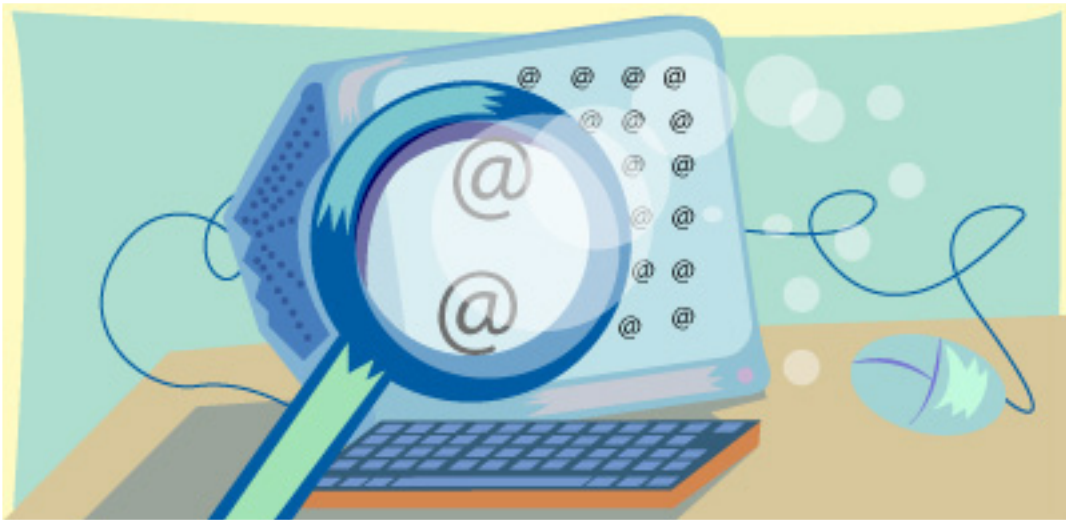
Testowanie aplikacji i stron internetowych



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 4

Koszty testów 4

Czy testy są konieczne i czy możemy ufać programistom? 4

Rodzaje testów 5

Testy automatyczne 5

Testy wydajnościowe 7

Retesty 7

Testy regresywne 7

Testy modułowe i systemowe 8

Testy z udziałem użytkownika 8

Kategoryzacja błędów 8

Zlecanie testów na zewnątrz 9

Dlaczego warto prowadzić testy? 9

Podsumowanie 10

Źródła 10

Wprowadzenie

Jesteśmy przyzwyczajeni do tego, że sprawdzamy jakość tworzonych produktów. Oprogramowanie też jest produktem, a jakość oprogramowania to coś, co jesteśmy w stanie zweryfikować. Środki używane do tego to:

1. kontrola sposobu wytwarzania oprogramowania

2. testowanie oprogramowania.

Pierwsza część związana jest z takimi aspektami jak użyta technologia (np. język programowania) czy metody wytwarzania. To w jaki sposób wytwarzamy aplikacje ma wpływ na ich jakość. Jednak nie jest to całkowicie wystarczające – należy jeszcze tę jakość zweryfikować. Do tego właśnie służy testowanie oprogramowania. Ważne jest to, że dla jakości oprogramowania istotne są oba te elementy. Dlatego tworząc aplikacje powinniśmy zadbać o to, by obydwa z nich wzajemnie się uzupełniały – dopiero to daje kontrolę nad jakością dostarczonego produktu.

W niniejszym e-booku skupimy się na drugim z wymienionych obszarów, czyli testowaniu oprogramowania.

Koszty testów

Testy, jak każdy element procesu wytwórczego, generują koszty. Testy trzeba najpierw zaplanować, potem przeprowadzić, przeanalizować wyniki oraz dokonać priorytetyzacji i podjąć decyzję o naprawieniu. Koszty samego poprawienia błędów trudno zakwalifikować jako koszty samych testów – trzeba zdawać sobie sprawę, że to nie testy wprowadzają błędy, błędy w aplikacji są, a testy tylko je ujawniają. Skoro one i tak tam są, to koszty ich naprawy musimy ponieść – a możemy to zrobić dopiero wtedy gdy błędy wykryjemy. Z punktu widzenia kosztów istotne jest inne spostrzeżenie – im później na etapie tworzenia aplikacji wykryjemy błąd, tym drożej będzie kosztowało jego poprawienie[1]. W skrajnym wypadku po odkryciu błędu dopiero po udostępnieniu aplikacji użytkownikom koszty są największe, bo dochodzą trudno mierzalne koszty jak pogorszenie wizerunku, utrata klientów czy utracone korzyści. Raport NIST z 2002r. szacował łączne straty gospodarki amerykańskiej w wyniku błędów oprogramowania na prawie 60 miliardów dolarów rocznie[2] .

Koszty przeprowadzenia samych testów stanowią istotną pozycję w budżecie tworzenia aplikacji. Dokładne liczby zależą od zarówno specyfiki samej aplikacji jak i wybranego procesu wytwórczego. Szacuje się, że koszty testów to poziom około 10-30% kosztów całego projektu.

Czy testy są konieczne i czy możemy ufać programistom?

Możemy zadać sobie pytanie – skoro testowanie aplikacji jest kosztowne, to może postarać się wyeliminować ten koszt lub chociażby ograniczyć. Czy możemy, przykładowo, zamiast robić testy wynająć lepszych programistów, którzy nie popełniają błędów? Okazuje się, że taki krok – czyli rezygnacja całkowicie z testów byłaby bardzo nierozważnym posunięciem. To tak, jakby na taśmie produkcyjnej zrezygnować z elementów kontroli jakości. Oczywiście poprawienie procesu wytwórczego, a więc użycie odpowiednich metod wytwarzania, zatrudnienie bardziej wykwalifikowanego personelu ma wpływ na jakość, jednak sam proces nie jest w stanie jej zapewnić. Testy są potrzebne i nie należy z nich rezygnować – w dalszej części e-booka powiemy, jak móc przeprowadzać je efektywniej.

Przyczyny powstawania błędów w oprogramowaniu mogą być różne – poniżej lista najczęstszych.

- Błąd w specyfikacji wymagań. Może się okazać, że podczas formułowania wymagań wobec oprogramowania oczekiwania klienta będą wewnętrznie sprzeczne – czyli albo będziemy w stanie spełnić jedno wymaganie, albo drugie. W rzeczywistym świecie kolizje wymagań pojawiają się w konsekwencji zastosowania wielu wymagań jednocześnie i nie zawsze jest je łatwo wychwycić na etapie analizy.
- Błąd w projekcie oprogramowania. Może się zdarzyć, że błąd powstanie na etapie tworzenia projektu – wtedy wymagania zebrane w analizie przekładane są na konkretne rozwiązania, które później będą implementowane.

---

1 Koszty poprawek na różnym etapie projektu były szacowane np. w raporcie NIST, The Economic Impacts of Inadequate Infrastructure for Software Testing Final Report, May 2002, http://www.nist.gov/director/prog-ofc/report02-3.pdf . Poprawienie błędu po udostępnieniu aplikacji klientom może być nawet 30-krotnie droższe niż wykonanie tego na etapie analizy.
2 Software Errors Cost U.S. Economy $59.5 Billion Annually, raport National Institute of Standards and Technology http://www.nist.gov/public_affairs/releases/n02-10.htm

- Pomyłka programisty. Jest to jedna z częstszych przyczyn powstawania błędów – jednak nie jedyna. Nie zawsze programista jest w stanie wszystko przewidzieć, czasem nie potrafi czegoś napisać w odpowiedni sposób, czasem po prostu popełnia błąd.
- Pomyłka administratora. Kolejna osoba, która ma wpływ na końcowy wynik – czyli działające oprogramowanie to administrator. Część wymagań – zwłaszcza pozafunkcjonalne związane z bezpieczeństwem czy wydajnością mogą w sporym stopniu zależeć od administratora. To on konfiguruje infrastrukturę i aplikację tak, by była dostępna dla użytkowników. Ma to szczególne znaczenie w przypadku aplikacji WWW (w tym stron internetowych), gdyż dzisiejsze technologie tworzenia serwisów bazują w sporej części na odpowiedniej konfiguracji.
- Zły proces produkcji oprogramowania. Jest to kolejny element, który może mieć wpływ na generowanie błędów. Można też to stwierdzenie odwrócić – dobrze dobrany proces wytwórczy może zmniejszyć prawdopodobieństwo wystąpienia danego typu błędu. W szczególności są specjalnie dopracowane metodyki wytwórcze[3], które stosuje się przy konkretnych rodzajach oprogramowania.

Widać więc z powyższej listy, że błędy programistów jakkolwiek często popełniane, nie są jedynymi. Możemy mieć do czynienia także z innymi powodami powstania błędów w aplikacji. Niektóre z nich, jak np. błędy w specyfikacji, mogą być wychwytywane na innym poziomie niż testy oprogramowania. W przypadku specyfikacji wymagań i projektu należy dążyć do tego, by na tyle przeanalizować te dokumenty, by nie dopuścić do błędnego wdrożenia. O ile poprawienie samego wymagania, czy projektu na poziomie przeprowadzania analizy nie jest drogie, o tyle poprawki podczas kodowania lub nawet po powstaniu aplikacji mogą wiązać się ze sporymi kosztami. W skrajnych przypadkach może to oznaczać konieczność gruntownych zmian w całym systemie poprzedzonej analizą wymaganych poprawek.

Warto też pamiętać o jednej rzeczy – o tym, po co testujemy oprogramowanie. Naszym celem nie jest samo wykrycie błędów (jest to środek do celu), lecz zweryfikowanie przy możliwie najwyższym poziomie ufności, że oprogramowanie nadaje się do tego, do czego zostało stworzone. Elementem testów jest wykrycie jak największej liczby istniejących błędów – błędy, które nie zostaną wykryte podczas testowania wpłyną później na funkcjonowanie aplikacji. Testy same w sobie nie tworzą błędów – testy tylko służą znalezieniu istniejących błędów.

Rodzaje testów

Tworzenie oprogramowania ma wiele aspektów a z drugiej strony jest wiele sposobów podejścia do testów. Testy możemy podzielić na następujące rodzaje

Testy ogólne

- Testy jednostkowe, modułowe – testy na poziomie pojedynczych, wydzielonych elementów technicznych aplikacji.
- Testy regresji – testy weryfikujące czy w wyniku zmian elementy aplikacji, które już były wcześniej przetestowane, nie mają błędów.
- Testy integracyjne – testy łączenia technicznych jednostek, modułów, czy całych systemów na okoliczność błędów współdziałania.
- Testy systemowe – testy całego systemu/aplikacji.

Testy specjalistyczne

- Testy bezpieczeństwa – sprawdzenie aspektów bezpieczeństwa i ochrony danych aplikacji
- Testy wydajnościowe – sprawdzenie aspektów wydajności oprogramowania, czasem pod tym określeniem rozumie się też testy skalowalności, obciążeniowe.

Testy z udziałem użytkownika

- Testy użyteczności sprawdzające aspekty użyteczności aplikacji.
- Testy uruchomieniowe – sprawdzenie działania aplikacji w prawdziwych warunkach.
- Alfa testy – testy w specjalnie przygotowanym środowisku uruchomieniowym.
- Testy akceptacyjne – testy związane z formalnym odbiorem aplikacji.

Powyższy podział jest jedną z możliwych klasyfikacji i w zależności od wyboru kryteriów możemy stworzyć inne podziały. Poniżej opisane zostaną niektóre z powyższych rodzajów testów, by przybliżyć ich przeznaczenie i sposób przeprowadzania.

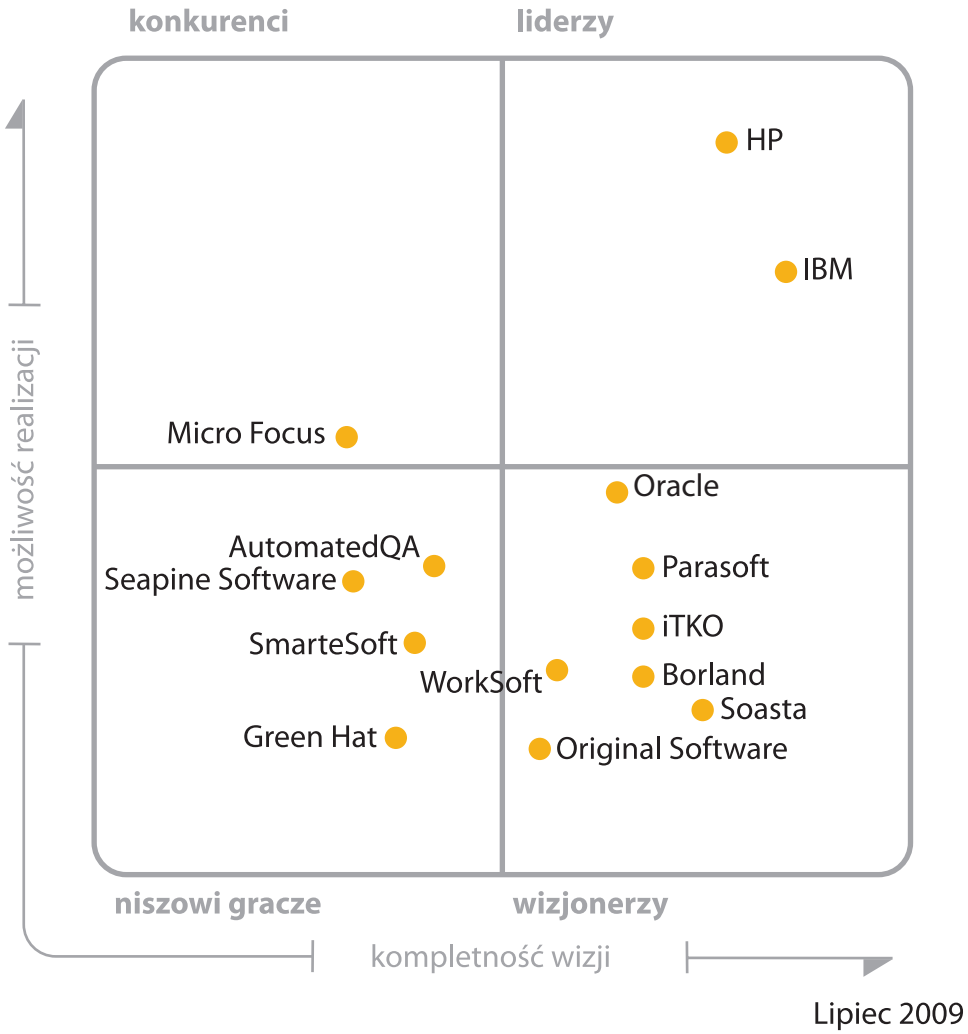
Testy automatyczne

Przy testowaniu oprogramowania bardzo często mamy do czynienia z sytuacją, w której czynności testera są mniej lub bardziej automatyczne. Przykładowo osoba przeprowadzająca testy scenariuszowe (są to testy, w których opisane są krok po kroku czynności, jakie ma wykonać oraz jak powinien w takiej

3 Temat ten omówimy szerzej w jednym z najbliższych e-booków portalu http://www.web.gov.pl

sytuacji reagować program) postępuje zgodnie z rolą podaną w scenariuszu. Sprawdza efekt swoich działań i w przypadku problemów odnotowuje zastrzeżenia. Zauważmy, że podobną czynność mogłaby wykonać maszyna – program komputerowy, który klikałby oraz podawał odpowiednie dane do aplikacji oraz sprawdzał, czy na ekranie pojawiają się spodziewane wyniki. Tak przeprowadzone testy nazywamy testami automatycznymi.

Wśród zalet testów automatycznych należy wskazać niską cenę przeprowadzenia pojedynczego testu oraz krótki czas realizacji. Dla pewnych rodzajów testów (np. testy wydajnościowe) jedyną sensowną metodą ich przeprowadzenia są testy automatyczne. Dla aplikacji internetowych istnieją bardzo przydatne narzędzia do tworzenia testów automatycznych. Wśród rozwiązań są zarówno produkty komercyjne jak i dostępne za darmo[4]. Rysunek 1 przedstawia magiczny kwadrat producentów zintegrowanego oprogramowania do przeprowadzania testów, w tym narzędzia do testów automatycznych.



Rysunek 1. Magiczny kwadrat pozycjonujący producentów zintegrowanego Oprogramowania do zapewnienia jakości oprogramowania. źródło: Gartner (lipiec 2009)[5] . Oś pozioma szereguje dostawców ze względu na kompletność wizji dostarczanego rozwiązania, oś pionowa to różnicowanie ze względu na możliwość realizacji tej wizji. Cztery obszary kwadratu to liderzy, konkurenci, wizjonerzy oraz niszowi gracze.

Trzeba pamiętać, że testy automatyczne to nie tyle oddzielny rodzaj testów lecz sposób przeprowadzania tych testów. Taki sposób realizacji procesu testowania jest naturalnym wyborem przy niektórych rodzajach testów – np. testach regresji, wydajnościowych. O testach tych będzie jeszcze więcej w dalszej części e-booka.

4 Przykładami takich aplikacji są SeleniumIDE, http://seleniumhq.org oraz jmiter, http://jakarta.apache.org/jmeter/, obie dostępne na licencji Apache 2.0
5 Magic Quadrant for Integrated Software Quality Suites, Thomas E. Murhpy, Gartner

Istotnym aspektem jest decyzja dotycząca tego, czy dany rodzaj testu powinien zostać zautomatyzowany[6]. Kwestie, które trzeba wziąć pod uwagę to m.in. czy test będziemy chcieli uruchamiać w przyszłości, czy aplikacja będzie podlegać zmianom, czy planujemy w przyszłości przeprowadzać testy regresji.

Jednocześnie należy pamiętać o tym, że są elementy aplikacji internetowych, które z założenia wykonane są tak, by nie dało się ich wykonywać narzędziami automatycznymi. Przykładem jest zabezpieczenie przez automatyczną obsługą pewnych funkcjonalności stron internetowych, tzw. CAPTCHA[7] – test, który realizuje się w formie obrazka ze zniekształconymi literami, które są bardzo trudne (a w praktyce niemożliwe) do odczytania przez maszynę a łatwy do rozpoznania przez człowieka.

Testy wydajnościowe

Wydajność aplikacji to jeden z aspektów powodzenia funkcjonowania rozwiązania w świecie rzeczywistym. O ile podczas zwykłych testów jednocześnie z aplikacji korzysta do kilku osób, o tyle w świecie rzeczywistym mamy do czynienia z wielokrotnie większą liczbą użytkowników. Testy wydajnościowe muszą pokazać, że system – przy obciążeniu zgodnie z planowanym wykorzystaniem – będzie nadal działał prawidłowo a odpowiedzi będą generowanie w akceptowalnym czasie. Zwykle już na etapie formułowania wymagań oblicza się spodziewane wykorzystanie aplikacji i tak określa parametry, by odzwierciedlały faktyczne, przyszłe obciążenie. Szacunków tych dokonuje się często w oparciu o analizy marketingowe związane z przewidywanym segmentem, sposobem wykorzystania aplikacji, rodzajem przeprowadzanych transakcji.

W przypadku aplikacji mówimy też o takim aspekcie jak możliwość skalowania. W uproszczeniu chodzi o ty, by w przypadku zwiększenia wykorzystania aplikacji wystarczyło rozszerzyć infrastrukturę sprzętową (np. dokupić kolejny serwer). Aplikacja, która jest skalowalna pozwala na równomierne rozłożenie się obciążenia na wiele maszyn bez konieczności dokonywania dużych zmian w samej aplikacji poza zmianą parametrów konfiguracyjnych. Warunkiem tego, by rzeczywiście tak się stało, jest sposób w jaki aplikacja została zaprojektowana i wykonana. Nie wszystkie aplikacje pozwalają na taką prostą poprawę wydajności. Czasem jest to inherentna własność typu oprogramowania a czasem może wynikać z błędów projektowych. Testy wydajnościowe nie dają bezpośredniej informacji nt. możliwości skalowania. Jednak stosując je, możemy sporo się dowiedzieć, np. zwiększając obciążenie aplikacji oraz sprawdzając jakie są czasy odpowiedzi otrzymujemy wskazówki związane z tym, jak skaluje się aplikacja.

Retesty

Retesty to skrótowe określenie ponownych testów. Istnieją dwa podstawowe konteksty, w których używane jest pojęcie retestów

- gdy chcemy powtórzyć jakiś test aplikacji z innymi danymi wejściowymi niż pierwotnie
- gdy podczas testu w aplikacji został znaleziony błąd np. w jakimś scenariuszu, to po poprawieniu błędu w aplikacji wykonujemy test ponownie, by zweryfikować, czy rzeczywiście został poprawiony.

W pierwszym wypadku retesty wykonujemy wtedy, gdy mamy poważne podejrzenia, co do funkcjonowania aplikacji dla zmienionych danych. Wynika to bezpośrednio stąd, że zachowanie aplikacji jest determinowane nie tylko samym oprogramowaniem ale także danymi, które są dostarczane aplikacji lub są w niej przechowywane. Pewne błędy mogą ujawniać się tylko wtedy, gdy przekroczymy pewną ilość danych, są one w specyficzny sposób przygotowane lub z różnych innych powodów.

W drugim wypadku często wykonujemy także testy regresywne dla tych testów, które działały poprawnie wcześniej aby być pewnym, że poprawiając błąd nie został wygenerowany nowy. Może się bowiem okazać, że poprawienie błędu spowodowało zmianę w innej części aplikacji.

Testy regresywne

Testy regresywne to testy, które polegają na testowaniu tych części aplikacji, które wcześniej były testowane i przeszły pomyślnie proces testowania. Istnieje wiele sytuacji, w których chcielibyśmy takie testy przeprowadzać – najważniejsze z nich wymienione są poniżej.

1. Przy tworzeniu oprogramowania często mamy do czynienia z etapowym przygotowaniem aplikacji. W ramach kolejnych wersji oprogramowania dodawane są nowe funkcjonalności a poprzednio dostarczone nie są zmieniane. Nie ma żadnej gwarancji, że części, które testowaliśmy wciąż jeszcze działają poprawnie. Dlatego testy w tych obszarach trzeba powtórzyć.

---

6 When Should a Test Be Automated?, Brian Marick
http://www.stickyminds.com/sitewide.asp?Function=edetail&ObjectType=ART&ObjectId=2010
7 ang. Completely Automated Public Turing test to tell Computers and Humans Apart

2. Oprócz tego aplikacja może podlegać zmianom w trakcie projektu. W projektach zmiany są niemal nieuniknione. Każda wprowadzona zmiana może potencjalnie popsuć już działające części.

3. Po oddaniu aplikacji do utrzymania często jest ona modyfikowana – chociażby rozszerzana jest funkcjonalność. Przy każdej takiej zmianie mogą pojawić się błędy w już istniejącej aplikacji.

4. Jeśli aplikacja jest poprawiana w wyniku odkrycia w niej błędu sama weryfikacja poprawienia błędu może być niewystarczająca.

Za każdym razem w takich sytuacjach należy przeprowadzić testy regresywne, które zweryfikują, czy zmieniając coś nie wprowadziliśmy nowych błędów. Ze względu na to, że okazji do przeprowadzania takich testów może być wiele, to wykonywanie ich ręcznie może być dość pracochłonne – a w efekcie także i kosztowne oraz długotrwałe. Dlatego też naturalną metodą przeprowadzenia testów regresji jest przygotowanie testów automatycznych i uruchamianie ich przy każdej kolejnej rozbudowie systemu. Wtedy koszty testów regresywnych będą sprowadzały się do uzupełniania zestawu testów o nowe scenariusze, aktualizowaniu istniejących oraz uruchamianie w odpowiednich momentach.

Testy modułowe i systemowe

Aplikacje stają się coraz bardziej złożone. Dlatego też w ramach aplikacji tworzone są wydzielone części funkcjonalności (moduły), które mogą działać niezależnie od pozostałych części. Taki sposób konstruowania aplikacji ma swoje zalety wśród których najważniejsze to możliwość podziału skomplikowanej aplikacji na mniejsze części czy ponowne używanie modułów w innych systemach. Z punktu widzenia weryfikowania poprawności działania aplikacji możemy poprawność tą badać na poziomie modułu bądź całego systemu. Stąd też pochodzą nazwy tego rodzaju testów – modułowe i systemowe.

Testy modułowe są bardzo wygodną formą weryfikacji poprawności aplikacji na etapie wytwarzania oprogramowania. Wykonywane są zwykle przez programistów, którzy zajmują się wydzieloną częścią aplikacji. Jej wielkość jest określona przez konkretny projekt aplikacji, wybraną technologię, wielkość i złożoność systemu. Przeprowadzenie takich testów pozwala programiście stwierdzić, że część kodu, za którą jest odpowiedzialny działa prawidłowo.

Prawidłowe działanie wszystkich części aplikacji nie gwarantuje poprawności całego systemu. Dlatego nieodłącznym elementem testów są także testy systemowe.

Testy z udziałem użytkownika

Nie we wszystkich testach aplikacji bierze udział użytkownik – jednak są klasy testów, które weryfikują takie obszary wymagań, które są istotne z punktu widzenia użytkownika i trudno jest je przeprowadzić bez niego. Jednym z takich obszarów jest użyteczność aplikacji . Określa ona łatwość posługiwania się oprogramowaniem przez użytkownika. Wymagania związane z użytecznością należą do klasy wymagań pozafunkcjonalnych, czyli takich, które nie wpływają bezpośrednio na to, co robi aplikacja, tylko na to w jaki sposób użytkownik wchodzi w interakcję z programem by uzyskać oczekiwany wynik. Testy użyteczności powinny być przeprowadzane z udziałem faktycznego użytkownika aplikacji, gdyż dopiero wtedy jesteśmy w stanie zweryfikować, jak będzie wyglądało korzystanie z aplikacji. Dotyczy to w szczególności stron internetowych, gdzie specyfika sposobu korzystania z nich użytkownika powoduje np., że potencjalny klient, który nie jest w stanie znaleźć w krótkim czasie tego, czego szuka, rezygnuje. Błędy takie – czy raczej można je nazwać niedociągnięciami – mogą powodować tak samo duże straty co błędy w samej funkcjonalności aplikacji.

Innymi testami, w które należy zaangażować użytkowników są testy akceptacyjne. Potwierdzenie przydatności aplikacji dla użytkowników, a w szczególności akceptacja aplikacji jako produktu powinna być związana z zaangażowaniem osób, które będą z programów korzystały na co dzień.

Kategoryzacja błędów

Wynikiem przeprowadzenia testów jest lista zastrzeżeń do aplikacji stanowiąca listę dostrzeżonych uchybień. Nie oznacza to, że wszystkie kwestie zgłoszone przez testerów powinny być automatycznie obsłużone. Zgłoszenia są tylko potencjalnymi błędami w aplikacji, trzeba też pamiętać, że mogą być to wykryte w oprogramowaniu luki, które nie zostały odpowiednio przeanalizowane na etapie tworzenia projektu. W szczególności ten drugi rodzaj może stanowić podstawę do wniosków o zmianę w aplikacji – testy stają się wtedy elementem doskonalenia.

Pomocą w takiej sytuacji jest odpowiednia kategoryzacja zgłoszeń z testów. Pozwala to na uzgodnienie pomiędzy klientem i dostawcą sposobu obsługi błędu oraz jego naprawy.

W szczególności wprowadzenie kategorii błędów oraz przypisanie do nich zgłoszeń może być pomocne w ocenie, czy aplikacja wymaga ponownego pełnego procesu testowania, czy też błędy do poprawienia są na tyle drobne, że akceptacja może być dokonana w oparciu o sprawdzenie, czy zostały poprawione znalezione błędy. Ogólne wytyczne dotyczące tolerancji błędów można zawrzeć już w kontrakcie na wykonanie (jeżeli aplikacja jest dostarczana przez zewnętrznego wykonawcę).

Zlecanie testów na zewnątrz

Jednym ze sposobów przeprowadzenia testów aplikacji jest zlecenie tego na zewnątrz organizacji – albo zewnętrznej firmie, albo wynajmując pracowników kontraktowych. Podejście takie może być uzasadnione w sytuacji, w której nie posiadamy wykwalifikowanego personelu, który posiada odpowiednią wiedzę i umiejętności do zaplanowania procesu testowania i jego przeprowadzenia. Nie musi to oznaczać, że wszystkie testy będą wykonywać osoby z zewnątrz naszej firmy. Przykładem testów, które trudno przeprowadzić samemu bez odpowiedniego przygotowania są testy bezpieczeństwa. Istotne jest, by odpowiednio określić zakres przeprowadzania takich testów – uniknąć można wtedy niewspółmiernych wydatków w stosunku do obszaru weryfikacji. W przypadku testów bezpieczeństwa mamy różne rodzaje sposobu badania podatności aplikacji. W szczególności może to być np. możliwość badania aplikacji pod kątem wykrycia podatności. Na rynku dostępnych jest wiele narzędzi, które w sposób automatyczny badają istnienie znanych i skonfigurowanych w narzędziu podatności. Samo uruchomienie takiego narzędzie oraz przedstawienie automatycznego raportu ma niewielką wartość dla klienta – może być wstępnym etapem to bardziej zaawansowanych testów, czy chociażby odpowiedniej interpretacji wyników w kontekście testowanego systemu.

Innym przykładem testów, które możemy zlecać do wykonania na zewnątrz, są testy automatyczne – stworzenie odpowiednich scenariuszy i nagranie testów wymaga sporego doświadczenia. Osoby, które zajmowały się testami automatycznymi w swojej pracy są w stanie je stworzyć wyjątkowo szybko i efektywnie, łatwo będzie te testy utrzymywać (pamiętajmy, że testy automatyczne trzeba aktualizować wraz z rozwojem i zmianami aplikacji). Dobre testy mogą oszczędzić dużo pracy w przyszłości.

Na istotną wartość jaką wnoszą zaawansowani testerzy do testów wskazuje publikacja Forrester Research[8]. W publikacji wskazano, że zatrudnienie wykwalifikowanych testerów pozwala zmniejszyć ich liczbę oraz skrócić testy, co przekłada się na zmniejszenie kosztów przeprowadzenia testów. Artykuł dotyczy tzw. lekkich metodyk tworzenia oprogramowania jednak niektóre wnioski można uogólnić także na inne typy projektów.

Wybierając firmę zewnętrzną lub pracownika kontraktowego dobrze jest zwrócić uwagę na posiadane certyfikaty. Istnieją dwa podstawowe, powszechnie uznawane systemy certyfikacji testerów oprogramowania: ISTQB (*International Software Testing Qualifications Board*[9]) oraz ISEB *(Information Systems Examination Board*[10]). Organizacje te mają własne systemy certyfikacyjne, przy czym na poziomie Foundation (podstawowym) certyfikaty są wzajemnie uznawane.

Biorąc pod uwagę istniejące standardy oraz wiedzę fachowa i doświadczenie potrzebne nie tylko do przeprowadzenia testów ale zwłaszcza do zaprojektowania i zaplanowania testów wykonanie testów siłami zewnętrznymi jest wartą do rozważenia opcją. W ogólnym rozrachunku może się okazać tańsza niż wykonywanie wszystkiego samodzielnie.

Dlaczego warto prowadzić testy?

Podstawowe argumenty za tym, że warto przeprowadzać testy oprogramowania związane są z aspektami finansowymi. Czynniki wpływające na decyzje o zakresie testowania aplikacji przedstawione są poniżej.

1. Koszty poprawienia błędu w aplikacji jest najniższy na początkowym etapie oraz rośnie wraz z zaawansowaniem prac. Dlatego wcześniejsze wykrycie błędu oznacza mniejsze koszty.

2. Koszty poprawienia błędu po produkcyjnym uruchomieniu aplikacji są najwyższe, gdyż poza poprawieniem samego błędu musimy uwzględnić także koszt sprawdzenia, czy poprawka nie psuje czegoś innego w aplikacji.

3. Błąd w aplikacji uruchomionej produkcyjnie powoduje także generowanie kosztów utraconych korzyści – jeśli np. błąd będzie polegał na tym, że klienci nie będą mogli dokonać zakupu to niechybnie zrezygnują z prób i dokonają zakupu gdzieś indziej.

---

8 A Few Good (Agile) Testers Introducing Agile Testing Brings Quality Without An Army Of Testers, Margo Visitacion we współpracy z Johnem R. Rymerem i Adamem Knollem, Forrester Research
9 http://www.istqb.org
10 http://www.bcs.org/iseb

4. W skrajnych sytuacjach może mieć to wpływ na wizerunek firmy co może powodować dalej idące konsekwencje.

Widać z tej krótkiej listy, że warto testować aplikacje a odpowiednio przygotowany i przeprowadzony proces testowania nie musi być bardzo drogi. W praktyce zależy on od wielkości i specyfiki aplikacji, jednak można przyjąć, że koszty testowania mogą sięgnąć 20-30% kosztów tworzenia aplikacji. Jednocześnie pozwala to na uniknięcie w przyszłości znacznie wyższych kosztów związanych z ujawnianymi błędami.

Podsumowanie

Aplikacje i strony internetowe można traktować tak samo jak inne produkty. Dlatego też podlegają one zarządzaniu jakością. Ze względu na specjalistyczny charakter tych produktów stosujemy do tego dedykowane metody – specjalistyczne testy. Większość typów testów została przybliżona w niniejszym e-booku – ze względu na specjalistyczny charakter – w uproszczony i przystępny sposób.

Koszty samych testów należy rozważać w kontekście kosztów błędów oprogramowania – związane jest bardziej z zarządzaniem ryzykiem. Decyzja o przerwaniu testów wiąże się z dojściem do momentu, w którym dalsze poprawianie aplikacji jest droższe niż oczekiwane ewentualne straty z niewykrytych błędów. Testy są narzędziem w oszacowaniu tego poziomu pewności działania oprogramowania.

Źródła

http://www.istqb.org/downloads/glossary-current.pdf, słownik terminów, ISTQB Standard Glossary of Terms used in Software Testing V.2.0, Dec, 2nd 2007

http://www.dmoz.org/Computers/Programming/Software_Testing/Products_and_Tools/, katalog narzędzi i produktów związanych z testowaniem oprogramowania

http://www.nist.gov/director/prog-ofc/report02-3.pdf, The Economic Impacts of Inadequate Infrastructure for Software Testing, NIST

http://sunnyday.mit.edu/papers/therac.pdf , artykuł o Therac 25 (błędy oprogramowania spowodowały śmierć pacjentów), Medical Devices: The Therac 25 – Nancy Leveson

http://www.bcs.org/iseb, http://www.istqb.org , strony organizacji certyfikujących w zakresie kompetencji związanych z testowaniem oprogramowania.