RADOSŁAW SMILGIN

DANE TESTOWE

TEORIA I PRAKTYKA

- Podstawowe typy danych i związane z nimi błędy
- Techniki projektowania testów
- Definiowanie danych testowych
- Zarządzanie danymi i rozwiązywanie problemów
- Praktyczne przykłady danych testowych i ich wykorzystanie
- Sposoby pozyskiwania rzeczywistych danych
- Samodzielne generowanie danych testowych

Książka, której nie może zabraknąć w biblioteczce ambitnego programisty!

Helion

Dane testowe. Teoria i Praktyka

Autor: Radosław Smilgin, Anna Piaskowy
ISBN: 978-83-246-2520-8
Format: 158×235, stron: 122

Książka, której nie może zabraknąć w biblioteczce ambitnego programisty!

Programowanie nierzadko uchodzi za sztukę magiczną, jednak nawet najbardziej pomysłowa aplikacja okaże się bezużyteczna, gdy znajdzie się w niej choć jeden poważny błąd – powodujący, że działanie programu będzie niezgodne z oczekiwaniami twórców i użytkowników. Dlatego nie mniej istotną kwestią jest należyte sprawdzenie poprawności oprogramowania. Takie działanie pozwoli nam zyskać pewność, że otrzymane za jego pomocą wyniki będą w pełni pokrywały się z przyjętymi założeniami. Zadanie to nie jest wcale tak banalne, jak mogłoby się wydawać, a zlekceważenie etapu testów może kosztować znacznie więcej, niż gotowi jesteśmy zapłacić.

Niestety, tematyce tej nie poświęca się zwykle odpowiednio dużo uwagi, co można łatwo stwierdzić, przeglądając dostępne na rynku opracowania dotyczące testowania aplikacji. Chlubnym wyjątkiem jest tu książka „Dane testowe. Teoria i praktyka", w całości poświęcona metodologii przygotowywania i praktycznego wykorzystywania danych testowych, które zapewniają maksymalną niezawodność oraz bezpieczeństwo działania programów. Autor wprowadza Czytelnika w teoretyczne podstawy definiowania i generowania tego rodzaju danych, lecz prezentuje również przykłady i możliwości zastosowania opisywanych technik w praktyce. Dzięki temu każdy programista i tester będzie mógł skrócić czas sprawdzania poprawności działania aplikacji i uniknie szeregu typowych błędów oraz zaniechań, popełnianych zwykle w tym nierzadko bardzo skomplikowanym procesie.

- Podstawowe typy danych i związane z nimi błędy
- Techniki projektowania testów
- Definiowanie danych testowych
- Zarządzanie danymi i rozwiązywanie problemów
- Praktyczne przykłady danych testowych i ich wykorzystanie
- Sposoby pozyskiwania rzeczywistych danych
- Samodzielne generowanie danych testowych

Helion SA
ul. Kościuszki 1c
44-100 Gliwice
tel. 32 230 98 63
e-mail: helion@helion.pl

Spis treści

Rozdział 4.

Dane generowane a dane rzeczywiste

4.1. Dane rzeczywiste

Najskuteczniejsze jako pozytywne dane testowe okazują się dane rzeczywiste. Nic tak nie sprawdzi systemu jak prawdziwe przypadki rodem z produkcji. I tak pierwsza rzecz, jaką wykonujemy podczas testowania, to sprawdzenie nowego systemu pod kątem jego współpracy z poprzednio zgromadzonymi danymi. To z kolei ma szczególne znaczenie dla testów przygotowujących system do migracji danych ze starego systemu — jeżeli chcemy mieć pewność, że dane skopiowane z poprzedniej wersji systemu będą poprawnie działały w jego nowej odsłonie.

4.1.1. Pozyskiwanie danych rzeczywistych

Dane pozyskuje się poprzez ich skopiowanie z istniejących systemów, przez wyszukanie ich w sieci lub poprzez zakup na rynku.

4.1.1.1. Kopiowanie danych rzeczywistych

W przypadku kopiowania danych testowych należy mieć na uwadze, że powinny być one przechowywane z taką samą uwagą i poziomem bezpieczeństwa jak w przypadku zwykłych danych produkcyjnych. Nagłośniony przez media przypadek, kiedy to z serwera testowego wykop.pl wyciekły nazwy i hasła tysięcy użytkowników, wymusił nie tylko zmiany haseł w samym serwisie, ale również w innych serwisach, takich jak nasza-klasa.pl czy allegro.pl. Pokazuje to również, czym może się zakończyć brak dbałości o bezpieczeństwo danych testowych. Popularną metodą zabezpieczenia jest częściowe kopiowanie danych, tak aby przypadkowy błąd nie spowodował nieodwracalnych skutków. Często kopiowany jest jedynie fragment danych, a resztę uzupełnia

się ustalonym wcześniej lub pseudolosowym ciągiem znaków. Przykładowy adres e-mail *jan.kowalski@gmail.com* można zamienić na *jan.kowalski@adresemail.com* lub na *nazwa.uzytkownika@gmail.com*.

Dzięki takim technikom Jan Kowalski posiadający konto w Gmail nie będzie narażony na otrzymanie przypadkowo wysłanej wiadomości z serwera testowego. Przy wydostaniu się danych na zewnątrz nie będzie również powiązania między nazwą użytkownika a przypisanym do niego hasłem. Działania takie mają na celu ochronę prywatności użytkowników.

Dane wrażliwe dla organizacji mogą być dodatkowo maskowane. Dzięki algorytmom kryptograficznym mogą być szyfrowane w procesie kopiowania z produkcji do baz danych testowych.

Część danych nie powinna być w ogóle kopiowana z serwerów produkcyjnych, aby nie narażać firmy lub jej pracowników na trudne sytuacje lub nawet niebezpieczeństwo. Wyobraźmy sobie, że z korporacji wyciekają dane pracowników wraz z ich adresami i zarobkami. Może to wywołać zazdrość wśród współpracowników w firmie lub też narazić pracowników na zainteresowanie światka przestępczego.

Często nie ma konieczności kopiowania wszystkich danych, szczególnie jeśli dane rzeczywiste można łatwo wygenerować. Jeśli wiemy, że najniższa pensja w firmie wynosi X, a najwyższa Y, to na potrzeby testów możemy wygenerować losowe wartości z przedziału <X; Y> i przypisać je do rzeczywistych pracowników.

4.1.1.2. Szukanie danych rzeczywistych w internecie

Dla średnio zaawansowanych użytkowników internetu znalezienie danych w sieci nie powinno być kłopotem. Internet to wielki śmietnik, w którym można znaleźć wszystko. Wystarczy odpowiednio szukać. Istnieje wiele serwerów, które przechowują dane lub nawet udostępniają je za darmo. Pozyskując dane, należy jednak mieć na względzie prawa autorskie ich twórców oraz ochronę danych osobowych. Istnieje pokusa, aby kopiować z internetu, chociażby dostępne tam adresy e-mailowe. Należy jednak pamiętać, że są to dane, których bez zgody posiadaczy nie można przechowywać ani tym bardziej przetwarzać.

4.1.1.3. Kupowanie danych rzeczywistych

Dane rzeczywiste są przechowywane przez różne instytucje i to od nich powinniśmy je pozyskiwać. Na rynku istnieje tylko kilka firm, które oferują takie dane. Komercyjne podmioty mają prawa do handlu danymi teleadresowymi czy demograficznymi. W zależności od naszych potrzeb należy zdefiniować typ danych, jakie chcemy pozyskać, i określić ich dostawcę. Część danych jest chroniona ustawą o ochronie danych osobowych i musimy się upewnić, że firmy oferujące takie dane mają do tego odpowiednie uprawnienia.

Kupując tą książkę, zakupili Państwo również bazy danych rzeczywistych typowych dla Polski. Nie są to wszystkie dane, a jedynie takie, które nie podlegają ochronie danych osobowych. Są one częścią dołączonego do książki generatora danych testowych.

4.1.2. Powiązania między danymi rzeczywistymi

Istnieją powiązania pomiędzy niektórymi danymi rzeczywistymi. Zależności te mogą być silne lub jedynie iluzoryczne (tabela 4.1). Data urodzenia ma wpływ na PESEL, jaki otrzymuje dana osoba, i jest to przykład silnej zależność między danymi. Z drugiej strony istnieje możliwe powiązanie między danymi nie na poziomie regulacji, a na poziomie statystyki. Okazuje się np., że ulubione hasło w systemach informatycznych to imię użytkownika. Można więc powiedzieć, że istnieje słabe powiązanie między imieniem a hasłem.

4.2. Generacja danych

W zależności od dostępnych narzędzi, aplikacji, form zapisu wymagań czy też technik budowania interfejsu możemy zastosować różne rozwiązania w zakresie generacji danych testowych.

4.2.1. Generacja danych w oparciu o funkcję random

Dane zazwyczaj tworzone są w oparciu o popularną i występującą w wielu językach programowania funkcję — `random`. Umożliwia ona generowanie losowych znaków i (lub) wartości w określonych formatach i o odpowiedniej długości. Użycie tej funkcji stało się już podstawą wielu narzędzi do budowania danych testowych.

Dostępne w Polsce rozwiązania do produkcji danych nie są w 100% adekwatne do lokalnych warunków. W większości przypadków funkcjonują wewnętrzne rozwiązania firmowe, pisane przez programistów lub testerów.

W internecie można znaleźć setki rozwiązań służących do generowania danych w formatach typowych dla krajów anglosaskich. Popularnym przykładem takiego rozwiązania jest skrypt wyprodukowany przez Black Sheep Web Software o nazwie GenerateData. Skrypt w wersji spolszczonej dostępny jest na stronie *http://generatordanych.testerzy.pl*.

Przewagą tego rozwiązania nad innymi jest jego dostępność z poziomu przeglądarki.

Najpopularniejsze funkcje generowania danych testowych to:

- Generacja plików wynikowych w różnych formatach, takich jak HTML, Excel, XML, CSV, SQL.
- Generacja różnych typów danych:
  - Dane ludzi: nazwa użytkownika, adres, numer telefonu, adres e-mail itp.
  - Dane tekstowe: losowe słowa.

	Imię męskie	Imię żeńskie	Płeć	Nazwisko męskie	Nazwisko żeńskie	Domeny internetowe	Adres E-mail	Login do poczty	Hasło	Miasto	Ulica	Kod pocztowy	Gmina	Powiat	Województwo	Państwo	Numer telefonu	Numery telefonu komórkowego	PESEL	REGON	IBAN	NIP	Nazwa/ login do systemu	Data urodzenia
Imię męskie			X	X			x	x	x														x	
Imię żeńskie			X		X				x															
Płeć	X	X		X	X																			
Nazwisko męskie	X		X																					
Nazwisko żeńskie		X	X																					
Domeny internetowe																								
Adres E-mail	x																							
Login do poczty	x																							
Hasło	x	x																						
Miasto											X	X	X	X	X	X	X							
Ulica										X		X	X	X	X	X	X							
Kod pocztowy										X	X		X	X	X	X	X							
Gmina										X	X	X		X	X	X	X							
Powiat										X	X	X	X		X	X	X							
Województwo										X	X	X	X	X		X	X							
Państwo										X	X	X	X	X	X		X	X			X	x		
Numer telefonu										X	X	X	X	X	X	X								
Numery telefonu komórkowego																X								
PESEL																								X
REGON																								
IBAN																X								
NIP																x								
systemu	x																							
Data urodzenia																			X					
Powiązanie: silne - **X** słabe/możliwe - x																								

Tabela 4.1. *Relacje między danymi rzeczywistymi*

- Dane definiowane przez użytkowników:
  - dane przyrostowe,
  - zakres numerów,
  - ciągi alfanumeryczne.
- Możliwość zapisu własnych formatów danych itd.

Generacja umożliwia produkcję zróżnicowanych danych testowych. Przy dużej ilości czasu może to pozwolić na obciążenie aplikacji różnego typu danymi.

Przy wytwarzaniu danych testowych istnieje poważne ryzyko redundancji testów. Losowy dobór danych nie daje nam pewności, że aplikacja została sprawdzona przez każdą ważną informację wejściową. Nie wiemy, czy została pokryta każda klasa równoważności i każda granica. Tracimy tylko czas na testowanie za pomocą danych tego samego lub pokrewnego typu.

4.2.2. Generacja danych z kodu

Niektóre aplikacje generują dane testowe automatycznie w oparciu o kod źródłowy. Analiza statyczna kodu umożliwia odpowiednim skryptom określenie, jakie wartości może przyjmować zmienna. Przykładowo w języku C zadeklarowanie zmiennej jako `signed char` powoduje, że może ona przyjąć wartość od –128 do 127. Dzięki temu znamy wartości graniczne dla zmiennej i możemy w łatwy sposób pokryć wszystkie klasy równoważności oraz sprawdzić działanie funkcji na granicach.

Słabą stroną generacji z kodu jest to, że nie jesteśmy w stanie automatycznie wykryć, czy programista dobrał poprawny typ zmiennej dla określonej wartości.

Przykład

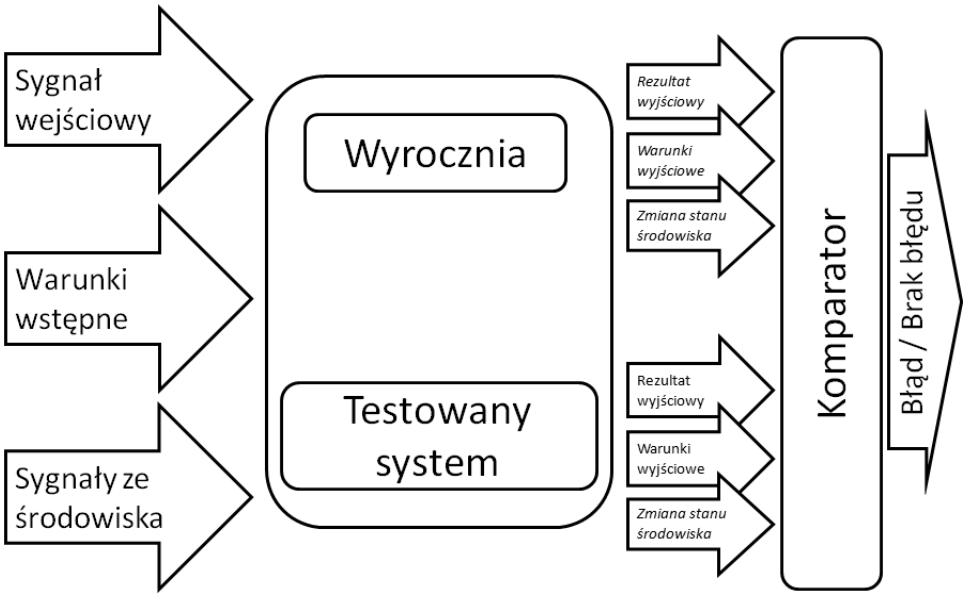
Ciekawe rozwiązanie oferuje firma Parasoft, która w swoim JTest generuje testy komponentowe w oparciu o monitorowanie pracy aplikacji. Dzięki temu rejestrowane są dane użyte podczas pracy z aplikacją i tworzy się z nich bazy danych testowych.

4.2.3. Generacja danych z dokumentacji

Zdarza się, że dokumenty projektowe, takie jak wymagania czy wytyczne dotyczące pracy aplikacji, zapisane są w języku maszynowym lub też zamodelowane w sposób umożliwiający przetwarzanie ich przez komputer. Istnieją również aplikacje umożliwiające określenie na bazie takiej dokumentacji wytycznych do danych testowych lub nawet bezpośrednio je generujące. Językiem modelowania, który idealnie się do tego nadaje, jest np. UML (ang. *Unified Modelling Language*). Składa się on z szeregu diagramów, za pomocą których można zapisywać między innymi zachowanie aplikacji i przepływ danych przez nią.

4.2.4. Generacja danych w oparciu o wyrocznię

W testowaniu znana jest technika generowania danych w oparciu o wyrocznię testową. Jest to metoda używana przede wszystkim do generowania oczekiwanego rezultatu, ale używa się jej również do generacji danych testowych. Wyrocznia jest oprogramowaniem lub — jeśli wolimy — sprzętem, który jest w stanie wygenerować na wyjściu poprawny, oczekiwany rezultat. W praktyce wyroczniami są zazwyczaj poprzednie wersje oprogramowania lub też specjalnie spreparowane układy logiczne (rysunek 4.1).



Rysunek 4.1. *Środowisko pracy wyroczni testowej*

Na wejściu wyroczni i testowanego systemu podaje się te same informacje i sprawdza się je na wyjściu. Jeśli informacje są identyczne, to komparator klasyfikuje je jako poprawne zachowanie testowanej aplikacji. W przypadku różnicy między wyjściem wyroczni i testowanego systemu komparator klasyfikuje je jako potencjalny błąd.

Gdy używamy wyroczni do generowania danych testowych, zakładamy, że system ten działa dokładnie tak, jak powinien działać tworzony przez nas system. Podając na wejściu wyroczni zestaw danych, otrzymujemy na wyjściu ich akceptację lub też komunikaty o błędach. W zależności od informacji zwrotnych z aplikacji dane klasyfikujemy jako pozytywne lub negatywne. Możemy je również podzielić na klasy równoważności odpowiadające wyjściowemu sygnałowi z aplikacji.

4.3. Generowane czy rzeczywiste?

Firma Grid-Tools przeprowadziła badania mające na celu wykazanie, jakie dane testowe dają największe pokrycie w testach. Przy tym samym zestawie przypadków testowych przebadano kilka zestawów danych testowych i w rezultacie otrzymano następujące wyniki pokrycia:

- Dane produkcyjne — 18% pokrycia.
- Bazy danych testowych — 21% pokrycia.
- Bazy danych testowych w narzędziu QTP Load Runner — 24% pokrycia.

Okazało się, że przydatność standardowych danych jest (delikatnie mówiąc) niezadowalająca. Dane te są zazwyczaj nadmiarowe i zawierają dużą liczbę powtarzających się przypadków. Potrzebne są więc generatory danych dopasowanych do konkretnego projektu lub też odpowiednio spreparowane dane.

W teorii dane rzeczywiste są podzbiorem danych generowanych. Istnieje niewielkie prawdopodobieństwo wygenerowania danych rzeczywistych przy wystarczająco dużej liczbie prób. W skończonym czasie, jaki mamy na przetestowanie oprogramowania, nie możemy sobie pozwolić na ryzyko testów w oparciu jedynie o dane generowane. Tabela 4.2 prezentuje różnice i podobieństwa między omawianymi danymi.

Tabela 4.2. *Dane rzeczywiste i generowane — różnice i podobieństwa*

Rzeczywiste	Generowane
Dane występujące w realnej pracy z aplikacją	Losowe, przypadkowe, niesystematyczne
Zbiór skończony — istnieje skończona ilość danych testowych	Zbiór nieskończony — istnieje możliwość nieskończonej generacji danych testowych
Dane pozytywne	Dane pozytywne oraz dane negatywne
Wysoka skuteczność walidowania wymagań klienta	Wysoka skuteczność walidowania zabezpieczeń w miejscach wprowadzania danych przez użytkownika
Podstawa testów funkcjonalnych i wydajnościowych	Podstawa testów funkcjonalnych, wydajnościowych i użytecznościowych
Dane używane zazwyczaj w testach akceptacyjnych aplikacji	Dane używane zazwyczaj w testach eksploracyjnych aplikacji

W praktyce testy warto rozpocząć od danych rzeczywistych, gdyż pomagają one wykryć podstawowe błędy funkcjonalności. Większość danych wprowadzanych przez użytkowników to właśnie dane rzeczywiste. Warto je zastosować, ponieważ gwarantują najwyższe prawdopodobieństwo wykrycia defektów, które mogą ujawnić się użytkownikom. Ma tu zastosowanie zasada Pareto, zgodnie z którą 80% użytkowników używa 20% funkcjonalności aplikacji. Oznacza to, że większość z nich będzie wykonywała jedynie podstawowe operacje z aplikacją i nie będzie zmuszała jej do popisów ekwilibrystycznych. W użyciu będą przede wszystkim te dane, które są powszechne i popularne.

Duże znaczenie ma również grupa odbiorców. Aplikacje pisane dla urzędów zazwyczaj będą traktowane z niewiarą i dystansem. Dane będą wprowadzane do systemu techniką jednego lub dwóch palców i z uwagą, na jaką taka technika pozwala. Obserwacje użytkowników pokazują, że błędy aplikacji traktują oni jako swoją pomyłkę i nie szukają alternatywnych rozwiązań. Dobitnie pokazuje to przykład pewnej urzędniczki z ZUS-u, która, obsługując aplikację naliczającą emerytury, znalazła błąd związany z nietypowym przypadkiem jej użycia. W uproszczeniu polegało to na nieuwzględnieniu w regułach kalkulowania emerytur wytycznych dotyczących pewnej wąskiej grupy emerytów. Pomimo że błąd został znaleziony, nie został zgłoszony do działu informatycznego. Wypłaty były kalkulowane ręcznie, poza systemem informatycznym. Ten typ aplikacji i rodzaj odbiorców wymusza przede wszystkim testy z danymi rzeczywistymi w wielu możliwych kombinacjach.

Z kolei oprogramowanie dedykowane specjalistom-informatykom będzie na pewno dokładnie sprawdzone. Pola aplikacji zostaną z dużym prawdopodobieństwem zweryfikowane pod kątem wszelkich możliwych kombinacji znaków, cyfr i liczb. W naturze doświadczonych użytkowników leży konstruktywny pesymizm połączony z chęcią udowodnienia, że aplikacja nie działa tak, jak powinna. Widać to chociażby w gorliwości, z jaką punktowany jest Microsoft za każdy błąd w swoim flagowym produkcie — Windowsie. Doścignięcie fantazji użytkowników wymaga w tym przypadku generowania danych testowych. Jedynie maszyna jest bowiem w stanie przygotować dane, które mogą zasymulować ich inwencję.