Wykład 8

Testowanie w JEE 5.0 (1)

Autor: Zofia Kruczkiewicz

1. Rola testowania w tworzeniu oprogramowania

Kluczową rolę w powstawaniu oprogramowania stanowi proces usuwania błędów w kolejnych fazach rozwoju oprogramowania na drodze **testowania** (różne metody testowania dostosowane do stopnia rozwoju oprogramowania).

W inżynierii oprogramowania poszukuje się **związku** między strukturą programu a możliwością powstawania pewnych błędów oraz trudnością ich wykrywania na drodze testowania.

2. Definicje

- ***Atestowanie*** (validation) - testowanie zgodności systemu z rzeczywistymi potrzebami użytkownika (czy zbudowano poprawny produkt).
- ***Weryfikacja*** (verification) - testowanie zgodności systemu z wymaganiami zdefiniowanymi w fazie określania wymagań (czy zbudowano produkt poprawnie w kolejnych fazach życia)
- ***Błąd*** (fault, error, defect) jest niepoprawną konstrukcją znajdującą się w oprogramowaniu, która może prowadzić do niewłaściwego działania.
- ***Błędne wykonanie*** - **uszkodzenie** (failure) to niepoprawne działanie systemu w trakcie jego pracy na skutek błędów. Takie same błędne wykonanie może pochodzić od różnych błędów. Jednak błędy nie muszą powodować błędnego wykonania programu!

Klasyfikacja błędów

- **błędy wymagań i analizy**: złe sformułowanie problemu, zaniedbanie istotnych parametrów, niewłaściwy algorytm,
- **błędy projektowania:** błędna interpretacja wymagań, błędy logiczne
- **błędy programowe:**
  - **błędy opracowania** szczegółowej struktury programu: zła interpretacja wymagań dla programu, niepełność struktury programu, nie uwzględnienie przypadków szczególnych, niedostateczne dopracowanie błędów, zlekceważenie warunków czasowych
  - **błędy kodowania:**
    - syntaktyczne, zazwyczaj rozpoznawane przez kompilator,
    - błędy merytoryczne (nieprawidłowe korzystanie z indeksów i wskaźników, zły przydział pamięci, pominięcie inicjalizacji zmiennych, pomieszanie parametrów funkcji, błąd w pętlach, zamiana wyników decyzji w instrukcjach warunkowych, błędy deklaracji typów i wymiarów danych, błędy zakresów wartości danych,
  - **błędy kompilacji i konsolidacji:** błędy kompilatora, błędy w zakresach nazw itp.)

Klasyfikacja testów

1. **ze względu na cel:**
   - **1.1. testy wykrywające błędy**
   - **1.2. testy statystyczne**, określające przyczyny najczęstszych błędnych wykonań oraz ocena niezawodności systemu

2. ze względu na technikę wykonania:

- **2.1. testy dynamiczne** polegające na wykonaniu fragmentu lub całego programu i porównaniu wyników jego działania z wynikami poprawnymi. Możliwe jest wykonywanie „metaprogramów" wykonanych w różnych fazach powstawania oprogramowania:

  testy funkcjonalne: Program traktowany jest jak „czarna skrzynka". Znane są jedynie wymagania wobec testowanych funkcji programu. Testuje się program w wybranych podzakresach danych, traktując je jako klasy danych wejściowych – testy dla każdej klasy przeprowadza się jedynie dla pewnych wybranych danych w kilku przebiegach, a wnioskuje się o działaniu programu dla całej klasy danych.

  strukturalne (metaprogramy): Struktura programu jest znana. Dane wejściowe należy dobrać tak, aby każda instrukcja programu była przynajmniej raz wykonana, oraz tak, aby każda instrukcja warunkowa i pętle były przynajmniej raz wykonane i raz nie wykonane (***kryterium pokrycia instrukcji warunkowych***).

- **2.2. testy statyczne: inspekcje struktury produktu**, **udowadnianie poprawności programu (**np. logika Hoare**), testowanie symboliczne** (testowanie oparte na strukturze programu i analizowaniu stanu danych w wyniku wykonania programu dla różnych przebiegów sterowania programem (wykonanie lub nie wykonanie instrukcji warunkowych i pętli podczas przejścia przez program) – p. następny wykład)

Testy dynamiczne i statyczne mogą służyć do wykrywania różnych błędów.

Zofia Kruczkiewicz

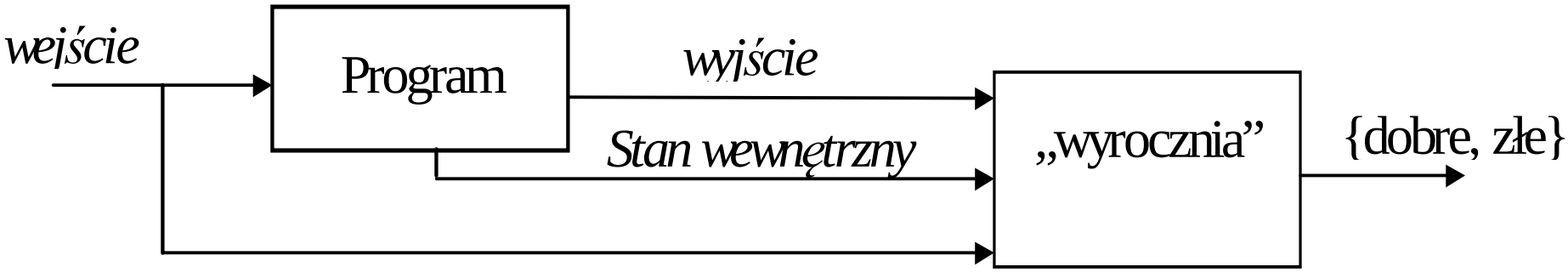
Ze względu na zakres wyróżnia się następujące testy:

- testy jednostkowe (modułów)
- testy integracyjne
- testy systemu
- testy akceptacji (testy alfa i beta).

Kolejność wykonania testów w procesie powstawiania oprogramowania jest zależna od przyjętej metody testowania i tworzenia oprogramowania

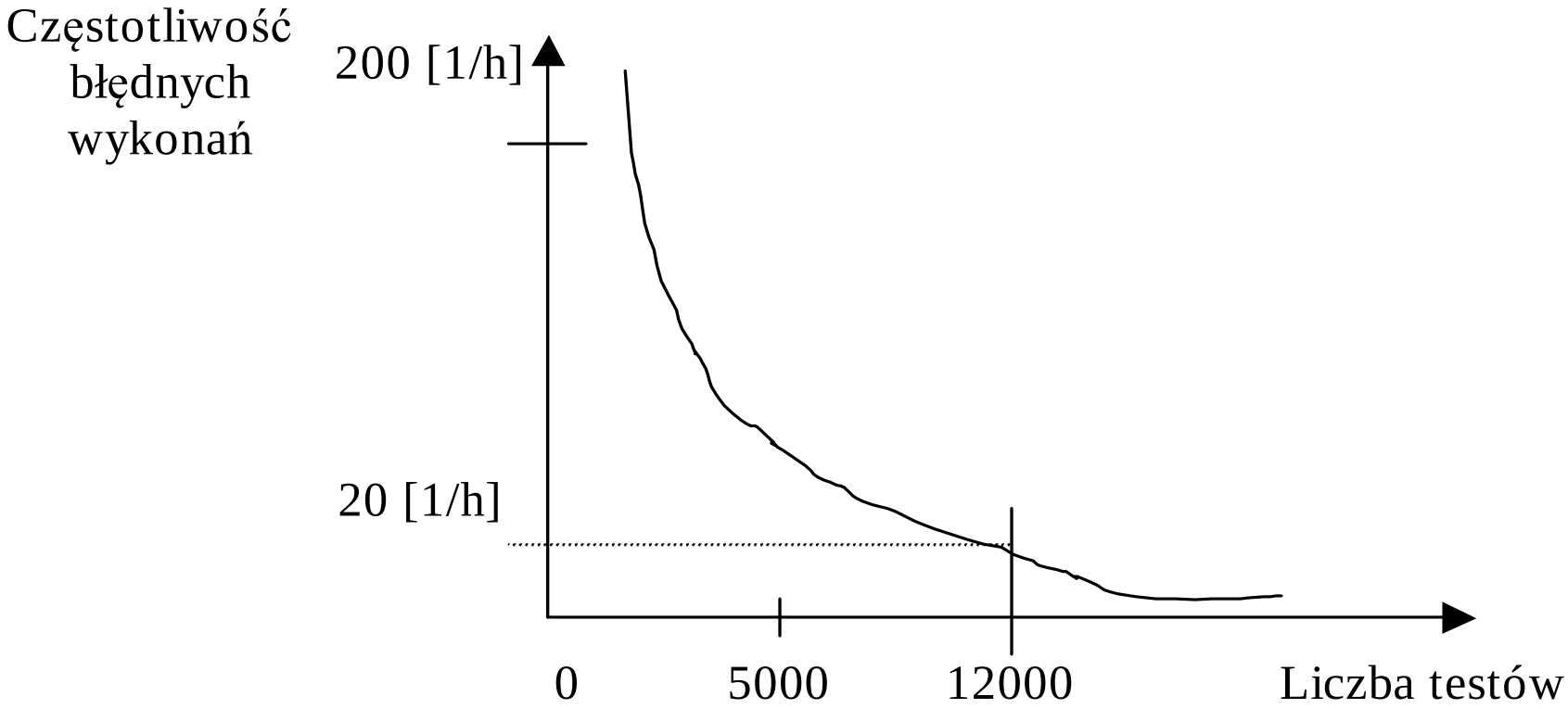
3. Testowalność (niezawodność)

[A.Bertolino,L.Strigini:On the Use of Testability Assessment,IEEE TRANSACTION ON SOFTWARE ENGINEERING,vol. 22, no. 2, February 1996]



Wyrocznia jest następującą funkcją:
Wyrocznia: D × R ×(zbiór_wartości_stanów_programu) →{dobry, zły}
gdzie
D - dziedzina danych wejściowych,
R - dziedzina danych wyjściowych
zbiór_wartości_stanów_programu– zbiór obserwowanych wartości zmiennych

Związek między liczbą przeprowadzonych testów i niezawodnością



Niezawodność programu jest częstotliwością jego błędnych wykonań. Rośnie ona logarytmicznie wraz ze wzrostem liczby przeprowadzonych testów.

4. Ocena liczby błędów metodą posiewania błędów

Na podstawie wszystkich znalezionych błędów oraz błędów sztucznie wprowadzonych do programu można oszacować liczbę błędów w programie.

N - liczba wprowadzonych błędów

- *M* - liczba wszystkich wykrytych błędów
- X - liczba wprowadzonych błędów, które zostały wykryte
- Szacunkowa liczba błędów przed wykonaniem testów:

$$Błędy_{calk} = \frac{(M - X) \bullet N}{X}$$

Liczba błędów po usunięciu wykrytych, w tym wszystkich sztucznie wprowadzonych:

$$Błędy_{Poz} = (M - X) \bullet (\frac{N}{X} - 1)$$

Współczynnik *X/N* opisuje efektywność wykonywanych testów.

Metody posiewania błędów:

- losowe zakłócenia w przypisywaniu danych
- losowe mutacje kodu - zmiany kodu źródłowego modyfikującego sterowanie lub dane w programie
- losowe zakłócenia między interfejsami modułów