwalifikowania działania poprawnego jako niepoprawne, który pojawia się, gdy mimo braku defektu tester zgłasza raport o incydencie. Ten typ błędów jest uciążliwy dla innych członków zespołu, analityków czy programistów, bo zmusza ich do poszukiwania problemu, którego realnie nie ma.

Innym typem błędu testera jest nieraportowanie o defekcie, tam gdzie on się pojawił. Istnieje wiele źródeł takiego błędu, takich jak zwykła nieuwaga lub niejednoznaczna specyfikacja produktu. Bez względu na powód przypadki takie mogą być krytyczne dla projektu, ponieważ z jednej strony marnują budżet przeznaczony na testy, a z drugiej dają złudne poczucie poprawności działania, podczas gdy w rzeczywistości pojawia się problem.

Błędy te przekładają się w konsekwencji na wspomnianych już „uciekinierów". Doskonalenie testera powinno się zacząć od analizy własnych zgłoszeń uznanych przez innych członków projektu za „niedefekt" i tych błędów, które mieliśmy okazję wychwycić, ale z jakiegoś powodu tego nie zrobiliśmy.

4.3.3. Defekty powodują defekty

Czasami zdarza się również, że defekty powodują defekty. Z tym stwierdzeniem wiąże się kilka prawd.

- Niepoprawne zachowanie aplikacji ogólnie możemy nazwać awarią. Nie wnikamy, skąd wiemy, że zachowanie jest niepoprawne. Ogólnie akceptujemy, że mamy wystarczającą wiedzę, aby uznać coś za „awarię". Jeśli w aplikacji pojawi się awaria, to może ona mieć wiele konsekwencji dla samej aplikacji, np. może powodować inne awarie. Źródłem awarii jest defekt. O defekcie wynikającym z defektu mówimy wówczas, gdy eliminacja defektu (źródłowego) spowoduje, że również powiązane z nim defekty przestają istnieć.
  Przykład: niepoprawna walidacja formularza może powodować, że nie uda się go uzupełnić i zakończyć mimo poprawności danych. Usunięcie defektu walidacji powoduje, że defekt formularza (defekt wynikający z defektu) rozwiązuje się automatycznie.
- Analizując oprogramowanie, czasami widzimy dużą zależność między niektórymi defektami. Część tych zależności wynika ze wspólnego źródła pomyłki, część to konsekwencja stosowania techniki programistycznej kopiuj – wklej. Niepoprawny kod jest kopiowany do wielu miejsc (które czasami wydają się zupełnie niepowiązane). Znalezienie wzorca defektu pomaga wyeliminować go z wszystkich miejsc, w których został „wklejony".
- Poprawki defektów mogą powodować nowe defekty. Dopóki liczba defektów naprawianych jest większa niż liczba defektów wprowadzanych przez poprawki, dopóty możemy mówić o postępie jakości oprogramowania.

Możemy więc mówić o czymś w rodzaju wzorca defektów. Ich świadomość i zdolność wykrywania będzie kolejnym (ważnym) aspektem rozwoju testera oprogramowania.

Zadanie

Przeprowadź prosty eksperyment poznawczy. W dowolnej aplikacji wskaż awarię i spróbuj udowodnić, że naprawdę jest ona warta naprawienia. Na jakie źródła, autorytety czy dokumenty się powołasz? Jak udowodnisz osobie odpowiedzialnej za dany problem, że awaria musi zostać poprawiona.
