Porównanie metod i technik testowania oprogramowania

Damian Ryś
Maja Wojnarowska

Testy oprogramowania

Testowanie oprogramowania jest to proces związany z wytwarzaniem oprogramowania. Jest on jednym z procesów kontroli jakości oprogramowania. Testowanie ma dwa główne cele:

- ***weryfikację oprogramowania***
- ***walidację oprogramowania***

Weryfikacja oprogramowania ma na celu sprawdzenie, czy oprogramowanie jest zgodne ze specyfikacją.

Walidacja sprawdza, czy wytwarzane oprogramowanie jest zgodne z oczekiwaniami użytkownika.

Po co są testy?

- Dzięki testom można wykryć błędy we wczesnych stadiach rozwoju oprogramowania. Pozwala to zmniejszyć koszty usunięcia błędu.
- Testy warto przeprowadzać na każdym etapie tworzenia oprogramowania.
- Testowanie należy rozpocząć jak najwcześniej, ponieważ podstawowymi źródłami błędów są specyfikacja i projekt.
- Im później wykryty zostanie błąd tym trudniej go usunąć (większy jest koszt jego usunięcia).

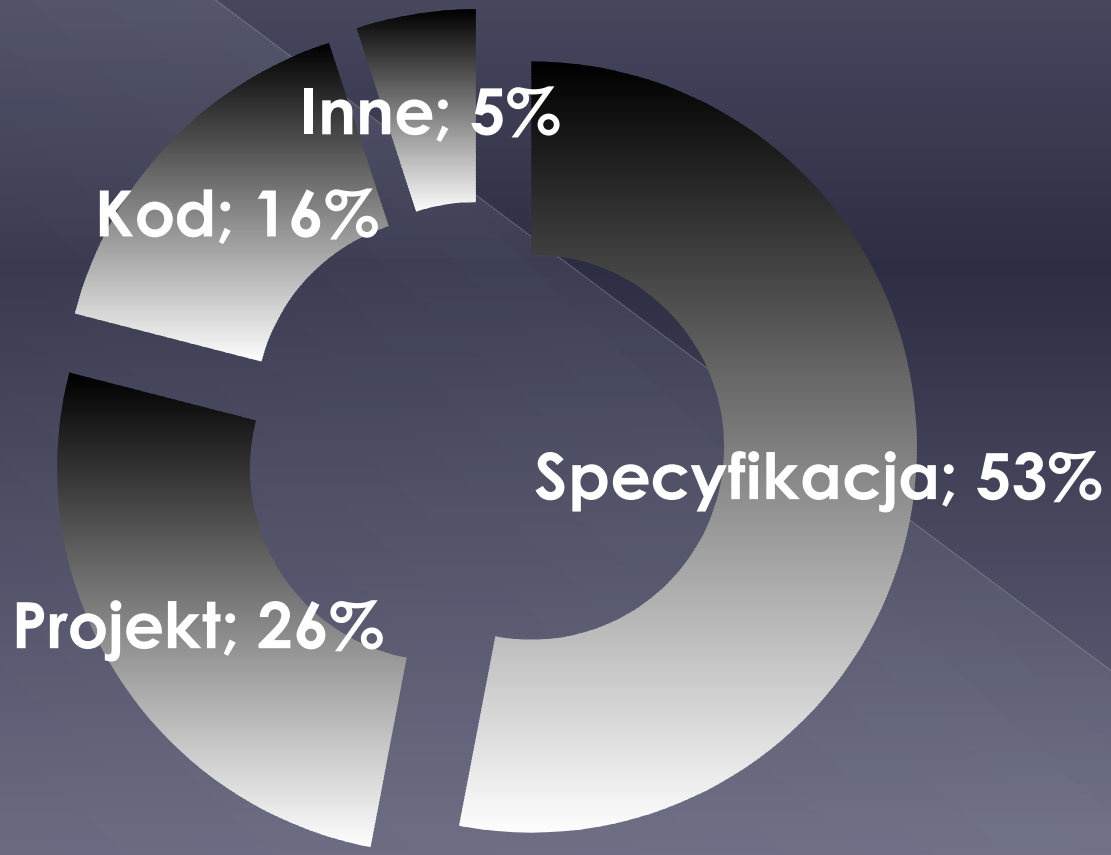
Rodzaje błędów

1. **Oprogramowanie nie wykonuje czegoś, co według specyfikacji powinno wykonywać.**
2. **Oprogramowanie robi coś, czego według specyfikacji nie powinno robić.**
3. **Oprogramowanie wykonuje coś, o czym specyfikacja nie wspomina.**
4. **Oprogramowanie nie wykonuje czegoś, czego nie ma w specyfikacji, ale powinno być.**
5. **Oprogramowanie jest niezrozumiałe, trudne w użyciu, powolne albo – zdaniem testera – będzie w oczach użytkownika po prostu nieprawidłowe.**

Gdzie najczęściej występują błędy:

- w odwołaniach do zmiennych
- w deklarowaniu zmiennych
- w obliczeniach
- w porównaniach
- w przepływie sterowania
- w parametrach procedur
- w wejściu i wyjściu

Gdzie powstają błędy?



Rodzaje testów

Testy można podzielić w różny sposób w zależności od punktu widzenia. Np. na:

- **Wykrywanie błędów – ich celem jest wykrycie jak największej liczby błędów w programie**
- **Testy statystyczne – sprawdzają jakie są przyczyny najczęstszych błędnych wykonań. Oceniają niezawodność systemu.**

Rodzaje testów

Testy można również podzielić na:

- **Testy statyczne – polegające na analizie kodu**
- **Testy dynamiczne – wykonywanie programu (lub jego fragmentów) i sprawdzanie poprawności otrzymanych wyników.**

Podział testów dynamicznych:

Testy dynamiczne możemy podzielić na:

- **Testy funkcjonalne**
- **Testy strukturalne**

Testy funkcjonalne

Testując funkcjonalnie wcielamy się w rolę użytkownika. Oprogramowanie programu jest dla nas tzw. „czarną skrzynką”. Oznacza to, że testując nie wnikamy w szczegóły techniczne działania programu.

Pełne przetestowanie rzeczywistego systemu jest praktycznie niemożliwe z powodu ogromnej liczby kombinacji danych wejściowych i stanów. Nawet dla stosunkowo małych programów ta liczba kombinacji jest tak ogromna, że pełne testowanie wszystkich przypadków musiałoby rozciągnąć się na miliardy lat.

Zwykle przyjmuje się, że jeżeli dana funkcja działa poprawnie dla kilku danych wejściowych, to działa także poprawnie dla całej klasy danych wejściowych. Jednak fakt poprawnego działania w kilku przebiegach nie gwarantuje zazwyczaj, że błędne wykonanie nie pojawi się dla innych danych z tej samej klasy.

Z tego powodu warto przeprowadzać testy dla wartości granicznych np. min i max, oraz dla takich wartości, które wynikają z opisu wymagań.

Testy strukturalne

Testy strukturalne nazywane są potocznie testami *białej lub szklanej skrzynki*. Testując oprogramowanie obserwujemy w jaki sposób zachowują się poszczególne elementy aplikacji, a także jakie moduły i biblioteki są wykorzystywane w trakcie testu.

Istnieje szereg kryteriów, według których testuje się oprogramowanie. Najważniejsze z nich to:

- **Kryterium pokrycia wszystkich instrukcji** – dane wejściowe należy dobierać tak, aby każda instrukcja została wykonana co najmniej raz.
- **Kryterium pokrycia instrukcji warunkowych** – dane wejściowe dobierane są w taki sposób, aby każdy elementarny warunek instrukcji warunkowej został co najmniej raz spełniony i co najmniej raz nie spełniony.

Testy regresyjne

Istnieje jeszcze jeden rodzaj testów, które należałoby w tym miejscu omówić. Są to testy regresyjne. Testy te sprawdzają czy, dodając nową funkcjonalność lub poprawiając błędy, nie naruszyliśmy innej funkcjonalności oprogramowania.

Testy regresyjne powinny być wykonywane zarówno na poziomie kodu aplikacji (jeśli to możliwe) – zazwyczaj są to testy jednostkowe – jak i na wyższym poziomie działania całej aplikacji. Aplikacja jest testowana w ten sposób, że przechodzimy przez wybrane ścieżki działania oprogramowania tak, jak by to robił jego użytkownik.

Rodzaje testów

Kolejnym podziałem testów jest podział na:

- **Testy jednostkowe**
- **Testy integracyjne**
- **Testy systemowe**
- **Testy akceptacyjne**

Testy jednostkowe

Testy jednostkowe wykonywane są przez programistów w środowisku laboratoryjnym. Ich celem jest sprawdzenie pojedynczej jednostki oprogramowania jaką jest klasa, metoda, czy też zbiór współpracujących ze sobą klas (tzw. klaster klas).

Testy integracyjne

Testy integracyjne przeprowadzane się podczas integracji. Ich zadaniem jest sprawdzenie łączonych fragmentów kodu. Testy te są również wykonywane są przez grupę programistów w środowisku laboratoryjnym odpowiedzialną za łączone moduły. Weryfikowana jest współpraca integrowanych jednostek między sobą. Testowanie ma na celu określenie czy po zintegrowaniu otrzymany podsystem nadaje się do dalszego testowania. Proces łączenia i testowania jest powtarzany aż do powstania całego systemu.

Testy systemowe

Testy systemowe wykonywane są przez programistów lub niezależny zespół w kontrolowanym środowisku laboratoryjnym po pomyślnej integracji jednostek wchodzących w skład systemu będącego przedmiotem testowania. Sprawdzają one czy system jako całość spełnia wymagania funkcjonalne i jakościowe postawione przez klienta.

Testy akceptacyjne

System jest poddawany ostatecznym testom, a testerami są tu użytkownicy (klienci), gdzie następnie poddawany jest kolejnym testom. Sprawdzane jest czy system spełnia oczekiwania klienta. Testowanie należy przeprowadzać poprzez kolejne etapy, zaczynając od testów jednostkowych, przez testy integracyjne na testach systemowych skończywszy.

Rodzaje testów

Testy można również podzielić ze względu na:

- **Sposób przeprowadzania**
- **Zakres aplikacji jaki obejmują testy**
- **Przeznaczenie**
- **Testy wydajnościowe i obciążeniowe**

Podział ze względu na sposób przeprowadzania

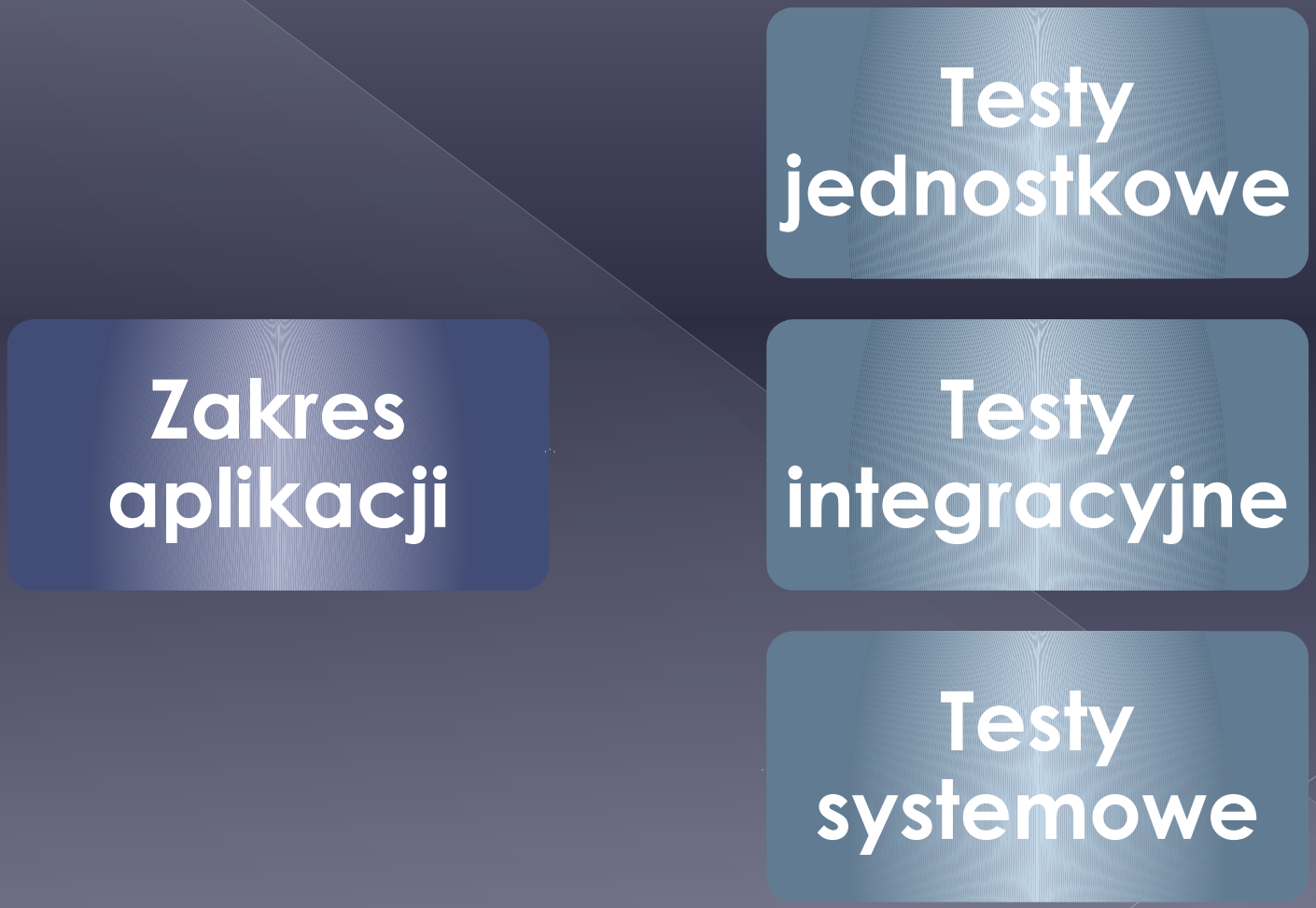
Testy oprogramowania mogą być wykonywane manualnie, bądź automatycznie.

Testy mogą być wykonywane ręcznie przez testera, który przechodzi przez interfejs użytkownika zgodnie z określoną sekwencją kroków lub automatyczne, których wykonanie nie wymaga udziału testera.

W sposób zautomatyzowany przeprowadza się zazwyczaj testy jednostkowe. Zrobienie tego jest dość proste, gdyż istnieją takie narzędzia jak Jakarta Ant, które mają wbudowaną funkcjonalność uruchamiania testów jednostkowych.

Znacznie bardziej skomplikowaną sprawą jest automatyzacja testów w schemacie czarnej skrzynki. Potrzebne do tego jest specjalistyczne oprogramowanie, które pozwala uruchamiać uprzednio napisane lub nagrane przez testera skrypty.

Podział ze względu na zakres aplikacji



Podział ze względu na zakres aplikacji

- **Testy jednostkowe** testują oprogramowanie na najbardziej podstawowym poziomie – na poziomie działania pojedynczych funkcji (metod).

- **Testy integracyjne** pozwalają sprawdzić jak współpracują ze sobą różna komponenty oprogramowania, obecnie rzadko mamy do czynienia z monolitycznymi aplikacjami, raczej są one tworzone modułowo, więc trzeba sprawdzić, czy wszystko razem działa poprawnie, nie ma niezgodności interfejsów itp.

- **Testy systemowe** dotyczą działania aplikacji jako całości, zazwyczaj na tym poziomie testujemy różnego rodzaju wymagania niefunkcjonalne: szybkość działania, bezpieczeństwo, niezawodność, dobrą współpracę z innymi aplikacjami i sprzętem.

Podział testów wydajnościowych i obciążeniowych

Testy instalacyjne / testy konfiguracji

Testy wydajnościowe i obciążeniowe

Testy używalności

Testy post-awaryjne

Testy wersji alfa i beta

Podział testów wydajnościowych i obciążeniowych

- **Testy instalacyjne/testy konfiguracji – służą do tego, żeby sprawdzić, jak oprogramowanie zachowuje się na różnych platformach sprzętowych, systemach operacyjnych, różnych wersjach tych systemów, przy różnym zestawie oprogramowania, jakie może mieć odbiorca.**
- **Testy wersji alfa i beta – testy te służą głównie zdobyciu informacji zwrotnej od użytkowników – wybranej grupie przekazujemy wstępne wersje produktu, następnie zbieramy ich opinie i komentarze dotyczące działania produktu.**

- **Testy używalności** – służą temu by sprawdzić jak szybko potencjalni użytkownicy mogą opanować działanie aplikacji, na ile użyteczna i jasna jest dokumentacja, itp.

- **Testy post-awaryjne** – testy służące sprawdzeniu, czy aplikacja zachowuje się poprawnie po wystąpieniu sytuacji awaryjnej. W pewnych przypadkach jest to bardzo ważny rodzaj testów, np. przykład producent bazy danych powinien sprawdzić na ile awaria wpłynie na integralność przechowywanych danych.

Techniki tworzenia testów i danych testowych

- **Podstawową techniką tworzenia testów jest *analiza funkcjonalna*, to znaczy definiowanie testów w oparciu o istniejącą specyfikację: przypadki użycia i wymagania. Zakładamy, że taka specyfikacja istnieje i jest na tyle dobrze napisana, że można na jej podstawie opracować testy. Trzeba pamiętać także o wymaganiach niefunkcjonalnych, których testowanie jest często zaniedbywane.**
- **Oprócz tego często posługujemy się *analizą ścieżek*, czyli dostępnych dla użytkownika sposobów przejścia przez aplikację. Nie muszą one koniecznie odpowiadać oczekiwanym sposobom jej użycia. Pozwoli to zbadać te sytuacje, w których użytkownik będzie działał w sposób „twórczy”, inaczej niż to przewidujemy.**

W wyborze danych do testowania pomoże *analiza wartości brzegowych* i *podział danych wejściowych na klasy równoważności*. Do klasy równoważności trafiają wszystkie te rodzaje danych, które powinny wywołać to samo działanie systemu. Na przykład jeśli hasło dostępu do naszego systemu nie może być krótsze niż 5 znaków, to wszystkie hasła krótsze niż 5 znaków trafiają do jednej klasy i są traktowane w ten sam sposób. Wystarczy wtedy testować tylko po jednym przedstawicielu każdej klasy równoważności, aby pokryć w ten sposób cały zakres wartości, które może podać użytkownik. Analiza wartości brzegowych pozwala sprawdzić jak zachowuje się aplikacja, gdy są jej przekazywane dane z poza dozwolonego zakresu. Wyobraźmy sobie, że nasza aplikacja nie zezwala na przyjęcie danych liczbowych mniejszych niż 10. Sprawdzamy wtedy co się dzieje w dozwolonym zakresie (np. dla 15), w zabronionym zakresie (np. dla 3), na granicy stosowalności, czyli dla wartości 10. Ponadto sprawdzamy co się stanie, gdy przekażemy aplikacji zamiast liczby litery, znaki specjalne itp.

- Często wykorzystywaną metodą jest *metoda tablic decyzyjnych*, pozwala ona zredukować liczbę kombinacji parametrów testowych tak, żeby pominąć testy, które są redundantne.
- Popularną techniką tworzenia testów jest *testowanie ad-hoc*. W tym przypadku polegamy nie na formalnych wytycznych co do sposobu testowania, a bardziej liczymy na intuicję i doświadczenie testerów czy programistów, którzy będą potrafili „wyczuć" co tak na prawdę należy przetestować. Mimo, że ta metoda jest niezbyt „naukowa" to często przy jej pomocy można opracować testy, których potrzebę trudno by było wykryć jakąś formalną metodą.
- Kolejną metodą testowania, także opierającą się na umiejętnościach i doświadczeniu testerów jest *metoda eksploracyjna*. W tym przypadku tester jednocześnie poznaje aplikację, tworzy testy i je wykonuje. Decydując się na takie podejście nie tworzymy uprzednio pełnego, formalnego planu testów, tylko powstaje on dynamicznie, w czasie bezpośredniej pracy z aplikacją.

Czynniki sukcesu

- **Określenie fragmentów systemu o szczególnych wymaganiach wobec niezawodności.**

- **Właściwa motywacja osób zaangażowanych w testowanie. Np. stosowanie nagród dla osób testujących za wykrycie szczególnie groźnych błędów, zaangażowanie osób posiadających szczególny talent do wykrywania błędów**

Rezultaty testowania

- Poprawiony kod, projekt, model i specyfikacja wymagań
- Raport przebiegu testów, zawierający informację o przeprowadzonych testach i ich rezultatach.
- Oszacowanie niezawodności oprogramowania i kosztów konserwacji.

Bezpieczeństwo oprogramowania

Bezpieczeństwo niekoniecznie jest pojęciem tożsamym z niezawodnością. System zawodny może być bezpieczny, jeżeli skutki błędnych wykonań nie są groźne.

Wymagania wobec systemu mogą być niepełne i nie opisywać zachowania systemu we wszystkich sytuacjach. Dotyczy to zwłaszcza sytuacji wyjątkowych, np. wprowadzenia niepoprawnych danych. Ważne jest, aby system zachował się bezpiecznie także wtedy, gdy właściwy sposób reakcji nie został opisany. Niebezpieczeństwo może także wynikać z awarii sprzętowych. Analiza bezpieczeństwa musi uwzględniać oba czynniki.

Ważne uwagi:

- **Nigdy nie możemy mieć 100% pewności, że nasze oprogramowanie nie zawiera błędów.**
- **Testy umożliwiają tylko zminimalizowanie ryzyka ich wystąpienia.**
- **Nie ma fizycznej możliwości sprawdzenia oprogramowania dla wszystkich możliwych danych.**
- **Każdy subiektywnie postrzega specyfikację wymagań. Właśnie specyfikacja opisuje jak powinno działać oprogramowanie. Jeśli program działa niezgodnie z tym co jest w niej zapisane to mamy do czynienia z błędem. Jeśli specyfikacja nie jest postrzegana obiektywnie to i błędy też nie są.**

Podsumowanie

Testowanie jest bardzo ważnym elementem tworzenia oprogramowania, ponieważ:

- **Może znacznie obniżyć koszty projektu, dzięki wczesnemu wykryciu błędów**
- **Wskazuje błędy w działaniu oprogramowania, błędne linie kodu**
- **Sprawdza stabilność systemu**

Źródła

- pl.wikipedia.org
- wazniak.mimuw.edu.pl
- www.users.pjwstk.edu.pl
- www.student-life.pl