Podstawy testowania oprogramowania

- 3 dni
- Zgodne z sylabusem ISTQB
- Autor: Bogdan Bereza
- bogdan.bereza@victo.eu

Materiały szkoleniowe - **część 2 (2)**
Wersja 1.3

ISTQB®
Certified Tester
Foundation Level

Według:

- **Certyfikowany tester, plan poziomu podstawowego**, wersja 2011.1.1 (sjsi.org)
- **Certified Tester, Foundation Level Syllabus**, version 2011 (istqb.org)

Yaron Tsubery

Chris Carter

Eric Riou du Cosquer

Mitko Mitev

Spis treści – program kursu

Materiały – część 1

Materiały – część 2

4. Techniki projektowania testów

- Testowanie w inżynierii oprogramowania
- Wstęp do certyfikacji oraz ISTQB

1. Podstawy testowania
2. Testowanie w cyklu życia oprogramowania
3. Statyczne techniki testowania
4. **Techniki projektowania testów**
5. Zarządzanie testowaniem
6. Testowanie wspierane narzędziami

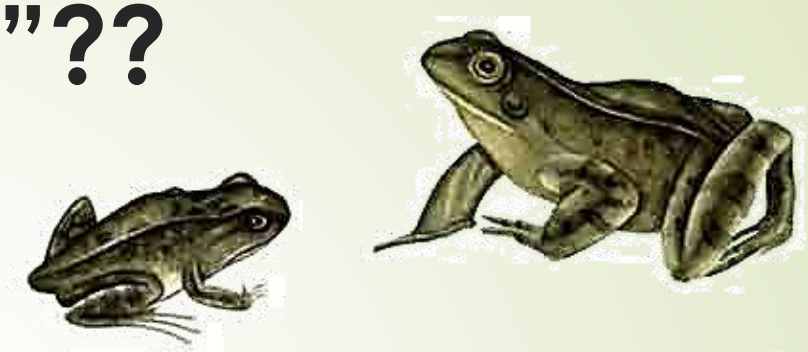
4. Techniki projektowania testów

4.1 Proces rozwoju testów
4.2 Kategorie technik projektowania testów
4.3 Techniki czarnoskrzynkowe
4.4 Techniki białoskrzynkowe
4.5 Na podstawie doświadczenia
4.6 Wybór techniki

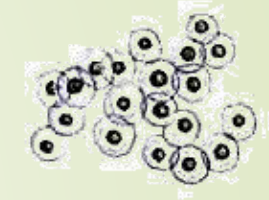
4.1 Proces rozwoju testów

Development = „rozwój”??

- Rozwój, rozwinięcie, zabudowa, **rozbudowa**, zagospodarowanie, **tworzenie**
- „Rozwój” po polsku to rozwój biologiczny, psychiczny, osobowy, gospodarczy, społeczny i technologiczny (postęp)



Fazy rozwoju przypadku testowego?



4.1 Proces rozwoju testów

- Mniej lub bardziej zdefiniowany i sformalizowany
- Mniej lub bardziej staranny
- Mniej lub bardziej sekwencyjny lub iteracyjny
- Mniej lub bardziej dokumentowany

Wracamy do procesu testowego

Planowanie i nadzór

Analiza i projektowanie

Tutaj tworzymy testy

Implementacja i wykonanie

Ocena kryteriów zakończenia i raportowanie

Czynności zamykające

Uwaga: to nie oznacza, że te czynności muszą występować sekwencyjnie: do analizy i projektowania można (nawet należy!) powracać także podczas wykonywania, oceny kryteriów zakończenia itp.…

Jak to przebiega?

Podstawa testowa

Analiza i projektowanie

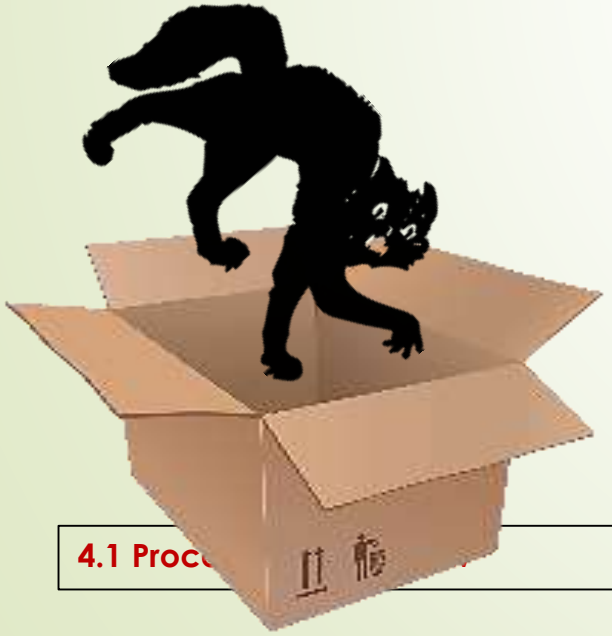
Warunki testowe

Jak projektować?

- Jest wiele sposobów i technik
- Niektóre tutaj omówimy
- Wybór odpowiedniej techniki w pewnym stopniu zależy od ryzyka…
- … ale brak systematycznej wiedzy, jakie techniki projektowania należy stosować dla jakiego ryzyka – tylko HEURYSTYKI są

Przypadek testowy

- Warunki wstępne
- **Dane wejściowe**
- **Wyniki oczekiwane**
- Warunki zakończenia



Wyniki oczekiwane

- Dane wyjściowe
- Zmiany danych
- Zmiany stanu systemu
- Inne skutki
- Jak głęboko sprawdzać?
- Wyniki oczekiwane – lepiej z góry

Dalsza praca podczas implementacji

- Przypadki doprecyzowujemy w **procedury**
- Automatyczne procedury – **skrypty**
- Zestawy procedur – **zestawy**
- Tworzenie **harmonogramu wykonywania testów** na podstawie priorytetów oraz innych zależności

4.2 Kategorie technik projektowania testów

4.1 Proces rozwoju testów
4.2 Kategorie technik projektowania testów
4.3 Techniki czarnoskrzynkowe
4.4 Techniki białoskrzynkowe
4.5 Na podstawie doświadczenia
4.6 Wybór techniki

4.2 Kategorie technik projektowania

- Czarna skrzynka = wg specyfikacji = funkcjonalne = **biznesowe**
- Biała skrzynka = strukturalne = na podstawie struktury = **techniczne**
- I jedne, i drugie mogą być bardziej lub mniej **formalne** (i nieformalne)
- I trzeci podział wg podstawy: metoda – intuicja – przypadek - **doświadczenie**

Jeszcze o technikach projektowania

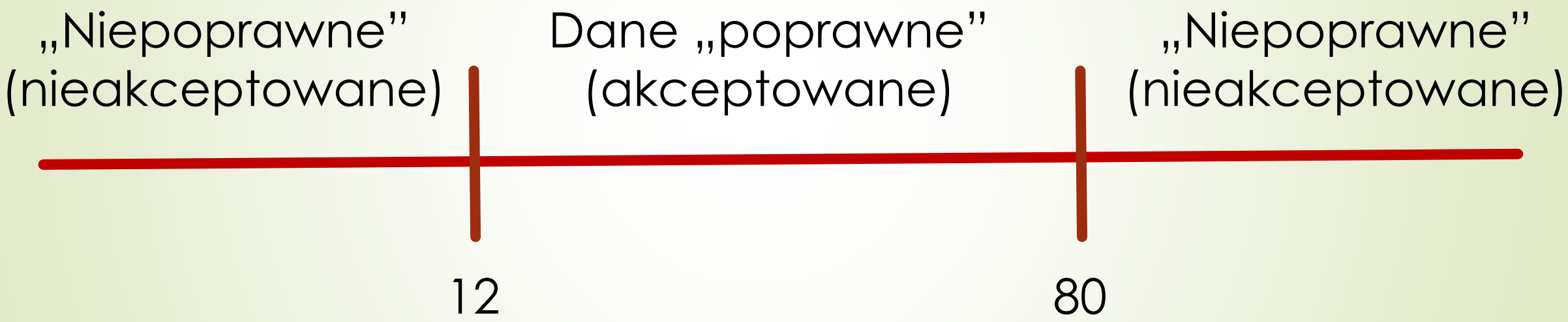
- ***Te same techniki mogą być stosowane czarno- i biało-skrzynkowo (np. pokrycie grafu modelu)!!***
- Miary pokrycia są częścią technik (dokładne – technik formalnych)
- ISTQB bzdurzy, że **miary pokrycia** tylko dla technik strukturalnych
- … i że **modele** tylko dla czarnej skrzynki

4.3 Techniki czarnoskrzynkowe

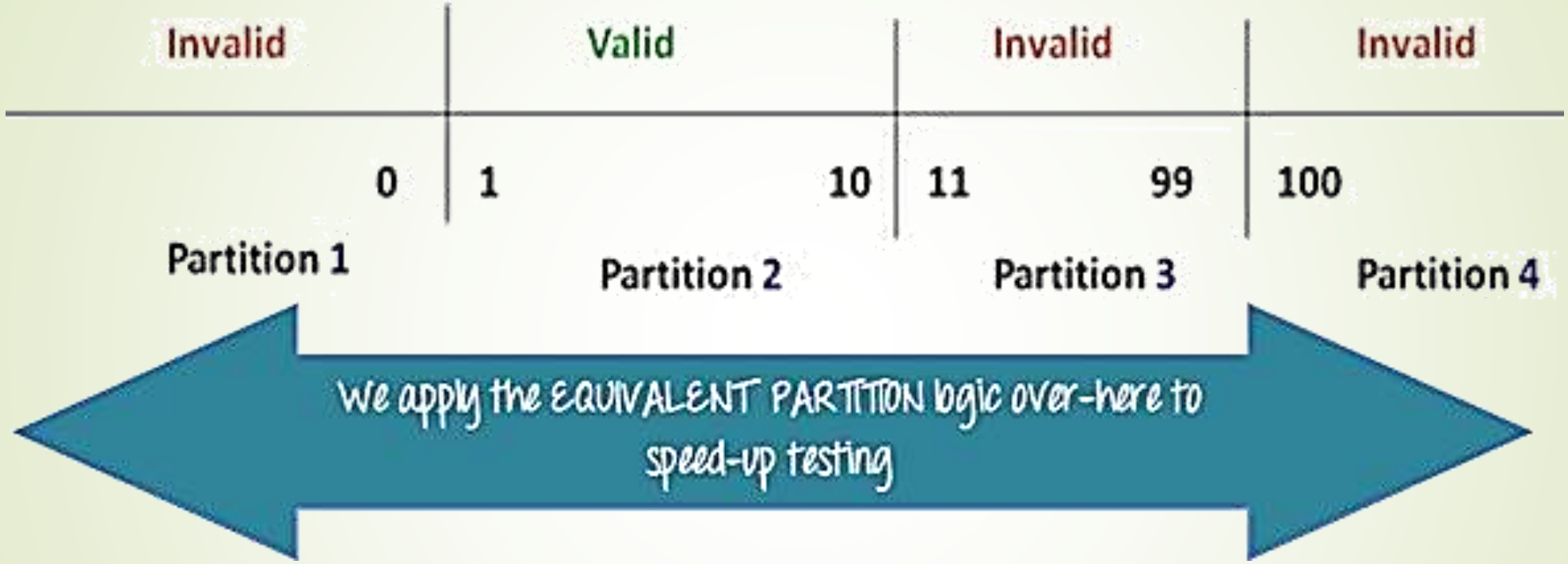
4.1 Proces rozwoju testów
4.2 Kategorie technik projektowania testów
4.3 Techniki czarnoskrzynkowe
4.4 Techniki białoskrzynkowe
4.5 Na podstawie doświadczenia
4.6 Wybór techniki

4.3.1 Podział na klasy równoważności

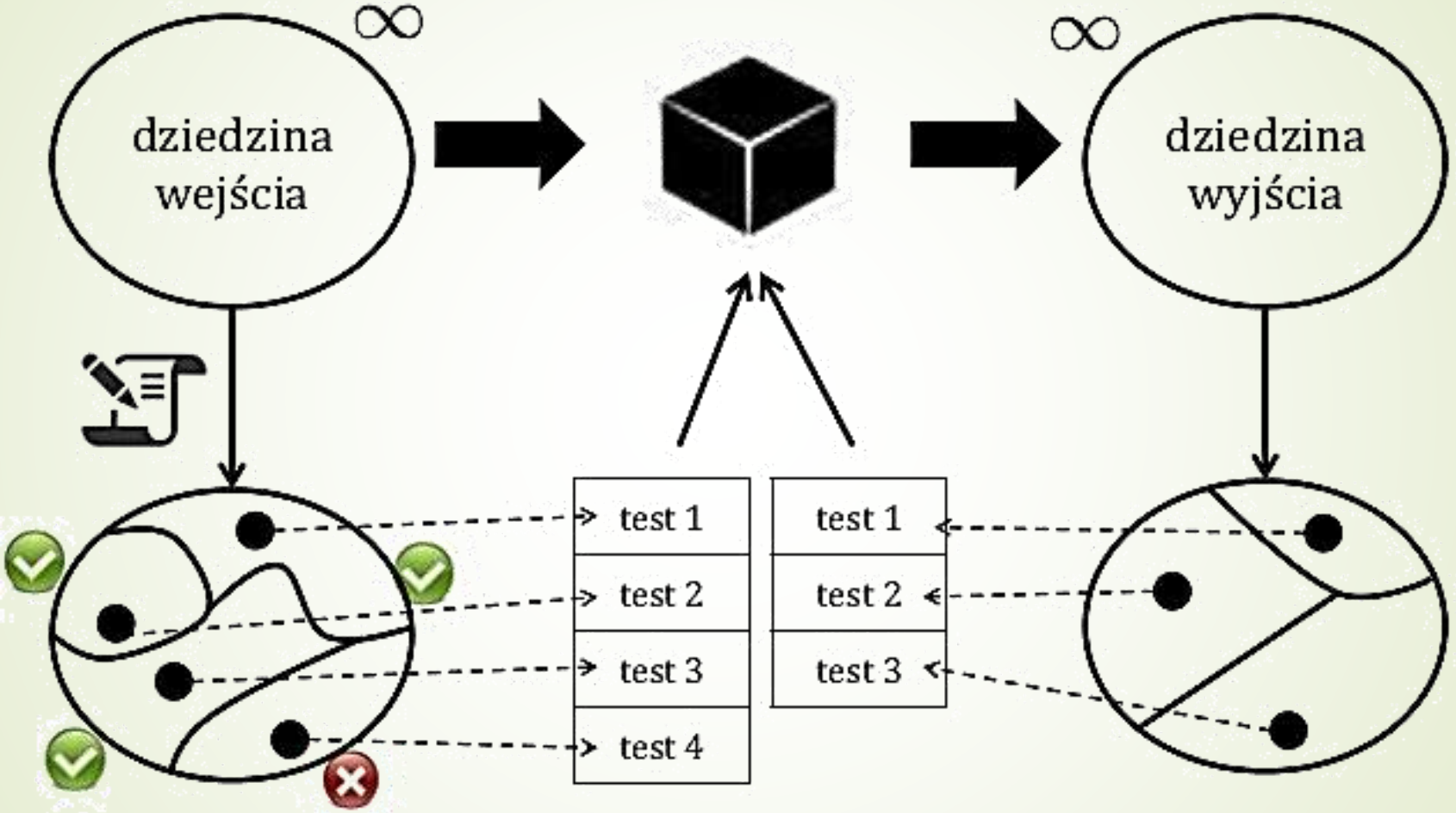
- „Wstęp mają osoby od 12 do 80 roku życia włącznie”



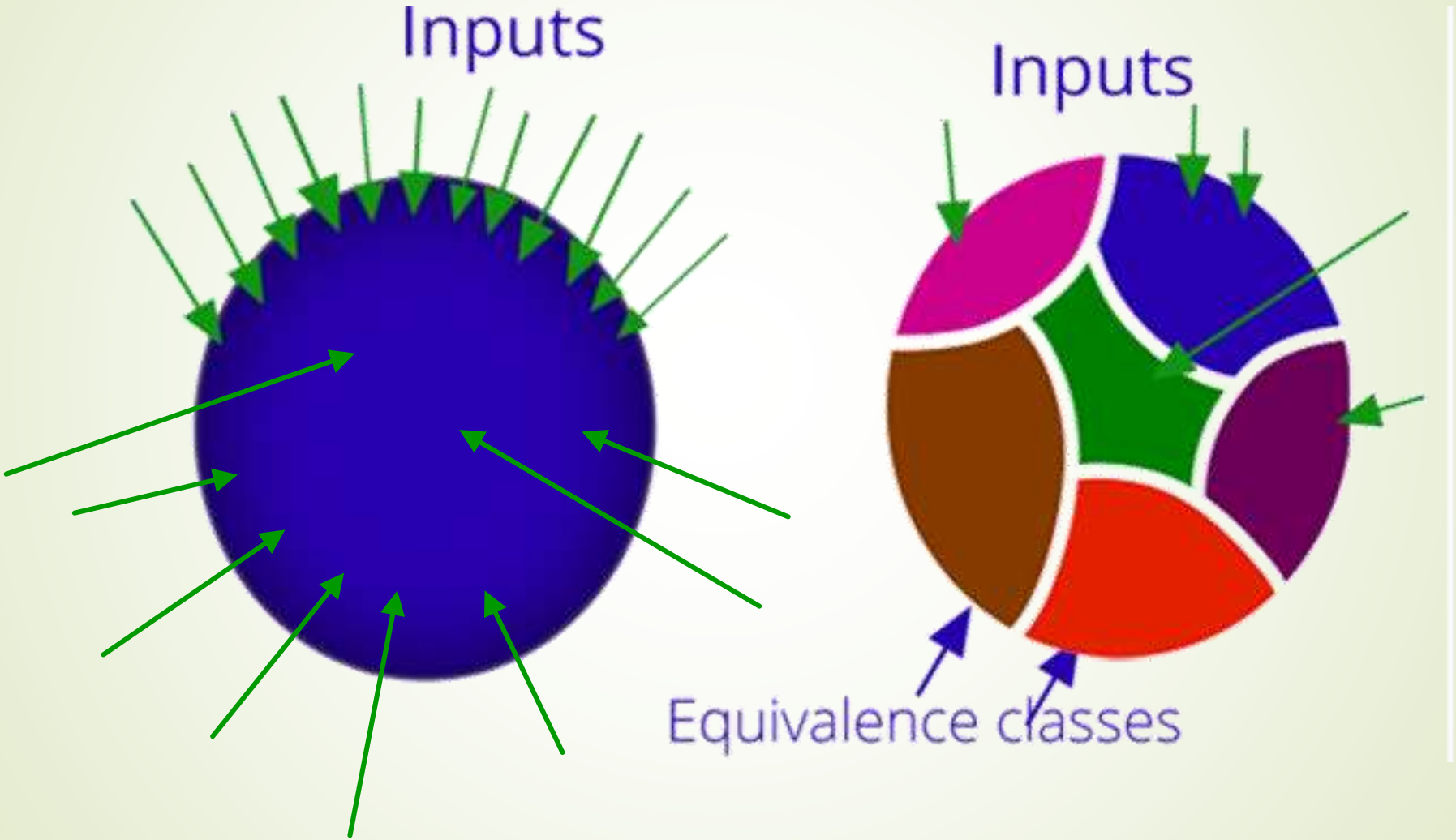
Klasy równoważności – przykłady 1(2)



Klasy równoważności – przykłady 2(2)



Po co klasy równoważności?



Zastosowanie klas równoważności

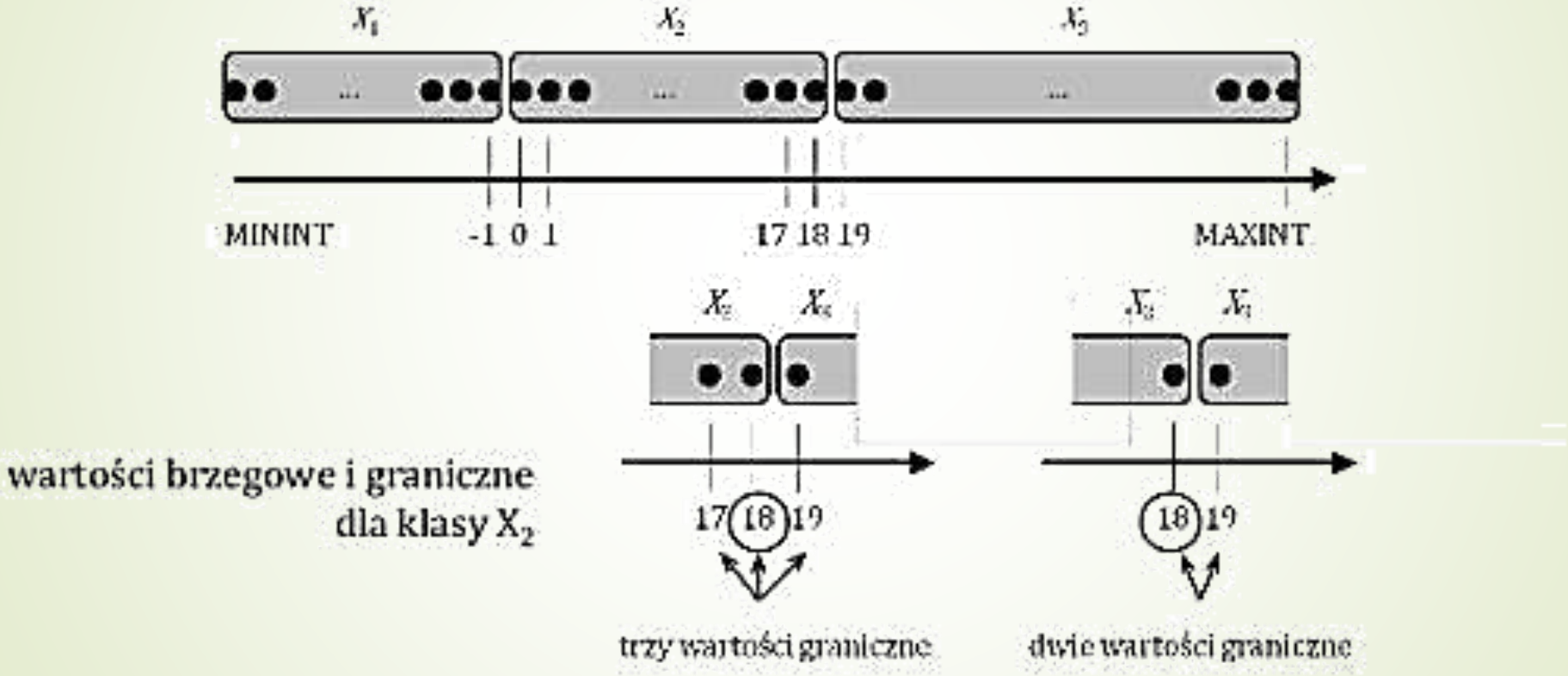
- Zastosowanie techniczne
- Zastosowanie biznesowe – testy „negatywne”
- Korzyści
- Kiedy taki podział nie daje zysków?
- Niebezpieczeństwa
- Dane wejściowe i wyjściowe
- Klasy wielowymiarowe

4.3.2 Analiza wartości brzegowych

- „[…] osoby od 12 do 80 roku życia **włącznie**”
- 12 i 80 to „**poprawne wartości brzegowe**” ☹
- 11 i 81 to „**niepoprawne wartości brzegowe**”
- Testuje się zwykle wszystkie… trzy ☺

AWB (nie ABW!) – przykłady 1(3)

- to największa i najmniejsza wartość klasy równoważności
- stosuje się zarówno dla klas wejść jak i dla wyjść programu



AWB – przykłady 2(3)

- rejestracja osoby w przedziale wiekowym 0 – 120,

testowane wartości brzegowe: {-1, 0, 1, 119, 120, 121}

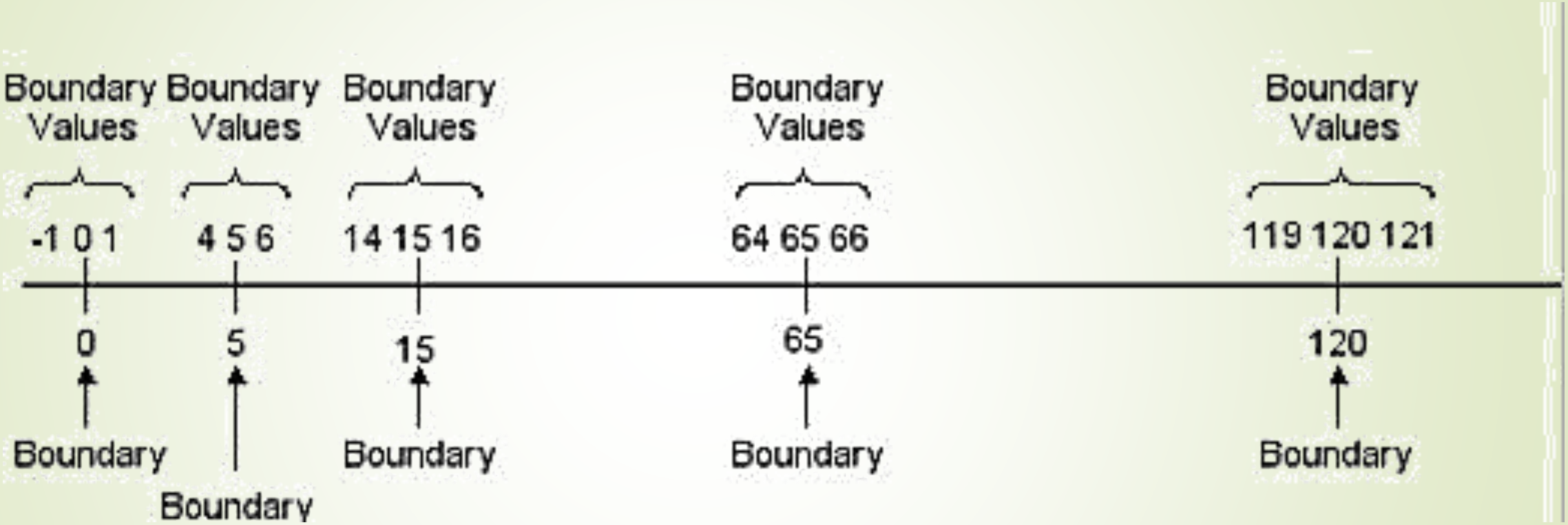
- długość wiadomości od 1 do 50 znaków:

testowane wartości brzegowe: {0, 1, 2, 49, 50, 51}

- napięcie od 0 do 100 V:

testowane wartości brzegowe: {-1, 0, 1, 99, 100, 101}

AWB – przykłady 3(3)



Psychologia ☺ wartości brzegowych

- Niejasne wymagania
- Pomyłki kodowania
- Gromadzenie się bugów „na brzegach”?
- Zmiana trybu pracy programu po przekroczeniu wartości granicznej
- Testy te są skuteczne zarówno ze względu na prawdopodobieństwo, jak i konsekwencje bugów

4.3.3 Testowanie wg tablicy decyzyjnej

- „Tablica decyzyjna” (ang. *decision table*) jest techniką modelowania zależności, gdzie **warunek** (zwykle złożony) powoduje **skutek** (wynik, działanie)
- Inny sposób modelowania takich zależności, to **grafy przyczynowo-skutkowe**
- Czyli tak naprawdę, to sposób… modelowania wymagań ☺

Testy tablicy decyzyjnej – przykłady 1(3)

Printer troubleshooter

		Rules							
Conditions	Printer does not print	Y	Y	Y	Y	N	N	N	N
	A red light is flashing	Y	Y	N	N	Y	Y	N	N
	Printer is unrecognized	Y	N	Y	N	Y	N	Y	N
Actions	Check the power cable			X					
	Check the printer-computer cable	X		X					
	Ensure printer software is installed	X		X		X		X	
	Check/replace ink	X	X			X	X		
	Check for paper jam		X		X				

Testy tablicy decyzyjnej – przykłady 2(3)

Wielkość zamówień powyżej 40000 PLN	T	T	T	T	N	T	N	T	N	N	N	N
Termin realizacji od 5 do 10 miesięcy	T	N	N	T	T	N	N	N	T	N	N	N
Termin realizacji powyżej 10 miesięcy	N	T	N	N	N	T	T	N	N	N	T	N
Ogólne obroty powyżej 1.5 mln rocznie	T	T	T	N	T	N	T	N	N	T	N	N
Opust cenowy 0%												X
Opust cenowy 1%										X	X	
Opust cenowy 2%								X	X			
Opust cenowy 3%						X	X					
Opust cenowy 4%				X	X							
Opust cenowy 5%			X									
Opust cenowy 6%		X										
Opust cenowy 7%	X											

Jakich testów tutaj być może brakuje?

4.3 Techniki czarnoskrzynkowe

Testy tablicy decyzyjnej – przykłady 3(3)

Problem : Dokonanie zapisu rezerwacji usługi turystycznej w magazynach danych

1 Rodzaj rezerwacji = wycieczka? 2. Czy komplet danych rezerwacji?.	T T	T N	N T	N N
1.Dopisz rezerwacje do magazynu „Rezerwacje”	X	-	X	-
2. Wyszukaj zapis wycieczki wg symbolu katalogowego	X	-	-	-
3. Zmodyfikuj ‘liczba wolnych miejsc”= ‘liczba wolnych miejsc” - 1	X	-	-	-
4. Wyszukaj zapis tygodnia pobytu wg symbolu apartamentu i daty początku	-	-	X	-
5. Zmodyfikuj atrybut Czy zarezerwowany=tak	-	-	X	-
6.Zapisz nr rezerwacji w atrybucie nr rezerwacji	-	-	X	-
7. Wyświetl formularz „kompletowanie danych	-	X	-	X

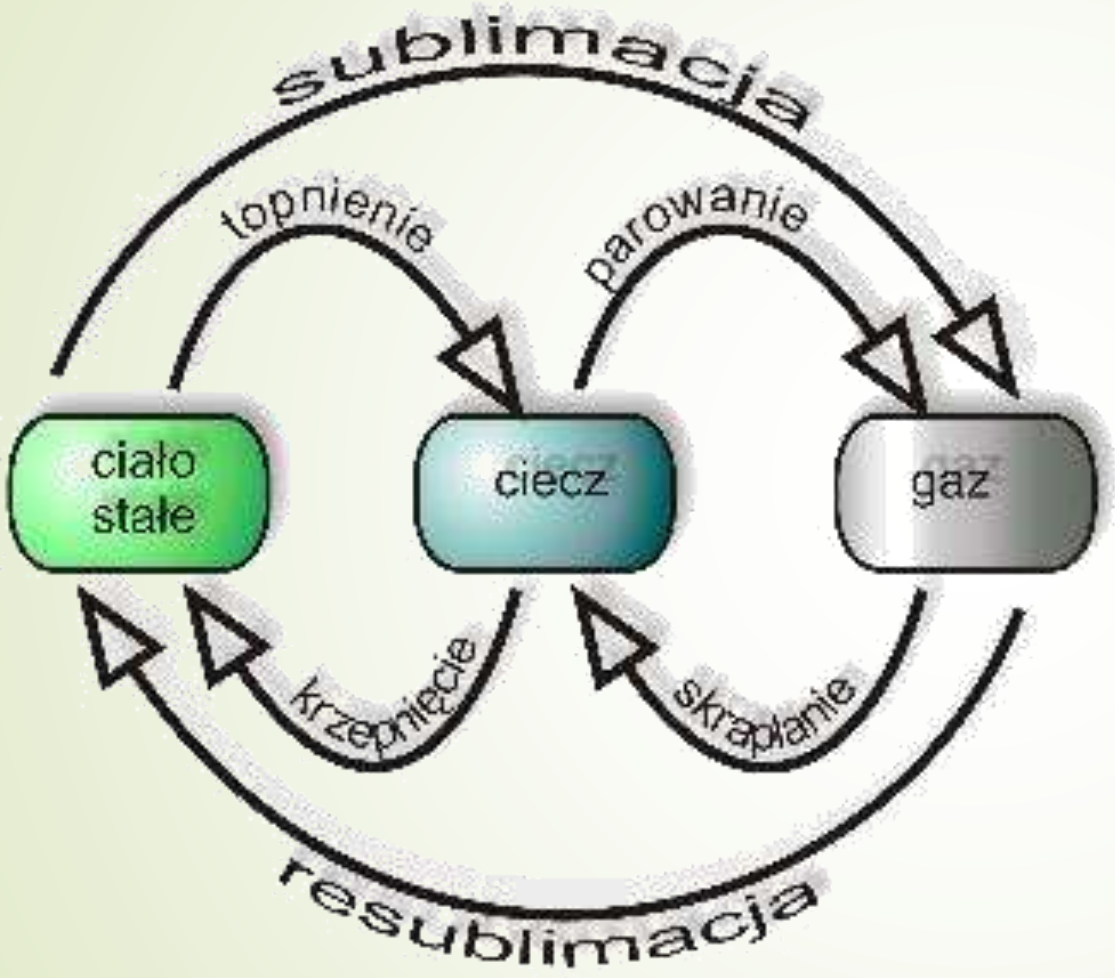
Tablice decyzyjne - podsumowanie

- To sposób opisania **reguł biznesowych**
- Wszystkie **kombinacje warunków** mogą być niemożliwe (wtedy są pominięte w tablicy), ale można próbować je przetestować
- Zaleta: **uwidocznienie** kombinacji, które inaczej łatwo byłoby pominąć

4.3.4 Testowanie przejść stanów

- Ponownie: tak zwane „diagramy przejść stanów" (diagramy maszyny stanów, automaty skończone…) to **sposób modelowania / opisywania wymagań** – nie technika testowania!
- Istnieje – wywodzących się z **teorii grafów** – szereg systematycznych sposobów projektowania testów w tych modeli

Testy przejść stanów – przykład 1(2)



Testy przejść stanów – przykład 2(2)

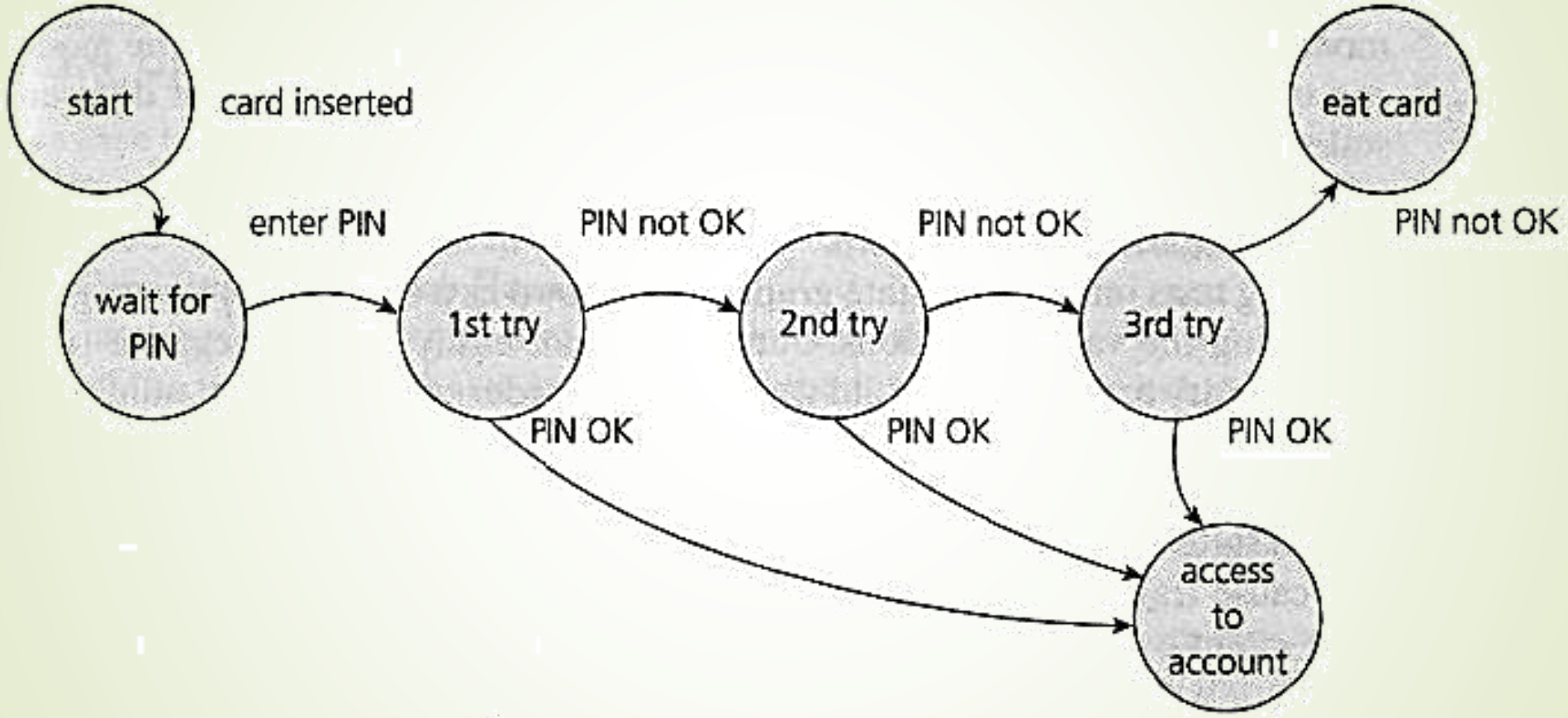


FIGURE 4.2 State diagram for PIN entry

Testy z diagramu przejść stanów

- Przejście (%) wszystkich stanów
- Przejście (%) wszystkich przejść między stanami (szatańska nazwa „pokrycie 0-przełącznikowe", ang. *zero-switch coverage*)
- Pokrycie (%) wszystkich ścieżek przejść między stanami
- Kombinacje bodźców – itd.

4.3.5 Testowanie wg przypadków użycia

- **Czemu akurat przypadków użycia??? ☹**
- UML ma 14 modeli w trzech grupach (modele działania, struktury oraz interakcji)
- Modele danych (przepływu danych, encji)
- Diagramy składni
- Wszystkie miary pokrycia grafów
- Diagramy BPMN →

→ więcej projektowania testów

- Testy statystyczne
- Testy kombinatoryczne, w tym „testy wszystkich par”
- Testy według innych opisów wymagań
- **Więc dlaczego ISTQB uczepiło się akurat tych czterech – klasy równoważności, tablice decyzyjne, diagramy stanów oraz przypadków użycia???**

4.4 Techniki białoskrzynkowe

4.1 Proces rozwoju testów
4.2 Kategorie technik projektowania testów
4.3 Techniki czarnoskrzynkowe
4.4 Techniki białoskrzynkowe
4.5 Na podstawie doświadczenia
4.6 Wybór techniki

Testy na podstawie wiedzy technicznej

- Wiedza powierzchowna i szczegółowa (przykłady)
- Dla systemów IT – wiedza z ogromnego zakresu obszarów (przykłady)
- Dla wbudowanych – także mechanika, chemia, elektryczność…
- **NIE TYLKO, A NAWET NIE PRZEDE WSZYSTKIM, POKRYCIE KODU ŹRÓDŁOWEGO!**

Black and white ☺

- Techniki, omówione w 4.2 jako „czarno-skrzynkowe", można świetnie stosować dla projektowania testów technicznych ☺
- Testy projektowane na podstawie kodu źródłowego, **to nie tylko testy pod kątem pokrycia**, lecz także np. testy pętli, testy parametrów, testy wyników operacji arytmetycznych, testy pojemności (struktur, list, pamięci) itp. itd.

Miary pokrycia testowego

- Miary **pokrycia testowego**, czyli miary staranności testów – jaki odsetek z ***czegoś***, co można przetestować, został przetestowany
- Często wynosi zero, jeśli *czegoś* jest ∞
- Mogą dotyczyć: wymagań, ryzyka (zagrożeń), funkcji, procesów biznesowych, elementów struktury systemu, struktury kodu, wartości danych, zbiorów danych…

Miary pokrycia - zastosowania

- Jakiś **pomiar staranności testów** (ale mocno niepewne są kryteria, jaka staranność, mierzona jaką miarą pokrycia, jest dostateczna! – nawet zależnie od poziomu ryzyka)
- Jakaś **metoda projektowania**: aby osiągnąć dostateczne pokrycie, aby przetestować również to, czego nie testujemy (a co ujawnił pomiar pokrycia)

Pokrycie *kodu źródłowego* programu

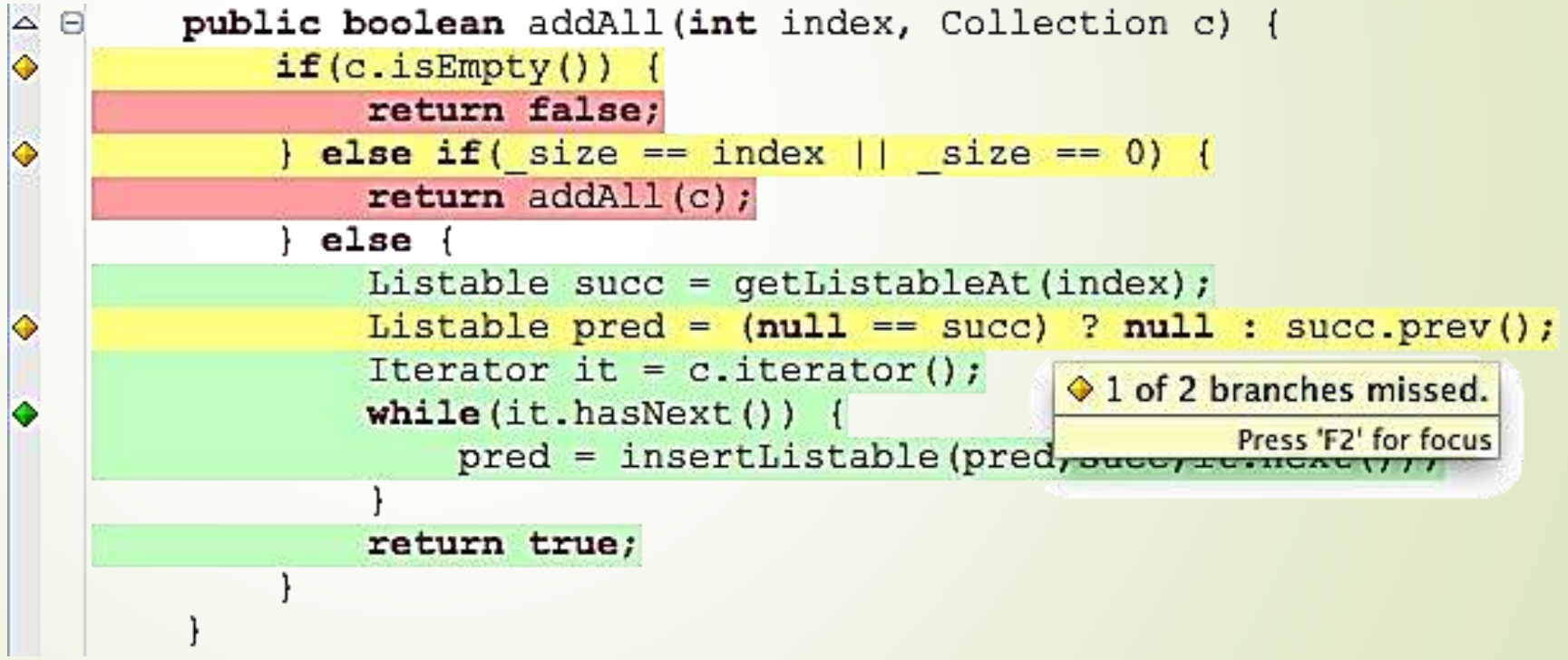
- Pomiar – wyłącznie automatyczny
- Konkluzje (jest OK, czy testować więcej?) – wyłącznie heurystyki
- Dodatkowe korzyści: świadomość ograniczeń testów, wykrywanie martwego kodu, identyfikacja kodu trudnego (niemożliwego) do przetestowania dynamicznego

Pokrycie kodu - trudności

- Jest inwazyjne (zwiększa *efekt próbnika*)
- Powoduje 2-3 wzrost czasu wykonywania testów
- Powoduje, że testy wykonuje się dodatkowo 3-4 razy
- Daje fałszywe poczucie bezpieczeństwa
- Czasem budzi opór programistów

Pokrycie kodu – jak to się robi?

- Instrumentacja kodu
- Wykonanie testów
- Wyniki:
- Wnioski?



4.4.1 Według pokrycia instrukcji

```
if (a > b)
  printf (“a większe niż b”);
else
  printf (“a jest nie > niż b”);
if (a == 5)
  printf (”a równa się 5”);
```

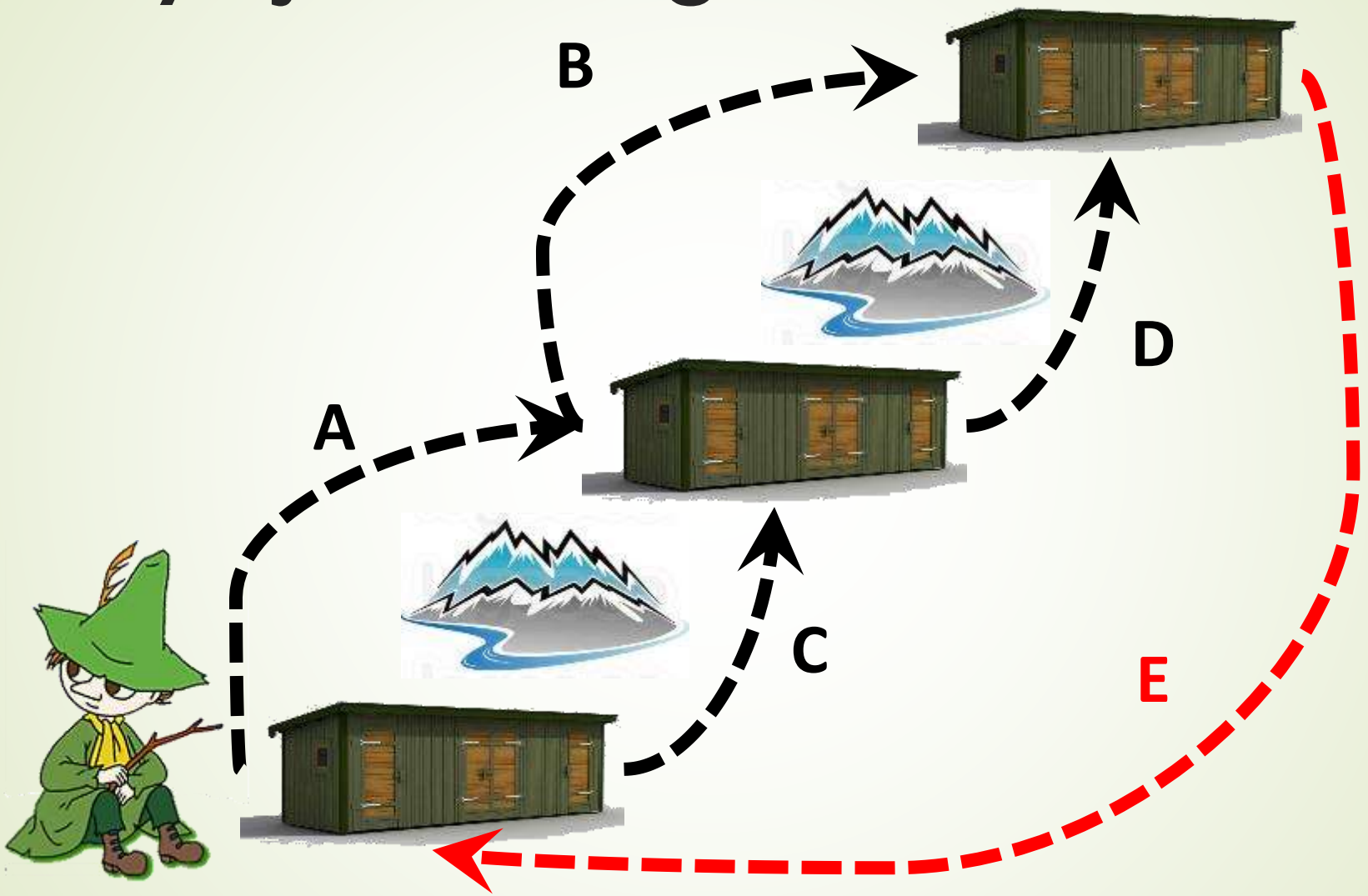
4.4.2 Według pokrycia decyzji 1(2)

```
if (a > b)
   printf (“a większe niż b”);
if (a == 5)
   printf (”a równa się 5”);
```

4.4.2 Według pokrycia decyzji 2(2)

```
if (a > b)
  printf (“a większe niż b”);
else
  printf (“a jest nie > niż b”);
if (a == 5)
  printf (”a równa się 5”);
```

Włóczykij – ile dróg?



4.4.3 Inne techniki strukturalne

- Na poprzednim slajdzie – **pokrycie % ścieżek** różnej długości
- Pokrycie **decyzji**, a pokrycie **rozgałęzień**
- Pokrycie tzw. **warunków decyzji** (*decision-condition*), oraz ich **kombinacji**

Na egzaminie ISTQB

- W rzeczywistości, NIGDY nie szacuje się ręcznie **liczby testów koniecznych**, aby uzyskać jakiś poziom pokrycia kodu
- W rzeczywistości, nie ma znaczenia znajomość **minimalnej liczby testów** potrzebnych, aby uzyskać jakiś poziom pokrycia kodu
- To są tylko sztuczki do egzaminu

4.5 Na podstawie doświadczenia

4.5 Na podstawie doświadczenia

- Doświadczenie, szerokie pojęcie… można je równie dobrze stosować wobec technik formalnych, które sprawdziły się poprzednio (wg doświadczenia)
- Oczywiście, doświadczenie może być zarówno czarno-, jak i białoskrzynkowe ☺
- Doświadczenie można utrwalać, np. w formie **list kontrolnych**

Co jest ważniejsze

- W praktyce, zdecydowanie najczęściej i najobszerniej stosowane jest właśnie projektowanie testów nieformalne, z doświadczenia
- Wg ISTQB natomiast, ma ono sens jako ***uzupełnienie*** technik „systematycznych" (formalnych – opisanych wcześniej), i powinno być stosowane po nich. **NIEPRAWDA**

Zgadywanie błędów 1(2)

- Oczywiście, „zgadywanie **błędów**” (kto, czemu i jak może się pomylić?) ma sens, ale ISTQB o nim nie wspomina ☹
- Nazwa „zgadywanie błędów” oznacza w sylabusie ***zgadywanie defektów***…
- … czyli tworzeniu (na podstawie doświadczenia) listy możliwych defektów, i…

Zgadywanie błędów 2(2)

- … i projektowanie testów, które tych defektów szukają
- Inna nazwa: „bug attacks”, zwane w sylabusie „**atakami usterkowymi**” ☺

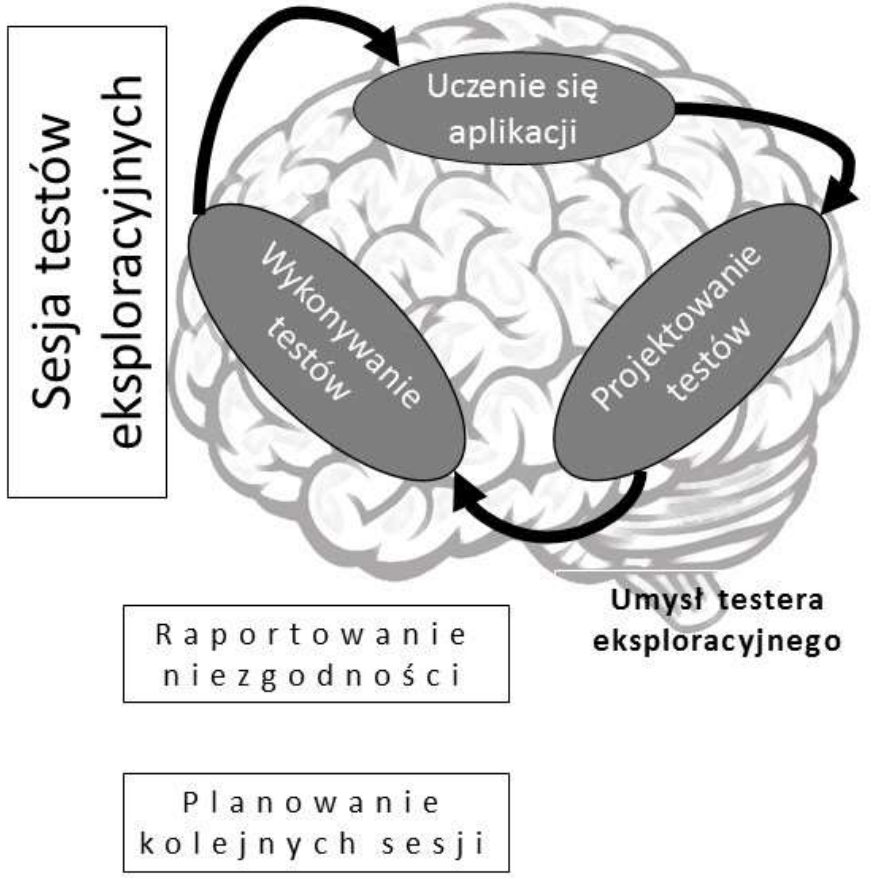
Testowanie eksploracyjne

- NIE jest przykładem testowania „niesystematycznego”, sorry
- Testowanie eksploracyjne, to podejście (szkoła, filozofia) do testowania, zakładająca:
  1. brak lub niedostateczność wymagań
  2. że marnowaniem czasu jest ich zapisywanie („testowanie skryptowe”)
  3. że znakomitą chwilą do projektowania testów jest wykonywanie poprzednich testów

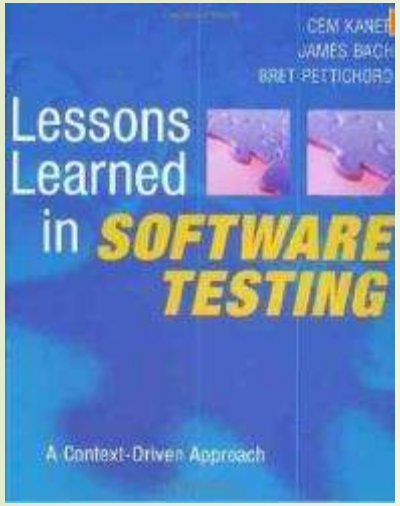
Proces eksploracji

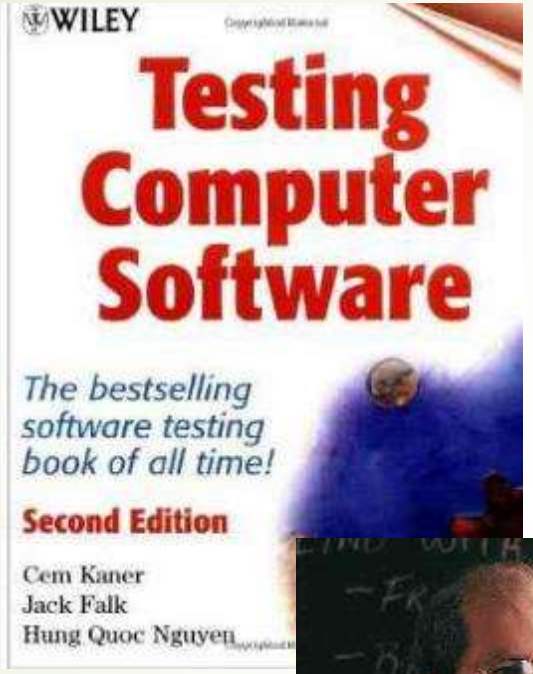
Podstawowy proces testowy wg ISTQB®

Stosowany eksploracyjny proces testowy



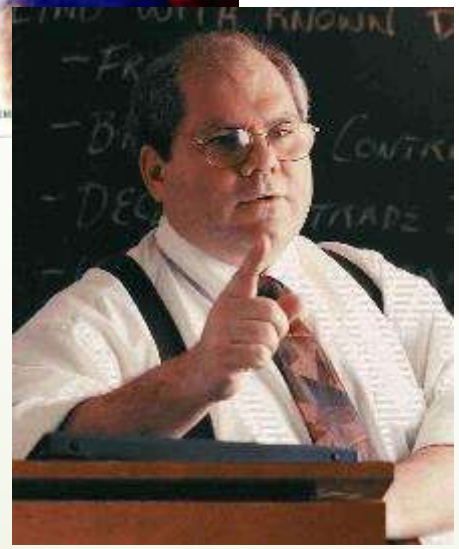
Mała historia eksploracji





SOFTWARE TESTING EDUCATION BBST

BLACK BOX SOFTWARE TESTING: INTRODUCTION TO TEST DESIGN: A SURVEY OF TEST TECHNIQUES



4.5 Testowanie na podstawie doświadczenia

pettichord.com

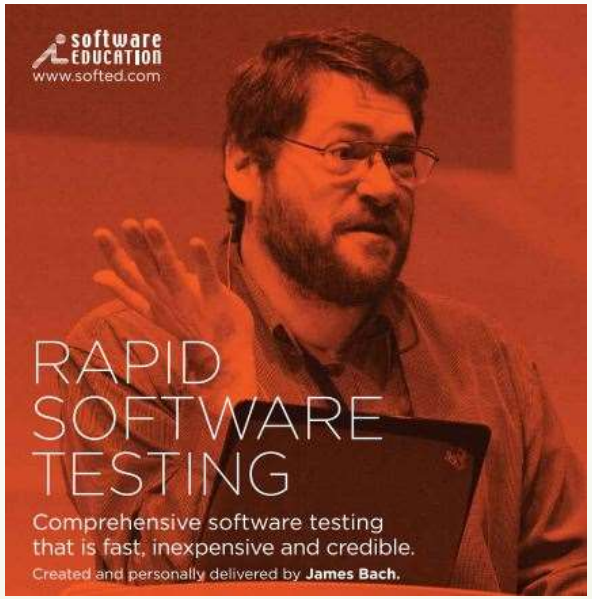
Szkoła analityczna
- matematyczno-logiczna
- dla potrzeb wysokiej niezawodności
- Beizer, Binder, Jorgenson, Musa

Szkoła fabryczno-standardowa
- według standardu, procesu
- tayloryzm i certyfikacja
- Rex Black, Dorothy Graham

Szkoła jakościowa
- testowanie jako motor jakości
- wskazania usprawnień procesu
- w dużych organizacjach
- Jarvis

Szkoła kontekstowa
- testowanie wykrywa potrzeby wg kontekstu
- testowanie eksploracyjne
- dla oprogramowania rynkowego
- Kaner, Marick, Bach, Bolton

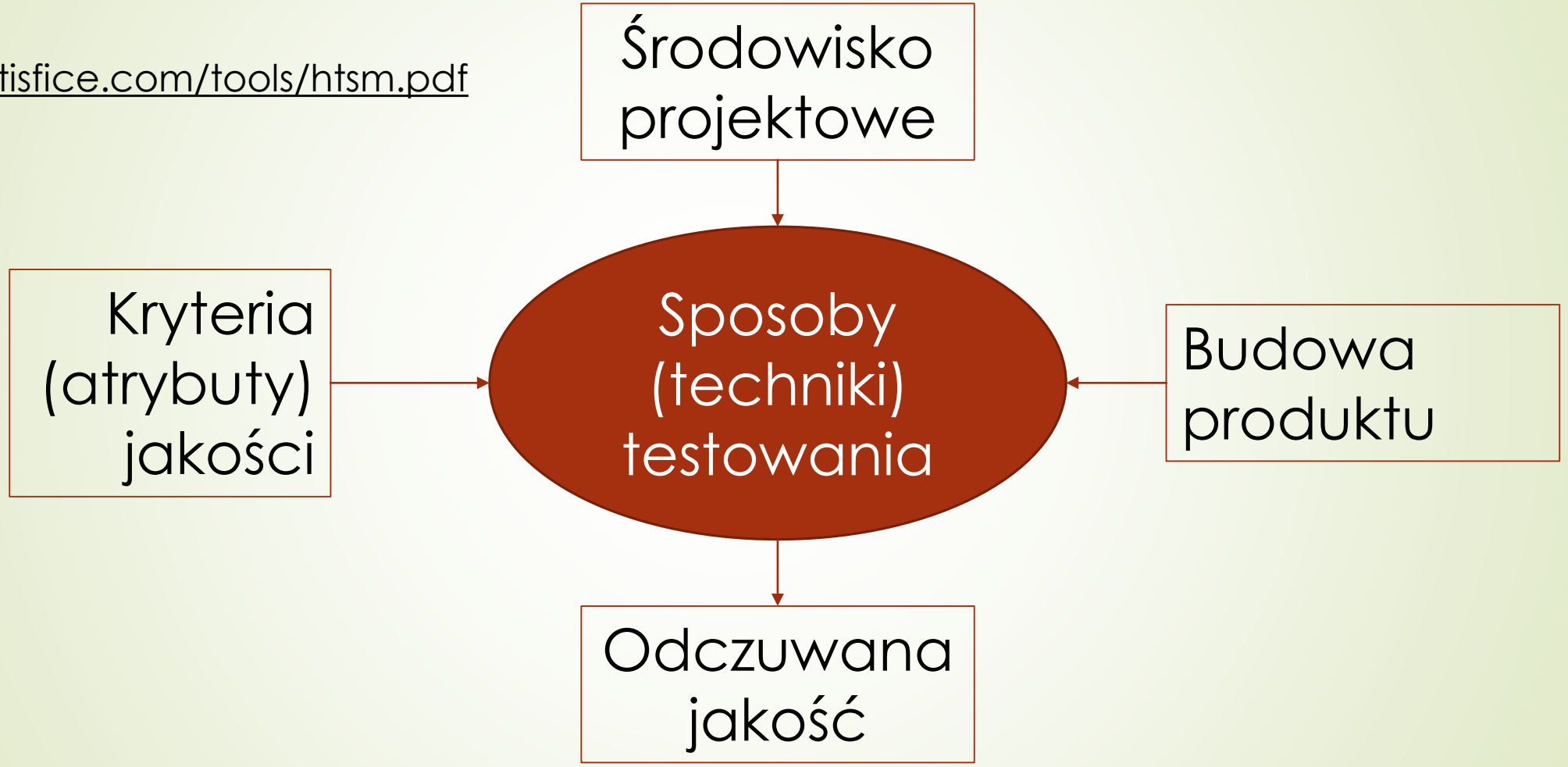
Guru szkoły kontekstowej ☺



4.5 Testowanie na podstawie doświadczenia

Heurystyczny model strategii testów ☺

satisfice.com/tools/htsm.pdf



Zarządzanie testowaniem w sesjach

- *Session-Based Test Management*
- satisfice.com/sbtm
- satisfice.com/articles/sbtm.pdf

keeping track of each tester's progress can be like herding snakes into a burlap bag

Ideologia eksploracji ☺

Podstępny zrzut bugów przez wroga

Tłusty, ospały zwolennik szkoły fabrycznej testowania niczego nie zauważył!

4.6 Wybór techniki

4.1 Proces rozwoju testów
4.2 Kategorie technik projektowania testów
4.3 Techniki czarnoskrzynkowe
4.4 Techniki białoskrzynkowe
4.5 Na podstawie doświadczenia
4.6 Wybór techniki

4.6 Wybór techniki testowania

- Zależy od wielu, wielu czynników… jakich?
- Brak ścisłych danych na temat względnej skuteczności różnych technik
- Czyli: ROI technik testowania, testowania w ogóle, i zapewnienia jakości w jeszcze większym ogóle, to sprawa niejasna…

5. Zarządzanie testowaniem

- Testowanie w inżynierii oprogramowania
- Wstęp do certyfikacji oraz ISTQB

1. Podstawy testowania
2. Testowanie w cyklu życia oprogramowania
3. Statyczne techniki testowania
4. Techniki projektowania testów
5. **Zarządzanie testowaniem**
6. Testowanie wspierane narzędziami

5. Zarządzanie testowaniem

5.1 Organizacja testów
5.2 Planowanie i szacowanie testów
5.3 Monitorowanie i nadzór
5.4 Zarządzanie konfiguracją
5.5 Ryzyko a testowanie
5.6 Zarządzanie incydentami

5.1 Organizacja testów

5.1.1 Organizacja a niezależność

- Już było w „psychologii" testowania
- No to jeszcze raz:
  - Każdy sam sobie testuje
  - Niezależni testerzy
  - Niezależny zespół testowy
  - Testerzy biznesowi (użytkownicy itp.)
  - Niezależni eksperci, wynajęci (outsourcing)
  - Zewnętrzni specjaliści

O niezależności testowania

- W dużym projektach, różne rozwiązania na różnych poziomach
- Zalety: inne spojrzenie, brak „uprzedzeń”, możność „weryfikacji założeń”
- Koszty / wady: „izolacja” (tzn. trzeba się pracochłonnie komunikować, tester staje się odkurzaczem… a więc i rzekomym wąskim gardłem

5.1.2 Zadania lidera testów i testera

- W praktyce, mamy dziesiątki rozwiązań (przykłady)
- OK, co robi „lider testów”? (wg ISTQB)
  - Koordynowanie strategii i planu „w górę”
  - Tworzenie (lub przegląd) strategii testów w projekcie
  - J. w. polityki testowej???
  - Inicjowanie i nadzorowanie rzeczywistego procesu testów w projekcie… (przykłady)

A co robi „tester”?

5.2 Planowanie i szacowanie testów

5.2.1 Planowanie testów

- Wykonuje się na 1000 sposobów
- Jeden plan („główny”) albo wiele
- ISTQB wyróżnia chętnie „plany dla poziomów”
- Powołuje się na IEEE 829, nie na ISO/IEC/IEEE 29119 ☹
- Trochę inaczej w agile…

5.2.2 Czynności planowania testów 1(2)

- Zakres (**co** i **na ile** jest testowane?)
- Ryzyko (konsekwencje i może prawdopodobieństwo)
- „Ogólne podejście" czyli lokalna strategia
  - W tym: co, jak, kim testować?
- Integracja z innymi działaniami w projekcie
- Harmonogram →

5.2.2 Czynności planowania testów 2(2)

- → zasoby do zadań (= ludzie do roboty)
- Metryki
- Dokumentacja

5.2.3 Kryteria wejścia

- Inaczej: kiedy można zacząć? (dany rodzaj, obszar lub poziom testów)
  - Środowisko testowe jest?
  - Inne narzędzia testowe?
  - Testy (zaprojektowane) i dane testowe?
  - Dostępny przedmiot testowania?
- Ważniejsze: czy przedmiot testów ma dostateczną jakość? (dlaczego)

5.2.4 Kryteria zakończenia

- Kiedy uznamy, że „jesteśmy gotowi”?
- Czyli to, co sprawdzamy w fazie „ocena spełnienia kryteriów zakończenia”
- Jakie zakładamy pokrycie, i czego?
- Jakie „estymaty niezawodności” (ile, jakie bugi wykryte, które naprawione?) i pozostałe ryzyko?
- Harmonogram – kiedy?

5.2.5 Szacowanie testów

- Cztery poziomy:
  1. Szacowanie pracochłonności przedsięwzięć
  2. Szacowanie pracochłonności projektów IT
  3. Szacowanie pracochłonności QA
  4. Szacowanie pracochłonności testów
- Szacowanie „ile i jak starannie trzeba testować” – to krok pierwszy (wg ryzyka)

Szacowanie pracy

- Po zdecydowaniu, **co** ma być zrobione, szacujemy **pracochłonność** (osobo-dni, czyli „mendejsy” ☺ - nie czas kalendarzowy, który zależy też od innych czynników)
- Metody:
  1. wg **modelu** (algorytmu)
  2. lub oszacowanie wg **doświadczenia** (metryki, albo opinie ekspertów)

Pracochłonność testów zależy od:

- Ile testów trzeba wykonać? (**ryzyko**, strategia ryzyka, inne działania QA)
- **Produkt** (wszelkie wymagania, w tym niezawodności)
- **Poziom ufności** oszacowań jakości
- **Jakości procesu** (w tym – umiejętności osób)
- I od… **wyników testów** ☹ →

Nieprzewidywalne 1(3)

- Jakość dostawy do testów
- Rada? Test dymny

Nieprzewidywalne 2(3)

- Termin dostawy do testów – chronicznie spóźniony

Nieprzewidywalne 3(3)

- Liczba bugów, które znajdzie ten poziom testów:
  - Trudności z wykonaniem testów
  - Konieczność oczekiwania, gdy awaria uniemożliwia dalsze testy
  - Czas na debugowanie, zwykle chytrze przypisywany testom
  - Więcej testów potwierdzających i regresji

5.2.6 Podejście i strategia testowa

- **Podejście do testów** *(test approach)* to ISTQB-owska nazwa na strategię testów w danym projekcie
- Są różne strategie / podejścia: pamiętacie?
- Ale ISTQB woli bardziej egzotyczną klasyfikację podejść

Strategie

1. Analityczne – np. na podstawie ryzyka
2. Modelowa (*model-based*), czyli… SUT ☹
3. Metodyczna (listy kontrolne, doświadczenie)
4. Zgodne ze standardem (np. agile ☺)
5. Dynamiczne/heurystyczne (np. eksploracyjne)
6. Konsultatywne (pytamy fachowców)
7. Regresywne (czytaj: anty-regresywne)

Strategie

5.2 Planowanie i szacowanie testów

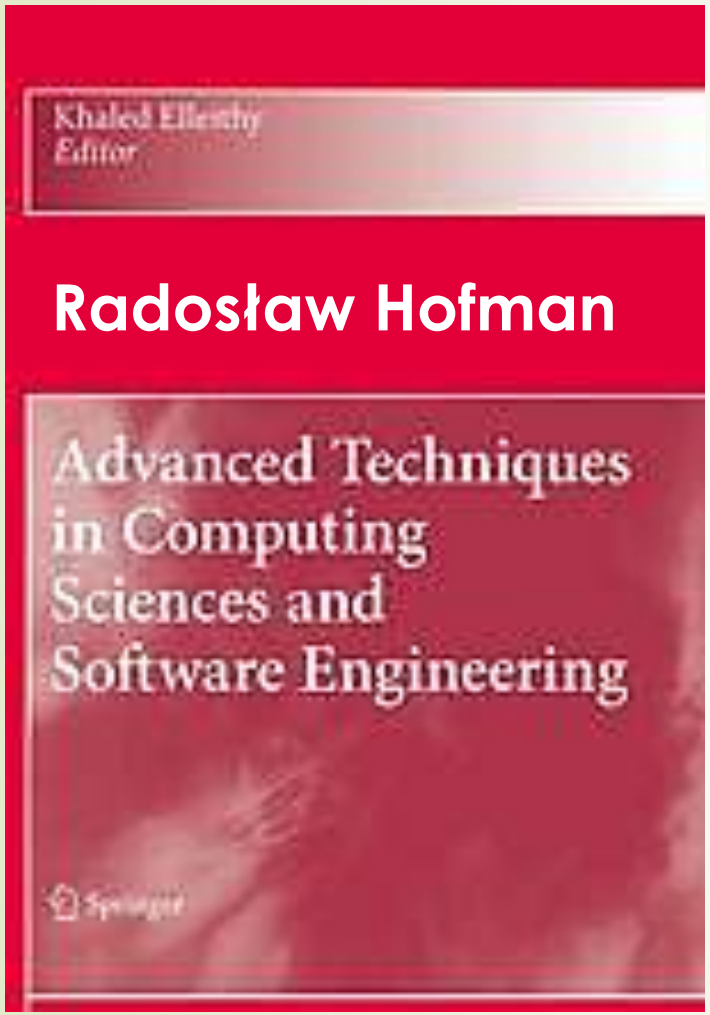
5.3 Monitorowanie i nadzór

5.1 Organizacja testów
5.2 Planowanie i szacowanie testów
5.3 Monitorowanie i nadzór
5.4 Zarządzanie konfiguracją
5.5 Ryzyko a testowanie
5.6 Zarządzanie incydentami

5.3.1 Monitorowanie postępu testów

- Co zrobiono? Gdzie jesteśmy? Co zostało do zrobienia? Kiedy będziemy gotowi?
- Stosowane metryki: **% pracy** wykonanej w danej fazie, dane o **bugach**, **pokrycie** testowe, **daty**, **koszty** i… **subiektywne zaufanie** →→ →→

Subiektywne zaufanie ☹



„Software Quality Perception”:

- Wpływ nacisku grupy
- Wpływ autorytetów
- Teoria perspektywy

SpringerLink

5.3.2 Raportowanie testów

- Raporty (i logi) bieżące
- Raport końcowy z testów

5.3.3 Kierowanie testami

- Hmm… angielskie *test* ***control*** to rządzenie, sterowanie, nadzór… ok, niech będzie „kierowanie”
- Na podstawie wyników monitorowania, decyduje się: zmiany priorytetów, zmiany harmonogramu, zmiany oszacowań, zmiany kryteriów wejścia (zwykle ↑) i wyjścia (zwykle ↓)

5.4 Zarządzanie konfiguracją

5.1 Organizacja testów
5.2 Planowanie i szacowanie testów
5.3 Monitorowanie i nadzór

5.4 Zarządzanie konfiguracją

5.5 Ryzyko a testowanie
5.6 Zarządzanie incydentami

5.4 Zarządzanie konfiguracją np.

Numer wersji

GZBRXXA/10.1.2/DT/AUAUA

Nazwa elementu konfiguracji

To znaczy, że w porównaniu z poprzednią wersją 10.1.1 dokonano tylko bardzo małą zmianę

Status, tutaj DT = do testowania

Nazwa wersji produktu, w skład której ta wersja elementu wchodzi

Zarządzanie konfiguracją w testach

- **To, co testujemy**, jest: zidentyfikowane, wersjonowane, ze statusem, powiązane
- **Artefakty testowe** („testalia”) są poddane zarządzaniu konfiguracją
- Po co?

Po co zarządzanie konfiguracją?

5.4 Zarządzanie konfiguracją

5.5 Ryzyko a testowanie

5.5.1 Obszary ryzyka projektowego

- A co to ma wspólnego z testowaniem?

NIC

- Tj. ryzyko projektowe może też dotknąć testowanie, ale testy mierzą je tylko za pośrednictwem ryzyka produktowego

Ryzyka projektowe – długa lista

- Czynniki organizacyjne
  - Brak umiejętności, brak ludzi, problemy polityczne, choroba współuzależnienia
- Problemy techniczne
  - Brak dobrych wymagań (???), nierealny budżet, brak dobrych innych rzeczy (programu, narzędzi, środowisk)
- Problemy z dostawcami

5.5.2 Obszary ryzyka produktowego

- To proste: **NIE DZIAŁA POPRAWNIE**
- Ale można skomplikować!
  - Awarie
  - Szkody, wynikające z awarii
  - Niedostateczne atrybuty (także poza-funkcjonalne)
  - Kiepskie dane
  - Oprogramowanie, działające niepoprawnie

Test a ryzyko (naprawdę)

- Pomaga **identyfikować** ryzyko (oj, nie działa!)
- **Mierzy** ryzyko (działa, czy nie działa?)
- Pomaga **zmniejszyć** ryzyko (nie działa, więc naprawiamy)
- Szacowane ryzyko produktowe (konsekwencje i prawdopodobieństwo awarii) pozwala określić **potrzebną intensywność testów**

Kilka marzeń ściętej głowy

- Ryzyko określa właściwe techniki testowania
- Ryzyko określa zakres testów (no… troszkę)
- Ryzyko pozwala określić właściwą kolejność wykonywanie testów (ważne najpie[illegible]) - trochę
- Ryzyko wskazuje, że warto unikać choroby współuzależnienia (?)

Proces zarządzania ryzykiem

- Identyfikacja (tak, test pomaga!)
- Ocena
  - Prawdopodobieństwa (test pomaga)
  - Konsekwencji (zadanie dla biznesu lub konstruktorów, nie testu)
- Wymyślanie i ocena opłacalności:
  - Czynności zapobiegawczych
  - Czynności zaradczych

5.6 Zarządzanie incydentami

5.1 Organizacja testów
5.2 Planowanie i szacowanie testów
5.3 Monitorowanie i nadzór
5.4 Zarządzanie konfiguracją
5.5 Ryzyko a testowanie
5.6 Zarządzanie incydentami

5.6 Zarządzanie incydentami

Co to jest „incydent”

- To **rozbieżność** między wynikiem oczekiwanym, a rzeczywistym
- Incydent może – ale nie musi – być objawem **defektu** (to trzeba zbadać)
- Jeśli jest, warto rozważyć **naprawę**
- Do tego wszystkiego warto mieć **określony proces**

Kiedy i skąd są incydenty

- Z użytkowania
- Z testowania
- Z przeglądów
- Z różnych czynności konstruowania oraz integracji

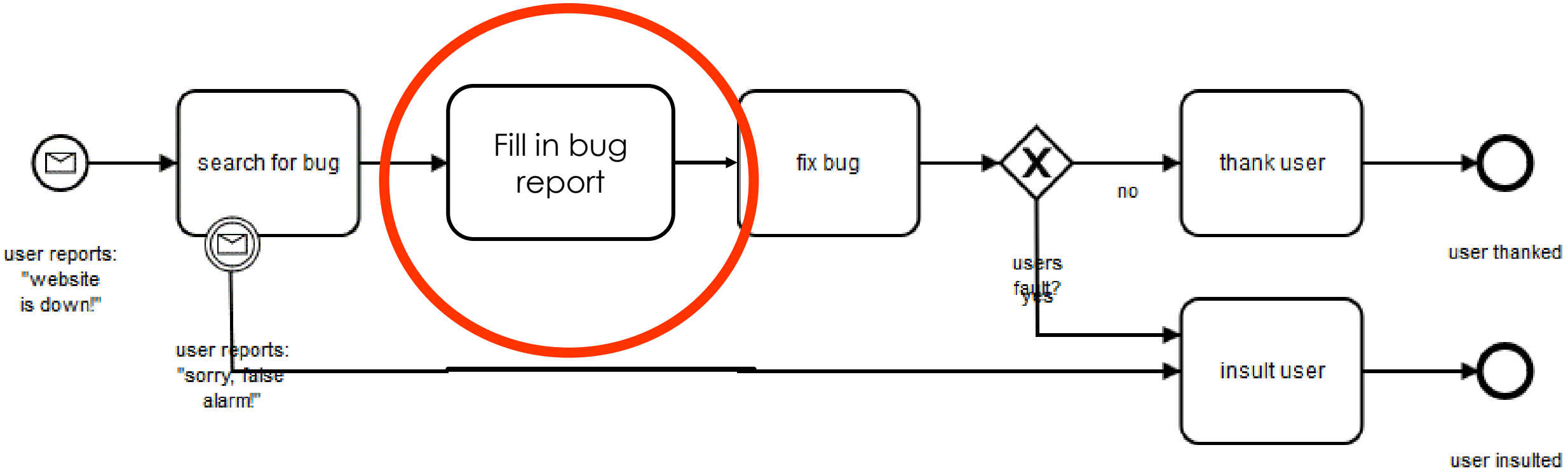
Po co zgłoszenia (raporty) incydentów?

- Jako forma przekazania informacji o incydencie (o jego zaistnieniu, przebiegu analizy, przebiegu naprawy i testów potwierdzających)
- Aby móc śledzić status produktu i projektu
- Aby móc mierzyć proces produkcji oprogramowania i planować jego ulepszanie

Co zawiera raport incydentu?

- Kto, kiedy, zrobił co? (wykrył, zanalizował, zadecydował, naprawił, przetestował)
- Jak wyglądał incydent? (wynik rzeczywisty oraz wskaźnik do oczekiwanego)
- Szczegóły wydarzenia (logi, zrzuty)
- Waga, priorytet itp.
- Status (wynika z procesu incydentów)

Proces zarządzanie incydentami ☺



6. Testowanie wspierane narzędziami

- Testowanie w inżynierii oprogramowania
- Wstęp do certyfikacji oraz ISTQB

1. Podstawy testowania
2. Testowanie w cyklu życia oprogramowania
3. Statyczne techniki testowania
4. Techniki projektowania testów
5. Zarządzanie testowaniem
6. **Testowanie wspierane narzędziami**

6. Testowanie wspierane narzędziami

6.1 Typy narzędzi
6.2 Korzyści i ryzyko narzędzi
6.3 Wdrażanie narzędzi

6.1 Typy narzędzi

6.1.1 Cele narzędzi dla testerów

- Najszersza klasyfikacja:
  1. Używane **wprost w testach** (do ich wykonywania, projektowania, tworzenia danych. Porównywania itd.)
  2. Wspomagające **zarządzanie testami**
  3. **Śledzące**, monitorujące, rejestrujące
  4. **Dowolne zastosowane w testowaniu** – np. edytor tekstu albo e-mail ☺

Po co narzędzia?

- Szybciej, dokładniej, bardziej niezmiennie
- Konieczna automatyzacja, gdy ręcznie coś jest niewykonalne lub fantastycznie kosztowne
- Większa niezawodność testów
- Spoza sylabusa: uwolnienie i podniesienie statusu testerów i testowania

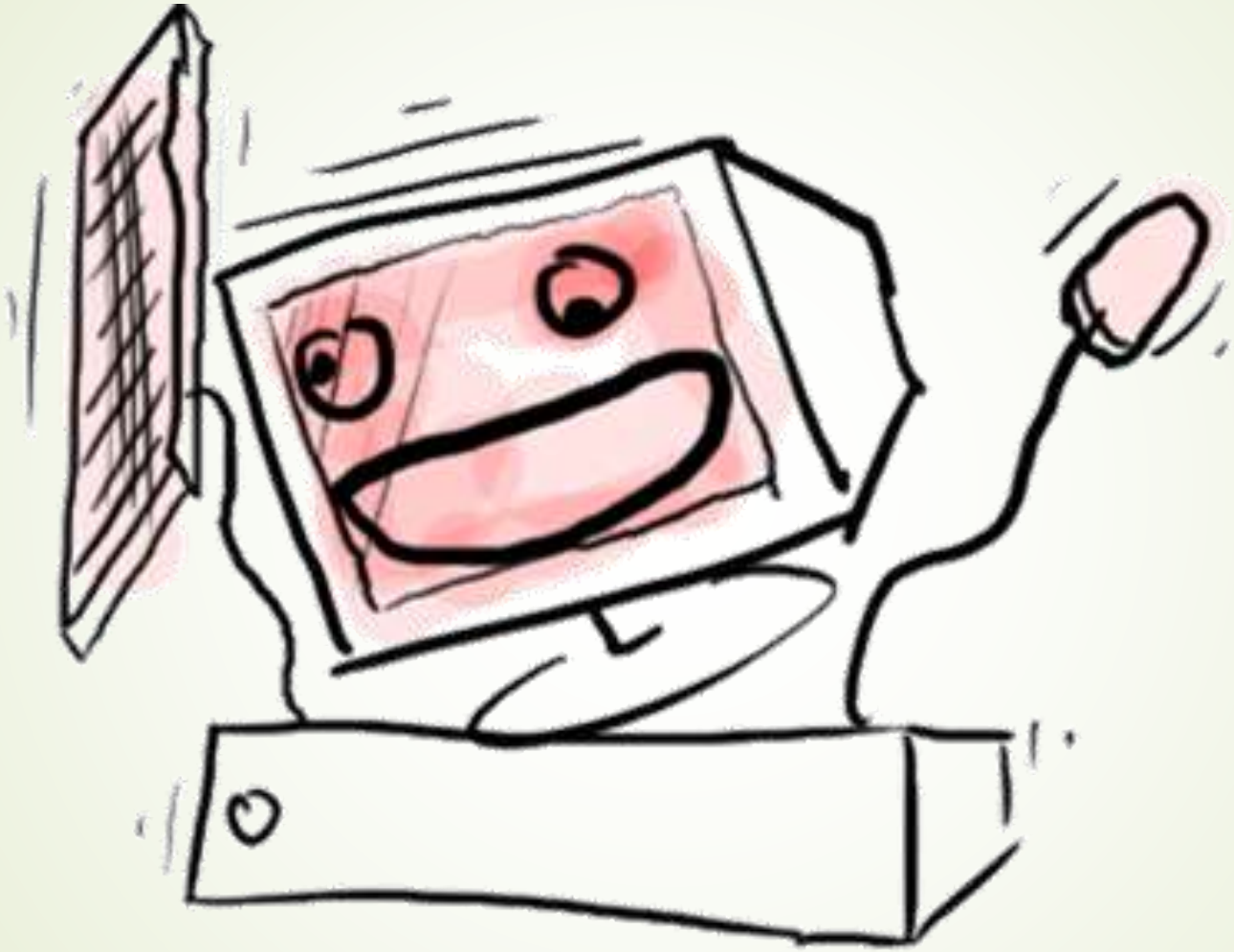
Tajemnicze pojęcie

- „Struktura testowa” ☹
- … czyli po polsku „*framework*” ☺
- To jeszcze głupsza nazwa, niż „jarzmo” ☹:
  - Jarzmo testowe?
  - Biblioteki funkcji testowych?
  - Metody automatyzacji testów?
  - Albo… proces testowy w ogóle?

6.1.2 Klasyfikacja narzędzi testowych

- 100 możliwych rodzajów klasyfikacji
- Sylabus – słusznie! ☺ - klasyfikuje wg tego, jaką **czynność testową** dane narzędzie wykonuje
- Oczywiście, jest wiele hybrydowych ☺, powiązanych w pakiety, itd.

Inwazyjność – efekt próbnika narzędzi



6.1.3 Narzędzia do zarządzania testami

6.1 Typy narzędzi testowych

https://secure.reqtest.com/ASPX/StartPage/Start.aspx?justActivated=true

Start page | Create new requirement | Create new test case | Create new bug report | My bug reports | Sent bug reports | Contact list | Help | Log out

- Agile board
- Requirements
- Plan tests
- Execute tests
- Bug reports
- Traceability
- Settings

Your evaluation license will expire in 11 days. Click here to upgrade your license now! Ri

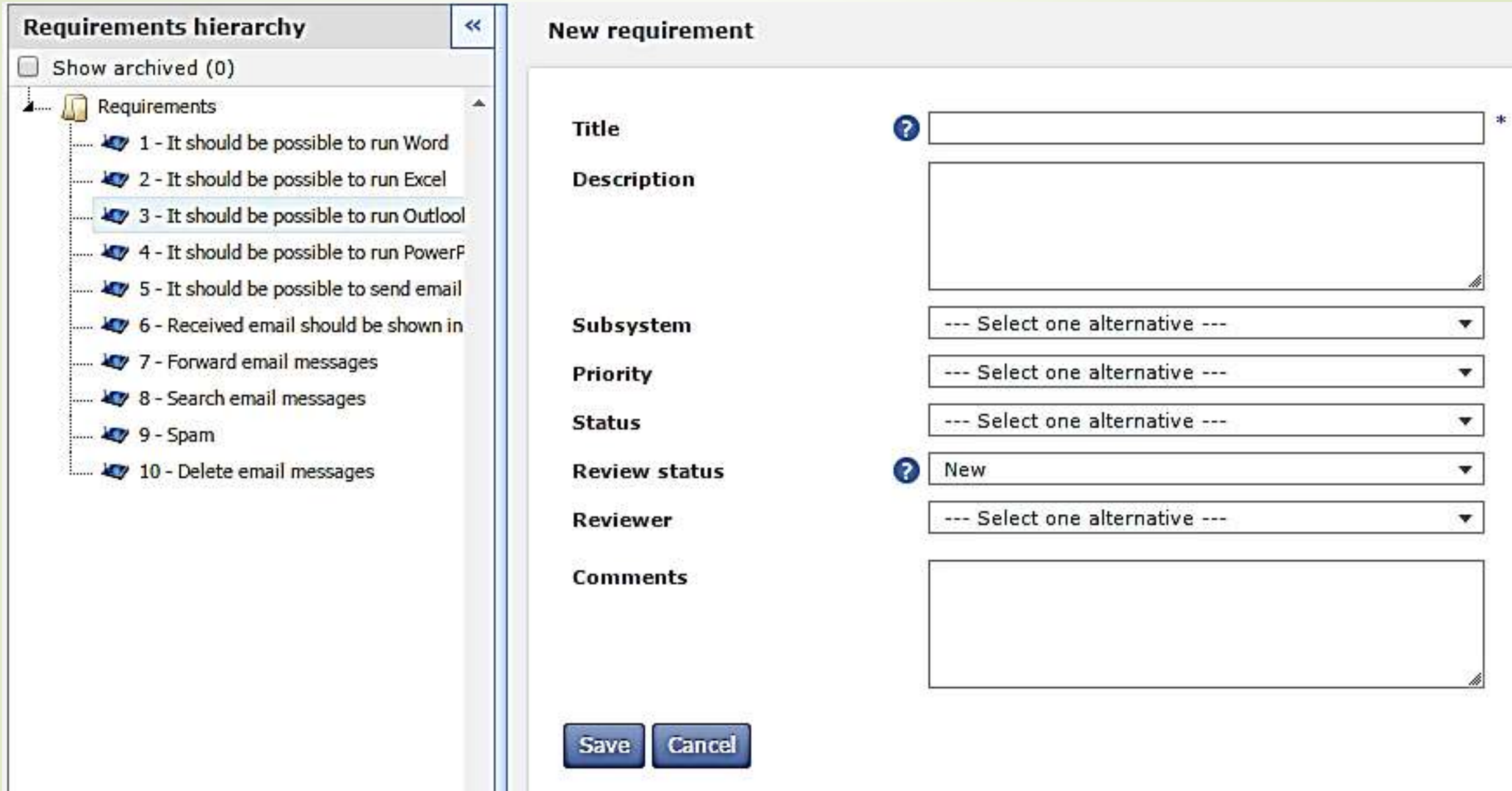
Welcome Bogdan

Get more from your ReQtest trial

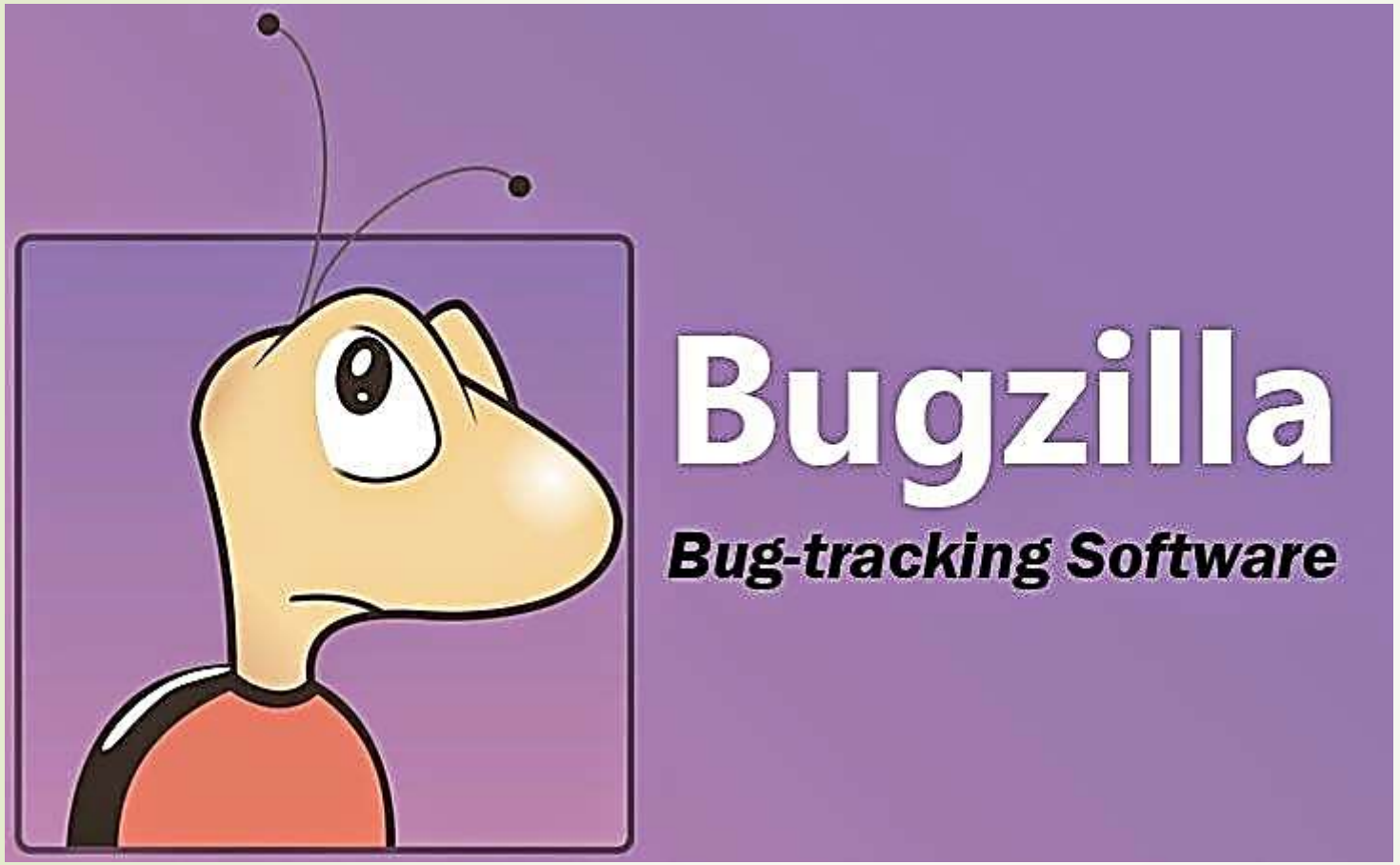
Here are a few tips to ge

- Watch an introductory
- In many places in ReQ that show you how to u
- In our help files you ca ReQtest. The menu at

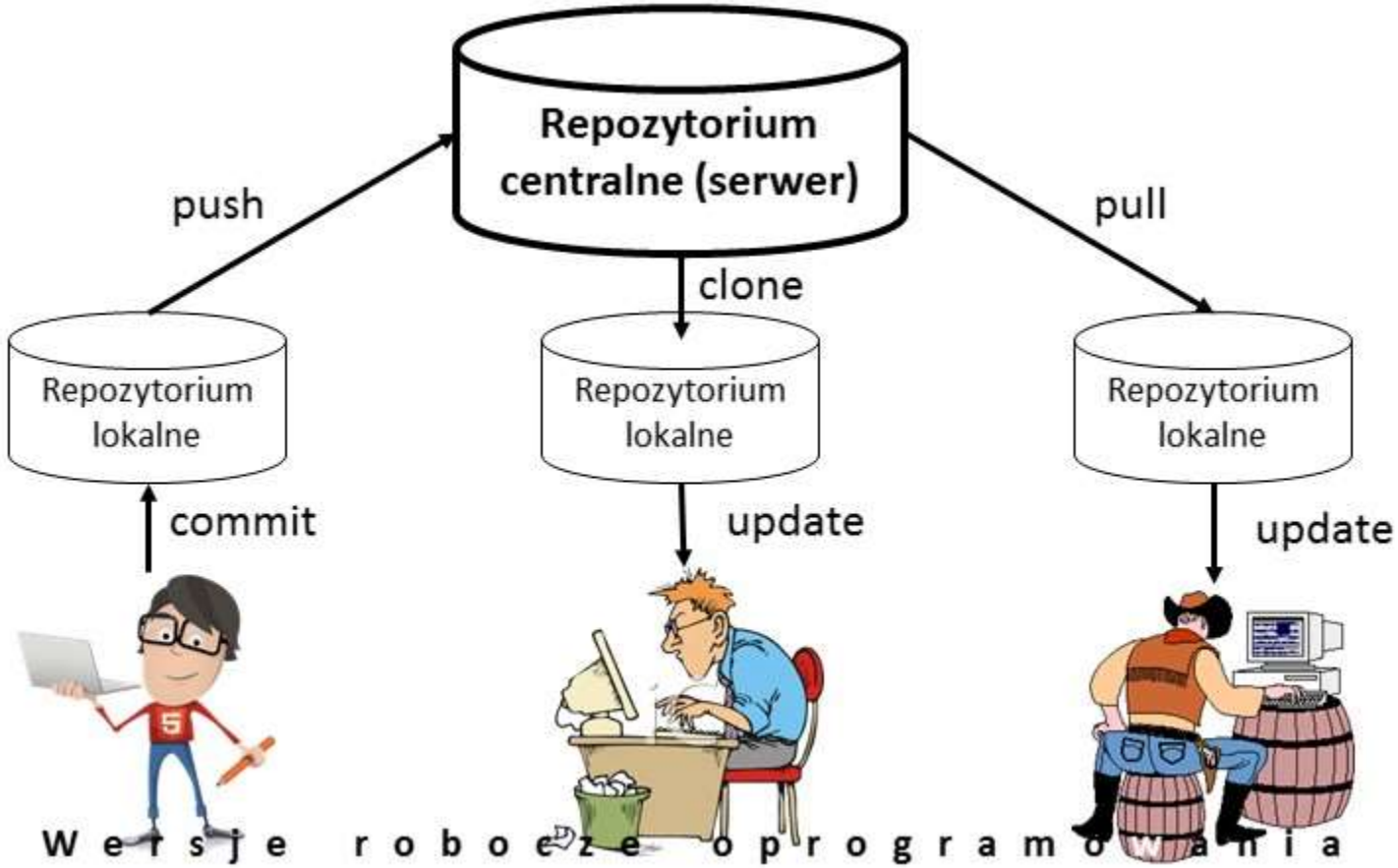
Do wymagań


Requirements hierarchy
Show archived (0)
Requirements
1 - It should be possible to run Word
2 - It should be possible to run Excel
3 - It should be possible to run Outlool
4 - It should be possible to run PowerP
5 - It should be possible to send email
6 - Received email should be shown in
7 - Forward email messages
8 - Search email messages
9 - Spam
10 - Delete email messages
New requirement
Title
Description
Subsystem
--- Select one alternative ---
Priority
--- Select one alternative ---
Status
--- Select one alternative ---
Review status
New
Reviewer
--- Select one alternative ---
Comments
Save
Cancel

Do incydentów

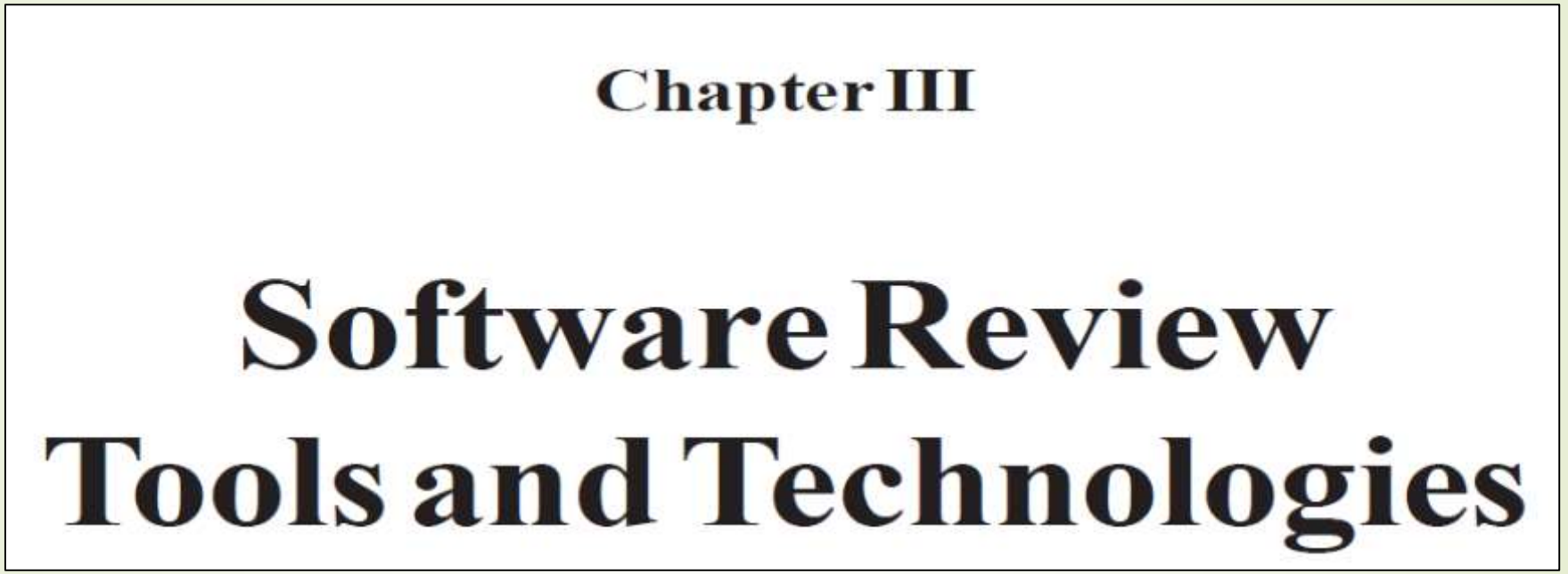


Do zarządzanie konfiguracją



6.1.4 Narzędzia do testów statycznych

- Wspierające przeglądy



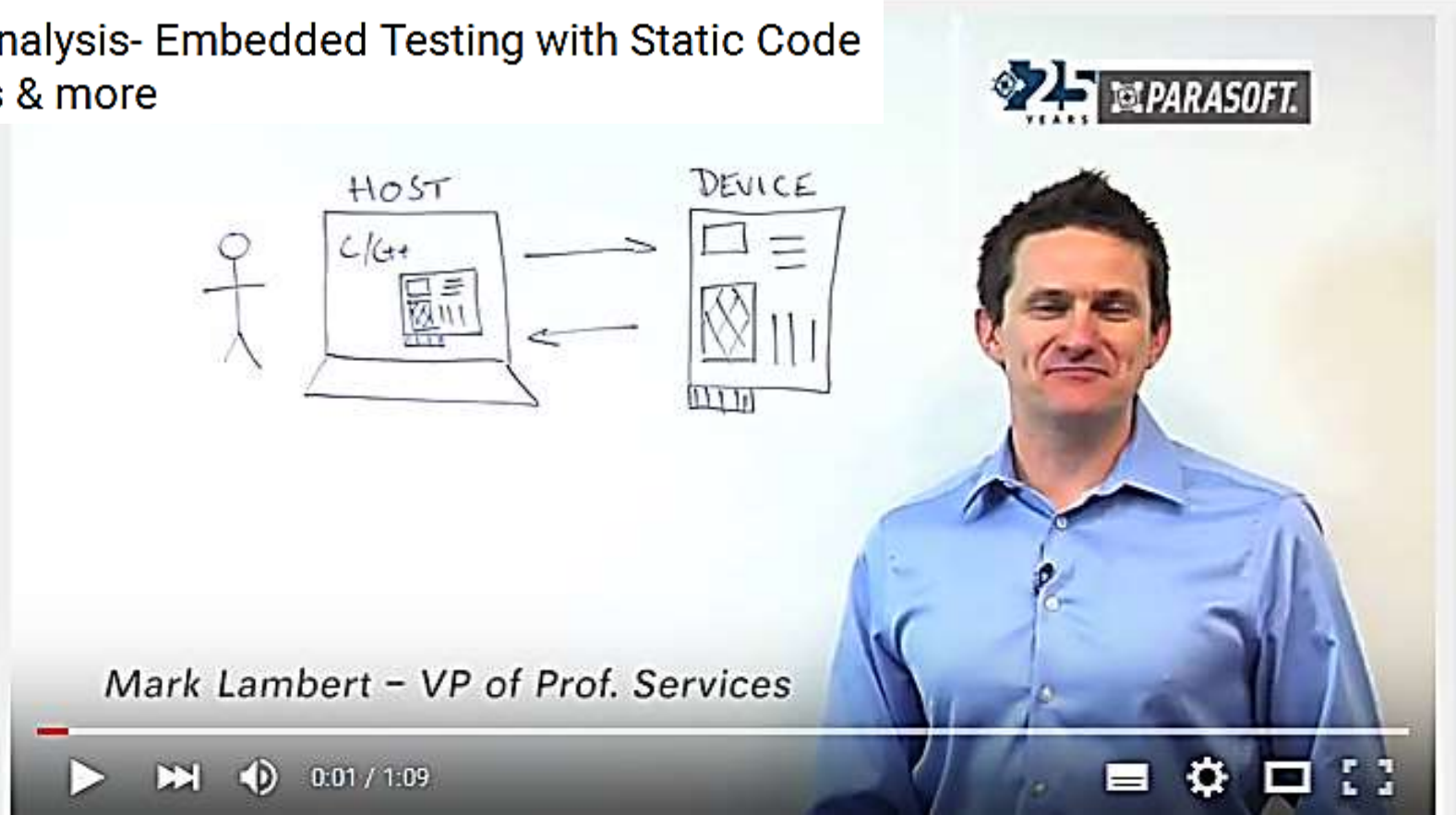
6.1.4 Narzędzia do testów statycznych

- Analiza statyczna

D

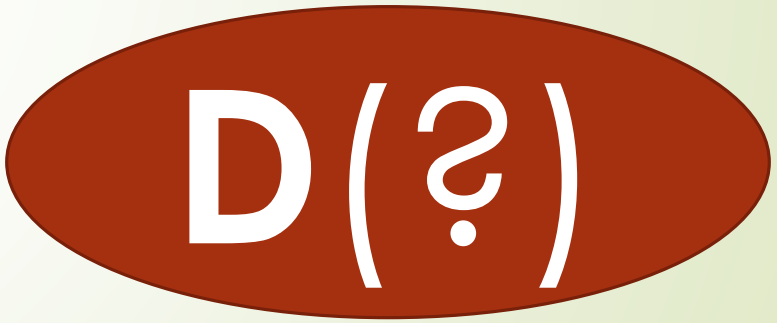
YouTube PL

Static Analysis- Embedded Testing with Static Code Analysis & more



Narzędzia do modelowania

- Modelowania czegoś… na przykład projektu oprogramowania, albo wymagań
- Mogą często tworzyć testy z modelu
- **pl.wikipedia.org/wiki/Lista_narz%C4%99dzi_UML**



6.1.5 Narzędzia do specyfikacji testów

- **Do projektowania testów**
  - Z wymagań (patrz: do modelowania)
  - Z GUI
  - Z modeli architektury (patrz: do modelowania)
  - Z kodu

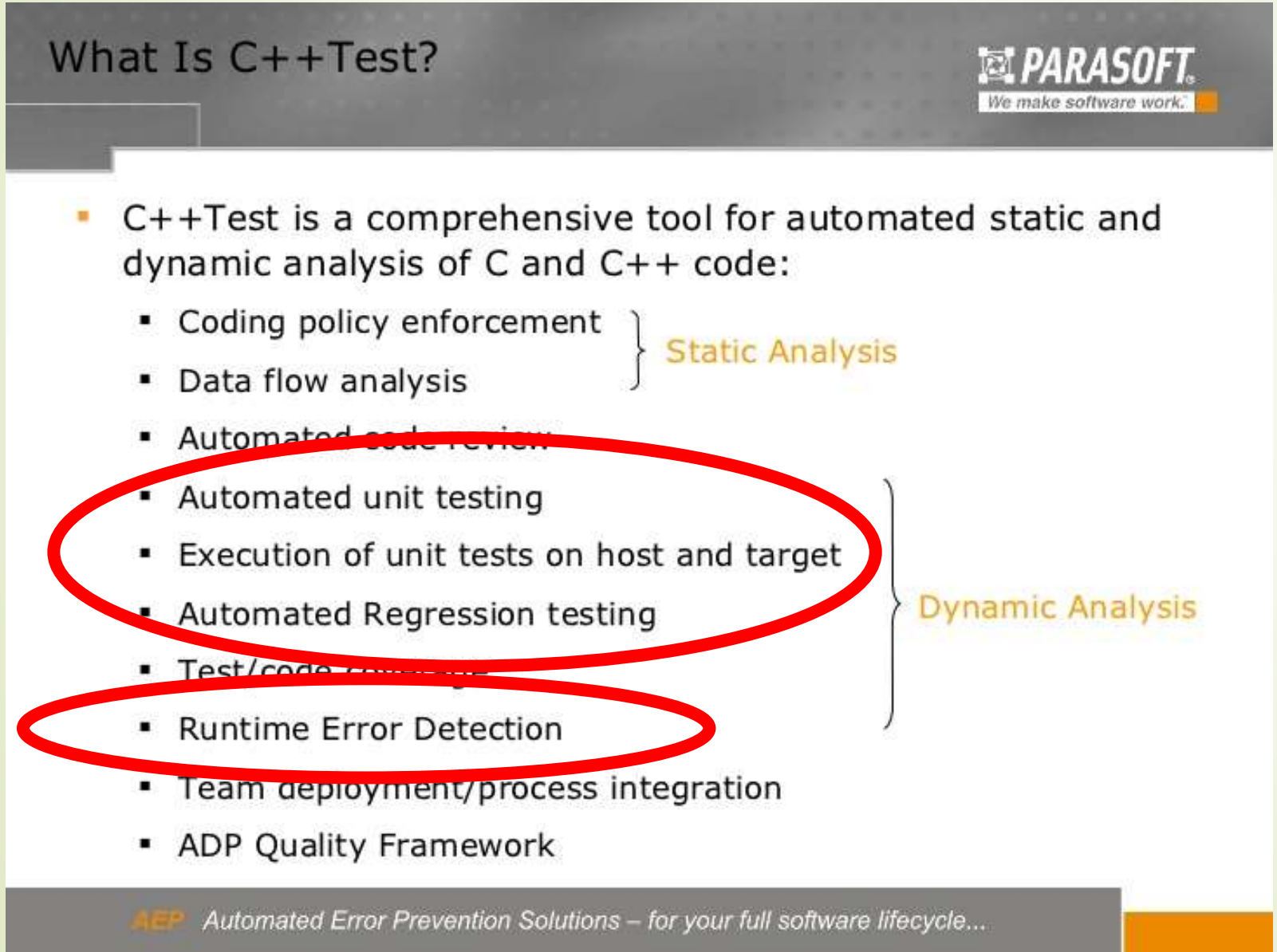
6.1.5 Narzędzia do specyfikacji testów

- Do tworzenia danych testowych – przetwarzanie danych np. z baz, na dane testowe (w tym np. anonimizacja)

6.1.6 Narzędzia do wykonywania testów

- Wykonują, logują…

Jarzmo do testów jednostkowych



Komparatory i pomiary pokrycia kodu

- Komparator porównuje wyniki rzeczywiste z oczekiwanymi – zwykle jest elementem robota testowego

Coverage Report - wire.jspbeans

Package	# Classes	Line Coverage	Branch Coverage	Complexity
wire.jspbeans	46	73%	86%	4.166

Classes in this Package	Line Coverage	Branch Coverage	Complexity
MacroSwitch	97%	78%	1.15
data_Bean	74%	98%	22.4
data_BeanHelper	94%	99%	4.162
data_BeanRetrieve	86%	72%	9
data_ImportBean	88%	100%	1.75

Do testów zabezpieczeń

- Robot specjalistyczny do testów zabezpieczeń

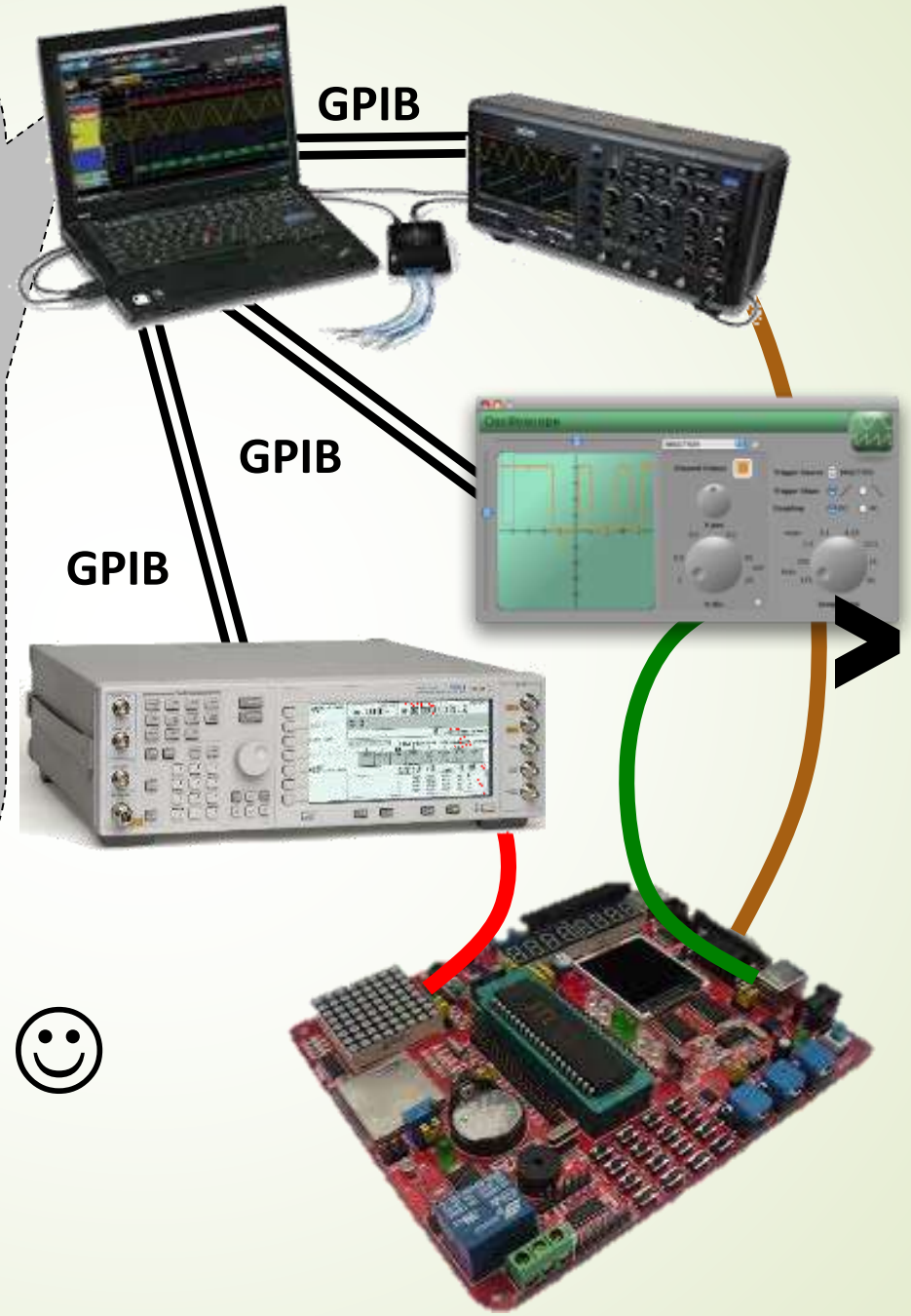
6.1.7 Narzędzia do testów wydajności

- Do analizy dynamicznej
  - Zależności czasowe
  - Zakleszczenia procesów
  - Wycieki pamięci
  - Obciążenie zasobów
  - Itp.

Sterowniki GPIB dla wszystkich urządzeń

Program nadzorujący wykonywanie testów

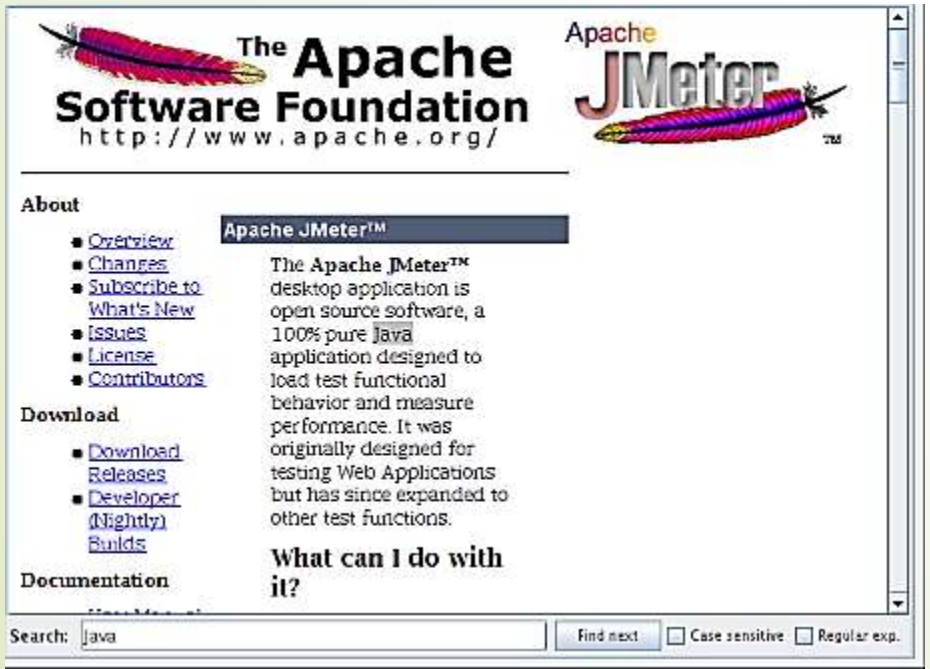
Program do rezerwacji i nadzoru dostępu



Np. o tak ☺

Obciążanie i pomiary

- Narzędzia do testów wydajnościowych/narzędzia do testów obciążeniowych/narzędzia do testów przeciążeniowych



6.1.8 Różne narzędzia do testów

- Ocena jakości danych
- Testy użyteczności

6.2 Korzyści i ryzyko narzędzi

6.1 Typy narzędzi

6.2 Korzyści i ryzyko narzędzi

6.3 Wdrażanie narzędzi

Korzyści

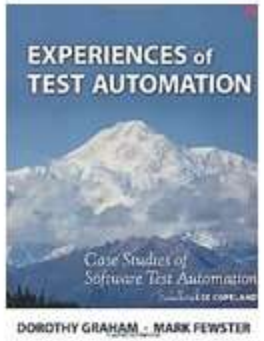
- Redukcja powtarzalnej pracy
- Wzrost spójności i powtarzalności testów
- Obiektywność oceny
- Łatwiejszy dostęp

Ryzyka i koszty

- Nierealistyczne oczekiwania korzyści
- Niedoszacowanie kosztów
- Nieświadomość utrzymania
- Zarzucenie testów ręcznych
- Słaba integracja narzędzi
- Słaby dostawca i wsparcie
- Licencjonowanie

Ryzyka i koszty dla robotów

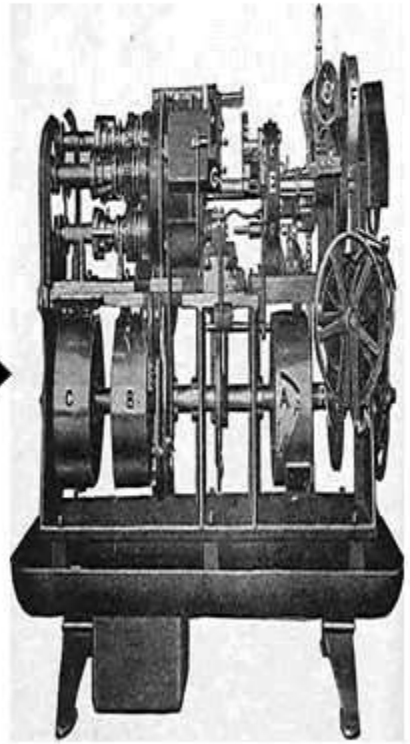
- Zarejestruj-odtwórz
- Sterowane danymi
- Wg słów-kluczy



Dorothy Graham,
Mark Fewster
“Keyword - driven testing”

Hans Buwalda
“Test-frame method

Skrypt w języku do opisu testów, tworzonym wedle potrzeb projektu, nie wymagającym umiejętności programowania



Program tłumaczący skrypt w języku testowym na skrypt robota do wykonywania testów

Skrypt dla robota do wykonywania testów

Automatyczne wykonanie testów

Ryzyka i koszty dla analizy statycznej

- Konieczność konfiguracji
- Początkowo, nadmiar komunikatów
- Niepoprawna konfiguracja

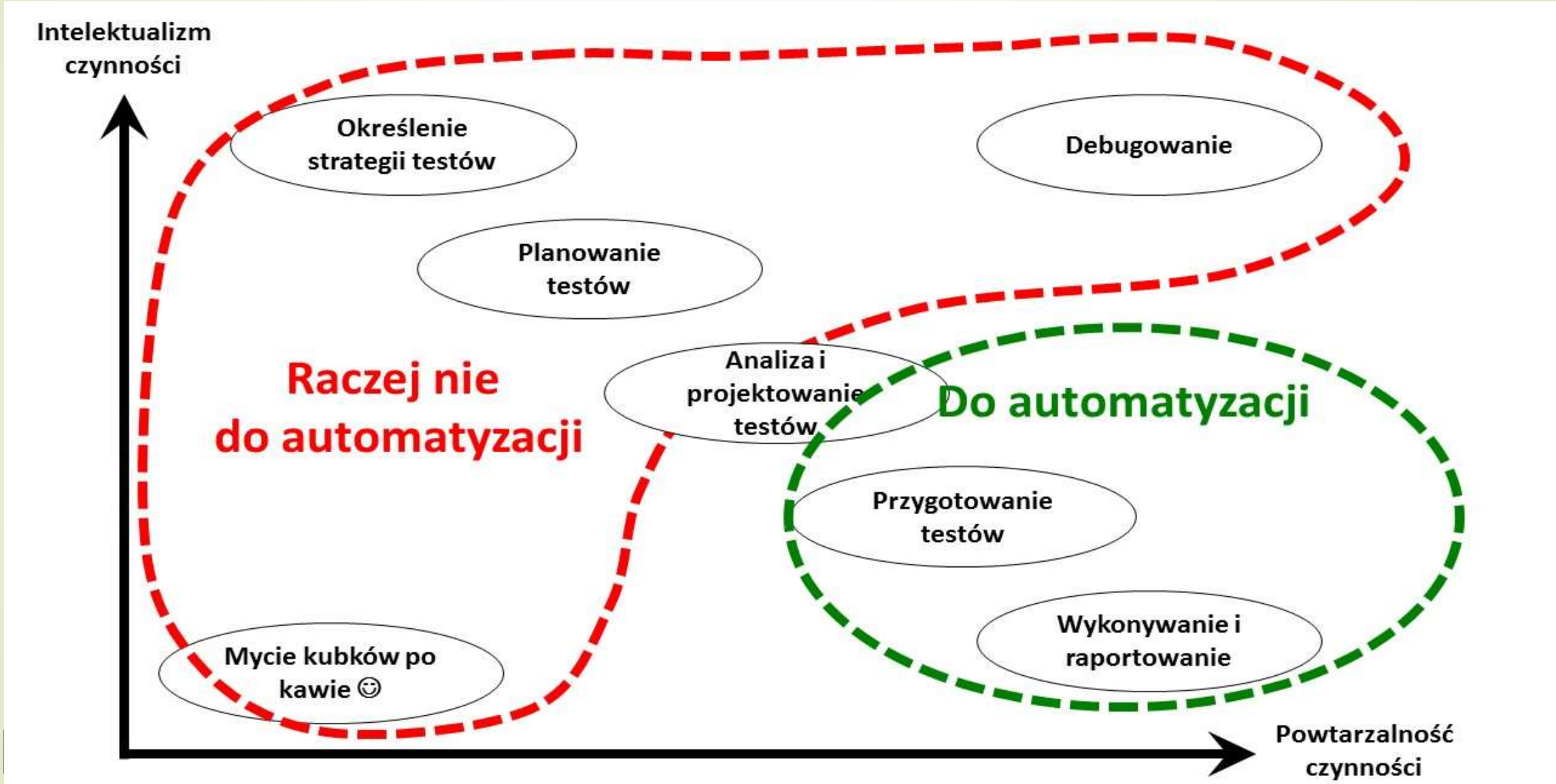
Ryzyka i koszty narzędzi do zarządzania

6.3 Wdrażanie narzędzi

6.1 Typy narzędzi
6.2 Korzyści i ryzyko narzędzi

6.3 Wdrażanie narzędzi

Co automatyzować?



O czym myśleć przed wdrożeniem?

- Jakość procesu
- Po co automatyzacja?
- Sprawdzenie skuteczności (*proof-of-concept*) i projekt pilotażowy
- Ocena dostawcy
- Ocena szkoleń i wsparcia
- Oszacowanie zwrotu z inwestycji

Cele projektu pilotażowego

- Poznanie narzędzia
- Ocenienie, na ile pasuje
- Ustalenie standardów użycia, przekazywania, archiwizacji artefaktów, wewnętrznego wsparcia
- Ponowna ocena zwrotu z inwestycji

Warunki powodzenia

- Stopniowe wdrażanie
- Dostosowanie własnych procesów
- Organizacja wsparcia i ulepszania własnych procedur

Podstawy testowania oprogramowania KONIEC części 2

- 3 dni
- Zgodne z sylabusem ISTQB
- Autor: Bogdan Bereza
- bogdan.bereza@victo.eu