Typy testów

Jakie typy testów wyróżniamy i co oznaczają?

Typy testów

Typy testów dzielimy ze względu na określone cele:

- przetestowanie danej funkcjonalności
- przetestowanie wybranych właściwości takich jak użyteczność, niezawodność i przenaszalność
- wykonanie testów architektury systemu
- weryfikacja naprawionych defektów
- poszukiwanie niezamierzonych zmian

Testy funkcjonalne (poziomy testów)

Testowanie funkcjonalne

Testy funkcjonalne koncentrują się na wymaganiach biznesowych aplikacji lub strony. Dzięki nim można sprawdzić, czy wszystkie jej funkcje i cechy są zgodne z określonymi wcześniej kryteriami. Co za tym idzie – czy produkt końcowy spełnia zakładane oczekiwania klienta.

Poszczególne testy funkcjonalne są zwykle przeprowadzane w określonej kolejności. Wśród nich wyróżnia się 4 poziomy, a każdy z nich dotyczy innej fazy wytwarzania oprogramowania.

Testowanie funkcjonalne - poziomy testów

- **Testy jednostkowe (modułowe)** – zazwyczaj wykonywane są przez programistę na etapie tworzenia aplikacji. Na tym poziomie testuje się poszczególne jednostki oprogramowania (najmniejsza testowalna część aplikacji) i sprawdza, czy każda z nich działa zgodnie z założeniami.
- **Testy integracyjne** – sprawdzają, czy moduły lub usługi używane przez aplikację prawidło ze sobą współpracują. Po przetestowaniu poszczególnych jednostek są one integrowane z innymi. Tworzą wtedy moduły lub komponenty, które mają wykonywać określone funkcje. Wtedy testuje się je jako całe segmenty w celu zweryfikowania, czy zachowują się zgodnie z założeniami, a interakcja jest płynna. Dzięki tym testom można np. sprawdzić, czy połączenie między aplikacją a bazą danych działa właściwie lub zbadać integrację z systemami zewnętrznymi.

Testowanie funkcjonalne - podział

Testy systemu – badają oprogramowanie pod kątem awarii lub błędów. Głównym ich celem jest kompleksowa ocena specyfikacji systemu – jego funkcjonalność jest testowana od początku do końca zanim zostanie wprowadzony do produkcji.
Testy akceptacyjne – to rodzaj testów wykonywanych przez klienta w celu zaakceptowania aplikacji przed przeniesieniem jej do środowiska produkcyjnego. Sprawdza on, czy produkt działa bez zarzutu, jest zgodny z wymaganiami biznesowymi i spełnia zakładaną funkcjonalność.

Testy niefunkcjonalne

Testowanie niefunkcjonalne

Testy niefunkcjonalne obejmują, ale nie są ograniczone do: testów wydajności, testów obciążeń, testów przeciążające, testów użyteczności, testów niezawodności, testów efektywności, testów przenaszalności i testów zdolności do pracy na różnych platformach. Ten typ testowania określa jak działa system.

Testy niefunkcjonalne mogą być wykonane na wszystkich poziomach testowych. Pojęcie niefunkcjonalnego testowania opisuje testy wymagane do pomiarów charakterystyk systemu i oprogramowania, które może być ocenione na skali. Przykładem może być czas odpowiedzi podczas weryfikowania wydajności.

Testowanie niefunkcjonalne - podział

- **Testy bezpieczeństwa** – proces mający na celu ustalenie, czy system posiada jakieś luki w zabezpieczeniach, które zwiększają zagrożenie złośliwymi atakami, które mogą skutkować np. utratą danych, pieniędzy lub zaszkodzić reputacji marki.
- **Testy wydajności** – to technika używana do określenia tego, jak aplikacja będzie się zachowywać w różnych warunkach. Jej celem jest sprawdzenie zachowania oprogramowania w rzeczywistych sytuacjach, w których będzie wykorzystywana przez użytkownika. Testom poddaje się szybkość aplikacji (czas jej reakcji), skalowalność (maksymalne obciążenie użytkownika, jakie może obsłużyć aplikacja) oraz stabilność przy różnych obciążeniach.

Testowanie niefunkcjonalne - podział

- **Testy użyteczności** – zwany też testowaniem UX. To metoda mierząca, jak łatwa i przyjazna dla użytkownika jest aplikacja. Jej główną ideą jest ustalenie, czy oprogramowanie pod względem wizualnym i funkcjonalnym spełnia swojej cele, a także jest zrozumiałe i wygodne w obsłudze.
- **Testy zgodności** – oceniają, czy oprogramowanie jest dostosowane do działania w różnych środowiskach. Kompatybilność testowana jest z uwzględnieniem rodzaju sprzętu, urządzenia, systemu operacyjnego, sieci, oprogramowania, przeglądarki internetowej itp..

Testy strukturalne

Czarna skrzynka

Funkcje jakie system, podsystem lub komponent wykonują mogą być opisane jako wymagania specyfikacji, przypadki użycia lub specyfikacja funkcjonalności, mogą też pozostać nieudokumentowane. Tryb nieudokumentowany sprawdza się w zespołach o wysoko rozwiniętej umiejętności komunikacji. Funkcje definiuje się jako czynności wykonywane przez system. Testy funkcjonalne oparte są na tych funkcjonalnościach lub funkcjach (opisanych w dokumentacji lub zrozumiane przez testera) i mogą być wykonane na każdym poziomie testowania.
Testowanie funkcjonalne analizuje zewnętrzne zachowanie oprogramowania, traktując ją jak **czarną skrzynkę**.

Czarna skrzynka

Testy czarnoskrzynkowe to technika projektowania przypadków testowych, w której przypadki są projektowane bez zaglądania w wewnętrzne mechanizmy działania modułu (bez znajomości kodu). Mogą być realizowane przez użytkowników (testerzy biznesowi), którzy nie muszą posiadać wiedzy technicznej. Takie osoby, korzystając z aplikacji na co dzień, mogą być nieocenionym źródłem informacji na temat tego, czy rzeczywiście działa ona niezawodnie, czy jest intuicyjna i czy posiada wszystkie niezbędne funkcje.

Biała skrzynka

Testy białoskrzynkowe mogą być wykonane na wszystkich poziomach testowych. Używa się ich w celu wsparcia pomiarów dokładności testowania poprzez ocenę pokrycia danego typu struktury. Pokrycie jest mierzone wykonaniem kodu za pomocą szeregu testów i wyrażone w procentach pokrytych elementów.

Biała skrzynka

Testy białej skrzynki to technika projektowania przypadków testowych na podstawie analizy struktury modułu, czyli wewnętrznej struktury kodu. Najczęściej wykonywane przez doświadczonych testerów lub programistów. Wymagają wiedzy technicznej — znajomości składni języka, w którym napisano program, umiejętności i analizy kodu, obeznania z dobrymi zasadami programistycznymi itp. Przykład: testowanie warunków logicznych w kodzie źródłowym.

Testy związane ze zmianami

Testy regresji

Testy regresji są powtarzalnymi testami na już przetestowanym wcześniej programie, po modyfikacjach, w celu wykrycia innych defektów wprowadzonych lub nie odkrytych podczas naprawy. Mogą one znajdować się w testowanym oprogramowaniu jak i w innym powiązanym lub niepowiązanym z nim komponencie. Testy regresji zostają wykonane gdy oprogramowanie, lub środowisko, zostaje zmienione. Głównym zadaniem tych testów jest ocena ryzyka, a nie znalezienie defektów w oprogramowaniu, które poprzednio działało. Mają zapewnić, że system działa poprawnie po każdej nowej modyfikacji.

Testy regresji cd.

Testy regresji powinny być powtarzalne, jeśli mają służyć do potwierdzenia usunięcia defektu. Testy regresji mogą zostać wykonane na wszystkich poziomach testowych i zajmują się funkcjonalnością, parametrami niefunkcjonalnymi i testowaniem strukturalnym. Zbiory testów regresji są wykonywane wiele razy i ogólnie spowalniają rozwój oprogramowania, są więc świetnym kandydatem na automatyzację.

Testy potwierdzające (retesty)

Testy potwierdzające mają na celu potwierdzenie, że naprawione defekty zostały skutecznie usunięte. Po wprowadzeniu poprawek testerzy przeprowadzają te same testy, które wcześniej wykazały błąd, aby upewnić się, że problem został rozwiązany.

Testy dymne (smoke testy)

To testy, które mają na celu potwierdzenie, że najważniejsze funkcje systemu działają, zanim przeprowadzone zostaną bardziej zaawansowane testy. Sprawdzają w pierwszej kolejności, kluczowe funkcje danego systemu.

Testy ręczne i automatyczne

Testowanie ręczne

W przypadku manualnego testowania oprogramowania, analiza i sprawdzanie jakości kodu wykonywane są ręcznie przez testera. Sprawdza on wszystkie istotne cechy aplikacji lub strony, wchodząc z nią w interakcje bez użycia jakichkolwiek zautomatyzowanych narzędzi czy skryptów. Kontrola jakości dokonywana jest na podstawie unikalnych scenariuszy testowych z perspektywy użytkownika końcowego.

Testowanie ręczne

I chociaż wiadomo, że automatyzacja oszczędza czas, testowanie ręczne jest bardzo istotną częścią tworzenia oprogramowania. Testerzy przyjmują wtedy rolę tych, którzy z aplikacji lub strony będą ostatecznie korzystać. Dzięki temu mogą myśleć jak użytkownicy końcowi i wyobrażać sobie wiele możliwych do zaistnienia scenariuszy dla aplikacji lub funkcji.

Testy automatyczne

Testy automatyczne są wykonywane również przez testerów z tą różnicą, że wspierają się oni specjalnymi narzędziami i skryptami, które pomagają zautomatyzować część procesów. Mogą one obejmować zarówno jakąś pojedynczą funkcję oprogramowania, jak i całą sekwencję zdarzeń w interfejsie użytkownika. Uwzględniając przygotowany scenariusz, testy automatyczne porównują otrzymane wyniki z tymi, które zostały w nim założone. Testy automatyczne pozwalają skutecznie przyspieszyć cykl testowania — wykazują wysokie wyniki przy niewielkim obciążeniu osób testujących. W praktyce oznacza to zwiększenie stopnia pokrycia testami dzięki rozszerzeniu automatyzacji o te części aplikacji, które mogły nie być dokładnie testowane w poprzednich wersjach.

Testy automatyczne

Automatyzacja testów pozwala na:

- wykonanie szybkich testów w celu potwierdzenia nowo dodanych funkcjonalność i braku błędów w niezmienionej części systemu
- wykonanie testów po naprawieniu błędu występującego w czasie pracy aplikacji w środowisku testowym
- sprawdzenie stabilności i działania systemu w trybie 24/7 (nieprzerwane działanie)

Testy automatyczne są bardziej **niezawodne** od testów ręcznych, jednak ich jakość zależy od tego, jak dobrze zostały napisane skrypty. Są odpowiednie w przypadku dużych projektów, które wymagają stałej weryfikacji tych samych obszarów oraz aplikacji, które przeszły już wstępny etap testowania ręcznego.