

Testowanie oprogramowania

Wstęp

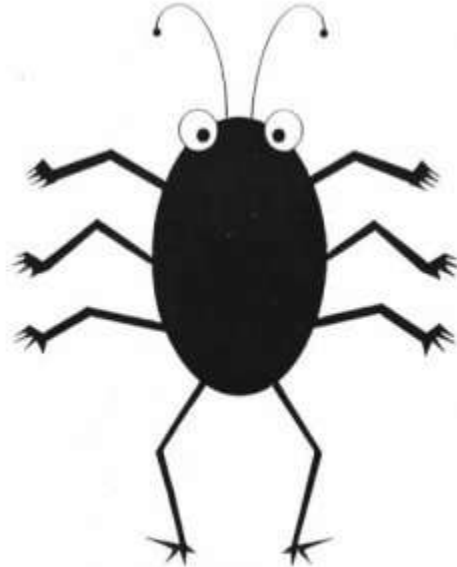
Cytaty

- *Kiedy zawiesz się program konkurencji, to jest awaria. Kiedy zawiesz się własny program, to jest „drobiazg”. Często po awarii pojawia się komunikat typu „ID 02”. „ID” to skrót od „idiotyczny drobiazg”, a następująca po nim liczba wskazuje, przez ile miesięcy produkt informatyczny powinien być jeszcze testowany*

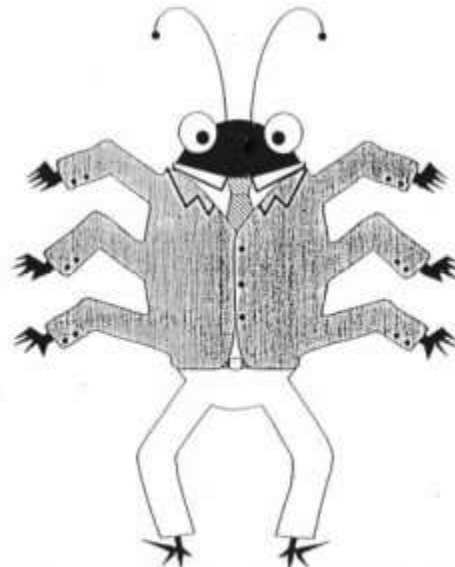
Guy Kawasaki, The Macintosh Way

- *Uwielbiam ostateczne terminy. Zwłaszcza podoba mi się świst, jaki wydają kiedy nas mijają.*
- Douglas Adams, „The Hitch Hiker’s Guide to the Galaxy*

Bug, skąd się wziął



BUG



FEATURE

Kilka definicji

- Pomyłka, **błąd** (mistake, error): Działanie człowieka powodujące powstanie nieprawidłowego wyniku
- Defekt, usterka, pluskwa (defect, fault, **bug**): skutek błędu twórcy oprogramowania. Usterka może, ale nie musi spowodować awarię.
- **Awaria** (failure): odchylenie od spodziewanego zachowania albo wyniku działania oprogramowania

Sukces vs porażka

Sukces

- Wykrycie błędów w procesie testowania

Porażka

- Sytuacja odwrotna

Analogia: pacjent z subiektywnymi objawami, wykonanie badań. Dobre wyniki laboratoryjne (testy) są porażką => nie wiadomo co pacjentowi dolega.

Dwa zadania testowania

Weryfikacja

- Zgodność produktu z jego bezpośrednią specyfikacją

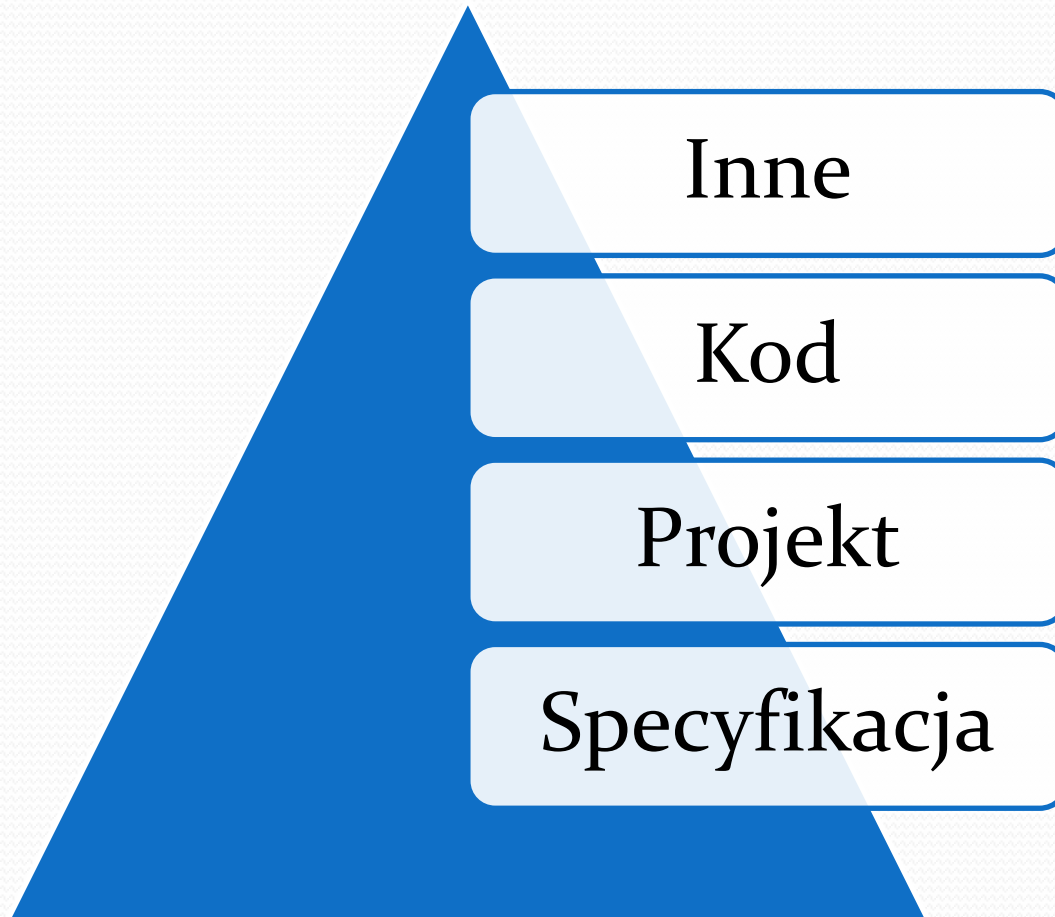
Walidacja

- Zgodność produktu ze specyfikacją wymagań lub – wobec jej braku – z faktycznymi oczekiwaniami klienta

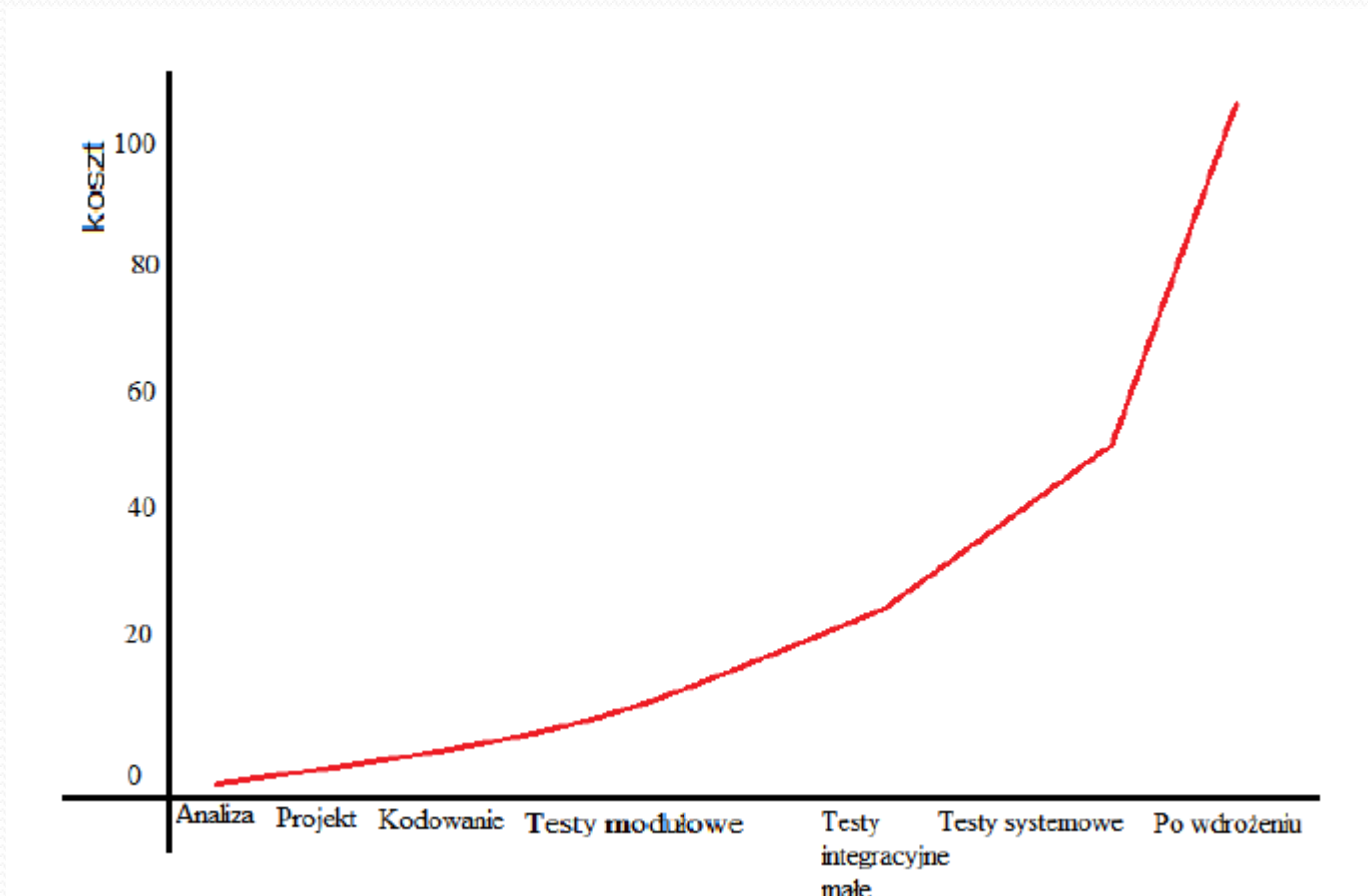
„„Błąd”” oprogramowania

1. Oprogramowanie nie wykonuje czegoś, co wg specyfikacji powinno
2. Oprogramowanie robi coś, czego nie powinno 😊
3. Oprogramowanie robi coś, o czym specyfikacja nie wspomina.
4. Oprogramowanie nie robi czegoś, czego nie ma w specyfikacji, ale być powinno.
5. Oprogramowanie jest trudne do zrozumienia, trudne do użycia, powolne – albo zdaniem testera „nieprawidłowe”

Kiedy powstają błędy?



Koszty usuwania błędów



Jaki jest cel testowania?

Nadrzędnym celem
testowania jest
odnajdywanie błędów ...
do tego jak najwcześniej!

Cechy dobrego testera

- Być odkrywczą
- Być łowcą problemów
- Być nieustępliwym
- Być twórczym
- Być perfekcjonistą... dojrzałym perfekcjonistą
- Być rozsądnym w podejmowaniu decyzji
- Być taktownym i dyplomatycznym
- Umieć przekonywać
- Dodatkowo: mieć przeszłość programistyczną

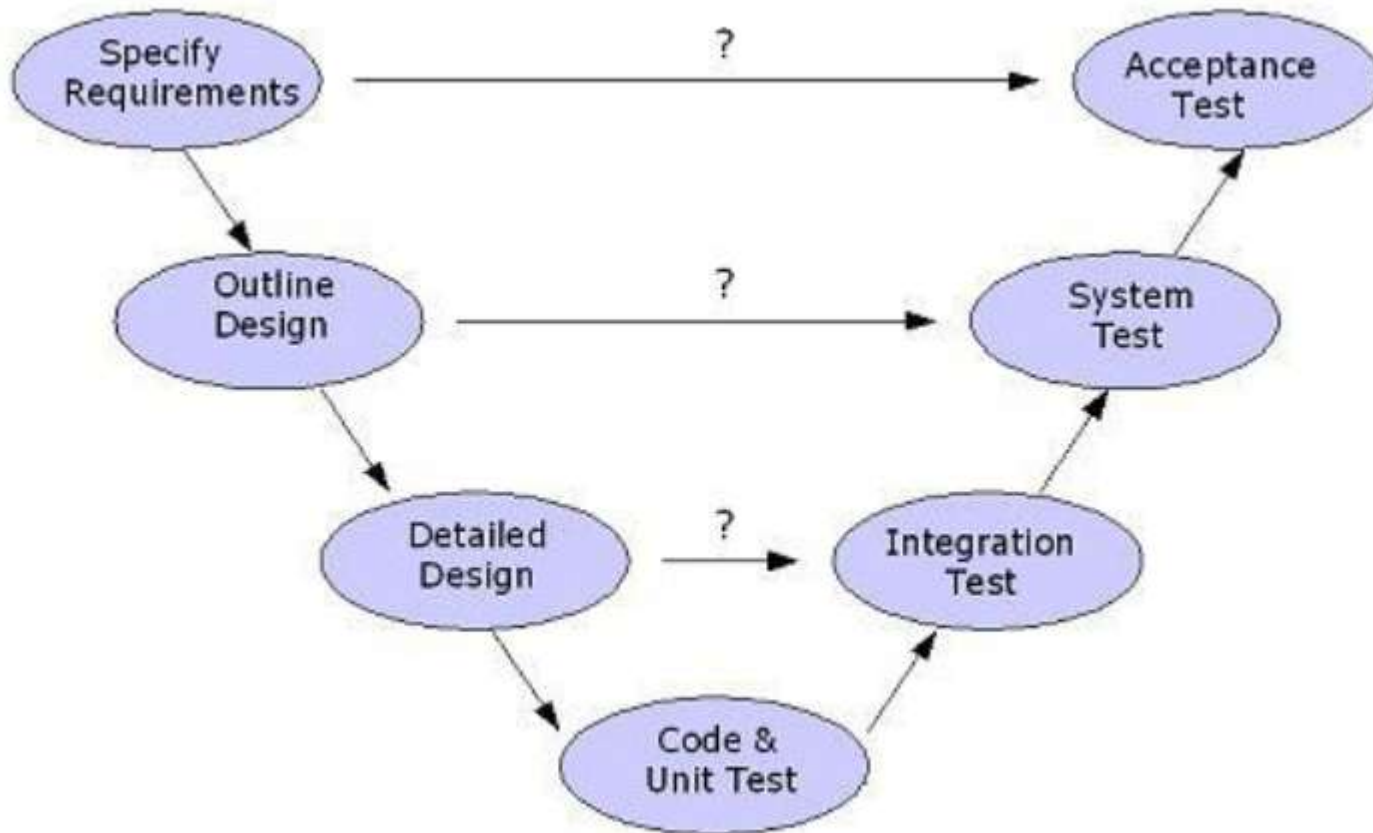
Co tak naprawdę składa się na „produkt informatyczny”?

- Pliki pomocnicze
- Pliki programu
- Próbkki i przykłady
- Komunikaty o awariach
- Konfiguracja i instalacja
- Podręcznik użytkownika
- Etykiety
- GUI
- Ogłoszenia i reklamy
- ...

„Ciekawe” błędy

- Błąd: brak klawiatury. Wciśnij F1 aby kontynuować
- Windows znalazł nieznaną urządzenie i instaluje dla niego sterowniki
- Nastąpił błąd krytyczny 006 pod adresem 000:00000007

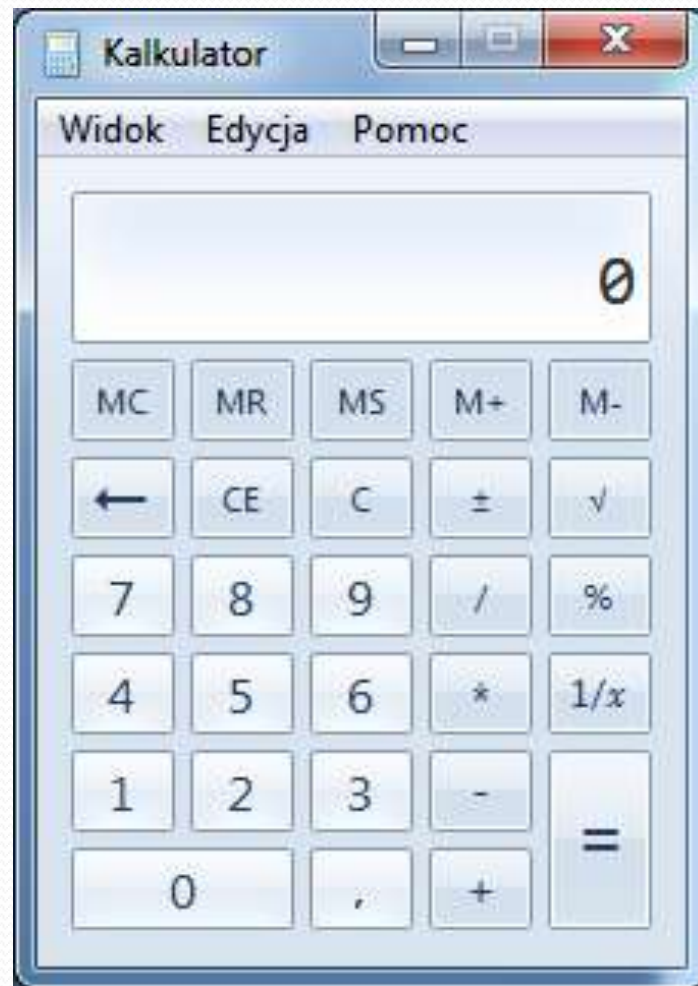
Model V



Testowanie totalne

- Programu nie da się przetestować całkowicie:
 - Dane wejściowe są ogromne
 - Dane wyjściowe jeszcze większe
 - Ścieżki działania programu są niezliczone
- Test nigdy nie udowodni braku błędów

Case Study: Kalkulator



Czy zawsze należy naprawiać błędy?

- Brak czasu
- Właściwie to nie jest błąd...
- Zbyt wielkie ryzyko naprawy 😊
- Po prostu nie warto 😊 😊 😊

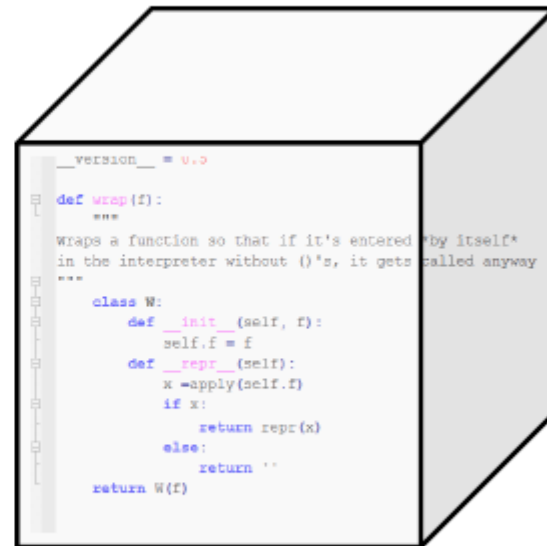
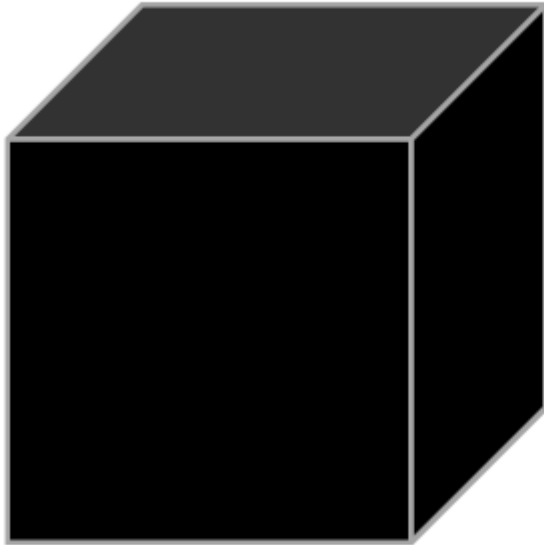
Typy testów

- Ze względu na „zakres”
 - Modułowy (*uwaga –kumulacja!!*)
 - Integracyjny mały
 - Systemowy
 - Integracyjny duży
 - Akceptacyjny
- Ze względu na „użycie kodu”
 - Statyczne (kod nie jest wykonywany –bez komputera)
 - Dynamiczne (kod jest wykonywany)

Typy testów c.d.

- Funkcjonalne
 - Wymagania (specyfikacja)
 - Proces biznesowy
- Niefunkcjonalne
 - Wydajnościowe
 - Użyteczności
 - Bezpieczeństwa

Testy



```
__version__ = 0.5  
  
def wrap(f):  
    """  
    Wraps a function so that if it's entered *by itself*  
    in the interpreter without ()'s, it gets called anyway  
    """  
  
    class W:  
        def __init__(self, f):  
            self.f = f  
        def __repr__(self):  
            x = apply(self.f)  
            if x:  
                return repr(x)  
            else:  
                return ''  
    return W(f)
```

Ćwiczenia

1. Zaplanuj jakiego rodzaju testy należałoby wykonać dla Twojej aplikacji (co trzeba przetestować, którą metodą, w jaki sposób, itp.)
2. Nadaj priorytety testom zaplanowanym testom.
3. Rozpoczynając od najbardziej priorytetowego, napisz scenariusze testowe tak, jak uważasz że powinny wyglądać.
4. **Jeśli musisz, to dzisiaj testuj tylko 'ad hoc'. Zaplanuj testy, ale nie wykonuj ich jeszcze.**
5. **Z dotychczasowych sprawozdań z iteracji wykonajcie dokumentację, którą można będzie przekazać innemu zespołowi projektowemu wraz z aplikacją do przetestowania.**