DANE TESTOWE W SYSTEMACH INFORMATYCZNYCH

TEST DATA IN INFORMATION SYSTEMS

Anna Piaskowy, *Politechnika Śląska*
Radosław Smilgin, *Politechnika Gdańska*

Abstract

The article contains the characteristics of the test data found in computer systems. This paper presents the functional classification of software test, and their brief description. Addresses the problem of automated tests. An attempt was made to classify most of the Polish systems. Article contains some of their analysis in terms of usefulness in the process of testing software in order to increase application performance.

Streszczenie

Artykuł zawiera charakterystykę danych testowych występujących w systemach informatycznych. W artykule przedstawiono klasyfikację funkcjonalną testów oprogramowania oraz ich krótką charakterystykę. Poruszono zagadnienie testów automatycznych. Podjęto próbę sklasyfikowania najpopularniejszych danych w polskich systemach informatycznych. Artykuł zawiera częściowo ich analizę pod kątem przydatności w procesie testowania oprogramowania w celu zwiększenia wydajności aplikacji.

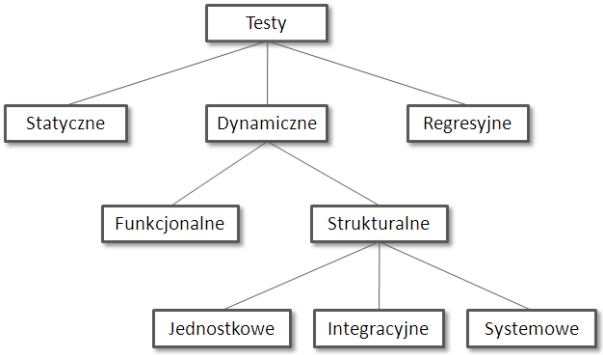
1. Wstęp

Podczas procesu tworzenia oprogramowania niezbędna jest kontrola jakości. Ocena jakości oprogramowania sprowadza się w ogólnym rozumieniu do oceny jakości eksploatacyjnej, czyli oceny charakterystyki produktu – oprogramowania, pod kątem spełnienia aktualnych wymagań użytkownika. W celu ustalenia jakości oprogramowania należy przeprowadzić jego weryfikację oraz walidację. Testowanie weryfikacyjne oprogramowania ma na celu sprawdzenie jego zgodności ze specyfikacją. Natomiast testy walidacyjne sprawdzają czy wytwarzane oprogramowanie spełnia oczekiwania użytkownika.

Testowanie oprogramowania umożliwia wykrycie błędów na każdym etapie tworzenia oprogramowania, oczywiście im później wykryty zostanie błąd tym większy będzie koszt jego usunięcia. Głównymi źródłami błędów są specyfikacja i projekt, dlatego testy przeprowadza się już podczas procesu wytwarzania oprogramowania.

2. Rodzaje testów oprogramowania

Klasyfikacja rodzajów testów według techniki ich wykonania jest przedstawiona na rysunku 1.



Rys.1. Klasyfikacja testów oprogramowania według techniki wykonania.

Fig.1. The classification of software test.

Testy dynamiczne oparte są na wynikach programu wykonywanego. Mogą to być testy statystyczne jeśli założy się że generowane dane wejściowe będą miały charakter losowy i określi się reguły poprawnego działania programu.

Testy funkcjonalne są nazywane często testami czarnej skrzynki (ang. black box testing) – są to testy działania programu bez znajomości jego struktury

Testy strukturalne są to testy oparte o znajomość struktury programu.

Testy jednostkowe są to testy oprogramowania na poziomie pierwotnym działania poszczególnych funkcji i metod

Testy integracyjne polegają na testowaniu interfejsów wydzielonych części i modułów programu.

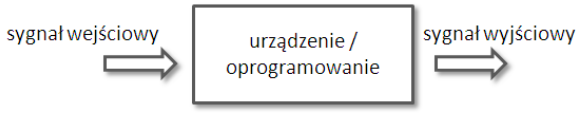
Testy systemowe jest to testowanie działania aplikacji jako całości. Testy te obejmują wymagania niefunkcjonalne tj.: szybkość działania, bezpieczeństwo, niezawodność, dobrą współpracę z innymi aplikacjami i sprzętem.

Testy regresyjne – testy przeprowadzane po wprowadzeniu poprawek, mają na celu sprawdzenie czy dodając nową funkcjonalność oprogramowania nie naruszono innej. Powinny być wykonywane na poziomie kodu (testy jednostkowe) oraz na wyższym poziomie działania całej aplikacji (testy funkcjonalne).

Test jest próbą odpowiedzi na pytanie jaka jest miara niezawodności danego oprogramowania. Testowaniu podlegają wydajność systemu oraz jego funkcji. W procesie testowym sprawdzane jest zużycie zasobów, niezawodność, utrzymywalność, skalowalność oprogramowania i możliwość przenoszenia, jego bezpieczeństwo oraz jakość dokumentacji. Badane są interfejsy pod względem zgodności z wymaganiami, mierzona jest obciążalność, zdolność wychodzenia z katastrof oraz inne parametry.

3. Przypadek testowy

Przypadek testowy jest to pojedyncza czynność, polegająca na sprawdzeniu pewnych właściwości oprogramowania lub urządzenia (rys. 2). Proces testowania zawsze polega na podaniu sygnału wejściowego, który wymusza określone działanie, w wyniku którego pojawia się sygnał wyjściowy. Proces testowania obejmuje zarówno urządzenia jak i oprogramowanie.



Rys.2. Przypadek testowy.
Fig.2. Test case.

Składowe przypadku testowego zgodnie z definicją to:

- dane wejściowe, stanowiące wymuszenie pewnego zachowania aplikacji,
- warunki wstępne, w jakich znajduje się urządzenie (oprogramowanie), na chwilę przed wymuszeniem,
- oczekiwane zachowanie urządzenia lub aplikacji w zależności od sygnału wymuszającego, które może być również sygnałem wyjściowym,
- warunki końcowe po wykonaniu wymuszenia.

Przykładowy przypadek testowania pola „nazwa użytkownika" dla portalu internetowego będzie składał się z następujących kroków:

- otwarcie strony logowania portalu http://stronaX.com/logowanie,
- uzupełnienie pola „nazwa użytkownika" daną X, gdzie X może być dowolną nazwą,
- uzupełnienie pola „nazwa użytkownika" daną X, gdzie X może być dowolną nazwą użytkownika,
- zatwierdzenie nazwy użytkownika, przez naciśnięcie klawisza „Potwierdź".

Dla powyższego przypadku oczekiwany, pozytywny rezultat zachowania aplikacji to brak komunikatu błędu, oznaczający, że nazwa została zaakceptowana. W przypadku testów negatywnych zakłada się, że poprawnym zachowaniem aplikacji będzie zwrócenie komunikatu błędu. Podstawiając pod X wcześniej przygotowane dane testowe można sprawdzić wiele różnych przypadków testowych.

Innym przykładem przypadku testowego może być podanie ciągu znaków składającego się na rzeczywiście istniejącą nazwę osoby „nazwisko". Ciąg ten ma pewne właściwości i pomimo, że nazwiska podlegają ochronie danych osobowych dostępność do nich jest praktycznie nieograniczona. Nie są one bowiem przypisane do jednej osoby, a do szeregu różnych osób. Nazwiska, z drugiej strony, są skończonym zbiorem znaków regulowanych polskim prawem.

Przeprowadzono test na testerach pokazujący jakie dane są zazwyczaj wprowadzane przy wykonywaniu testowania pola opisanego etykietką „nazwisko". Doświadczenie pokazuje, że najpopularniejsze typy danych testowych to:

- Qwerty,
- Adsdwedwe,
- Dqw3/efew.

Poddając takie dane skróconej analizie łatwo określić, dlaczego mają one znikomą skuteczność.

Ciąg znaków „Qwerty" jest najbardziej typową daną testową zaraz obok 123456. Bierze się to z łatwego wprowadzania tych danych z klawiatury (pierwszy rząd przycisków patrząc od lewej strony). Wadą takiej „symulacji" jest jednak brak powiązania ciągu znaków z typowymi ciągami znaków w polskim nazwisku. Co prawda Q istniało w staropolszczyźnie, ale nie jest już powszechnie używane. Ilość osób noszących znak „Q" w nazwie jest więc zbiorem o niewielkiej próbie. Powoduje to, wprowadzenie danej o niskiej skuteczności zweryfikowania rzeczywistego przypadku użycia. Oczywiście patrząc na QWERTY z drugiej strony, jest to ulubiony ciąg znaków wprowadzany przez internautów. Zazwyczaj pojawia się, jeśli zmusza się ich do podawania swojego nazwiska podczas logowania do strony internetowej. Jest to tak zwany przypadek testowy negatywny, którego używa się po sprawdzeniu przypadków pozytywnych, czyli najbliższych rzeczywistości.

Ciąg znaków „Adsdwedwe" jest ciągiem losowym. Jest on na pierwszy rzut oka absolutnie poprawną daną testową, ale również konsekwencją losowego naciskania przycisków klawiatury. Podany

ciąg znaków może spowodować błąd aplikacji, a ponieważ był on wygenerowany losowo nie ma możliwości jego odtworzenia w celu usunięcia defektu.

Trzeci przypadek czyli błędnie wpisane dane zawierające inne znaki. Podobnie, jak w powyższych przykładach, może się okazać, że istnieje na świecie ktoś, kto będzie miał wartość liczbową (3) lub z angielskiego slash (/) w nazwisku. Jednak regulacje prawne w Polsce uniemożliwiają używanie cyfr w nazwie osoby. Dlatego tak wprowadzona informacja może być w prosty sposób eliminowana przez walidator pola.

Te trzy proste przykłady pokazują najpopularniejsze dane i ich niską przydatność do testów. Używając podobnych wzorców obniża się jakość testowania w procesie tworzenia aplikacji. Wskazywanie błędów w aplikacji jest złożonym procesem i nie powinno być działaniem nie powiązanym z rzeczywistością, przypadkowym i dodatkowo obarczonym błędami.

4. Automaty testowe

Ciągłe powtarzanie tych samych czynności na ogół zastępuje się wykonywaniem skryptu, który ułatwia pracę testera. Na rysunku 3 przedstawiono przykład testu automatycznego w aplikacji Selenium IDE:

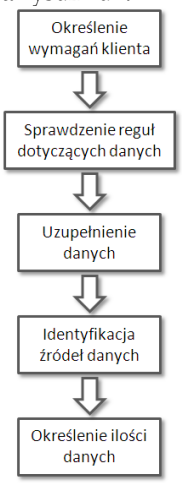
Command	Target	Value
loadTestData	file://d:/selenium/user_name.xml	
while	!testdata.EOF()	
nextTestData		
open	http://forum.testerzy.pl/ucp.php?mode=register	
clickAndWait	agreed	
type	username	${phrase}
clickAndWait	submit	
assertTextNotPresent	Wprowadzona nazwa użytkownika jest	
endWhile		

Rys.3. Test oparty na danych dla aplikacji Selenium IDE.
Fig.3. The test based on data for the application Selenium IDE.

Dane testowe pobierane są z zewnętrznego pliku, gdzie zostają uprzednio zapisane, lub do którego są wygenerowane. Następuje otwarcie pętli i pobieranie kolejnych wpisów z pliku. Aplikacja automatycznie otwiera stronę testową i we wskazane miejsce „wkleja" skopiowaną z pliku wartość. Aplikacja przez modyfikację kodu strony symuluje naciśniecie klawisza. Pojawiający się komunikat potwierdzający jest weryfikowany. Pętla kończy się, kiedy kończą się wpisy w pliku. Jest to przykład, jak bazując na jednym przypadku testowym i wielu danych można w sposób automatyczny zweryfikować poprawność działania pojedynczego pola aplikacji.

5. Proces definiowania danych testowych

Definiując dane testowe należy określić w jaki sposób będzie testowana dana aplikacja. Dane testowe mogą być rzeczywiste bądź sztuczne, będące przykładowo optymalnym zestawem znaków dla danego przypadku testowego. W uproszczeniu proces definiowania danych składa się z czynności przedstawionych na rysunku 4.



Rys.4. Proces definiowania danych testowych.
Fig.4. The process of defining the test data.

Pierwszą czynnością w procesie identyfikacji zestawu danych, potrzebnych w procesie testowania jest określenie wymagań klienta dla danej aplikacji. Dzięki temu można określić większość pozytywnych przypadków testowych i powiązanych z nimi danymi. Oczywiście dane testowe nie pojawiają się jako osobny rozdział specyfikacji wymagań. Wyszukanie danych to złożony proces analizy wytycznych klienta lub zleceniodawcy.

Ukryte wymaganie dotyczące danych zostanie przedstawione na przykładzie numeru identyfikacyjnego pracowników pewnej firmy. Każdy pracownik ma przypisany identyfikator w systemie kadrowym składający się zawsze z litery „P", następnie litery identyfikującej dział w jakim pracuje (B-biuro, P-produkcja, S-sprzedaż). W dalszej części znajduje się cztero-cyfrowy numer porządkowy. Pierwszy identyfikator ma numer 0001 a ostatni będzie miał numer 9999. Numer jest generowany przez system kolejno w zależności od dostępności. Opierając się o tak zdefiniowane dane wiadomo, że ciąg identyfikujący pracownika ma 6 znaków. Pierwszy to litera P, drugi to litera B, P lub S, kolejne znaki mogą przyjmować wartości od 0 do 9. Dane, takie nie są regulowane przepisami prawnymi tak jak nazwiska czy numer PESEL. Mogą jednak istnieć wewnętrzne regulacje, które nakładają klasyfikację według pewnych z góry zdefiniowanych zasad. Przykładowo, pracownicy, którzy rozwiązują umowę o pracę nie są usuwani z systemu księgowego, a ich numery nie są zwracane do puli dostępnych identyfikatorów. Tak więc część numerów pomimo istnienia w systemie będzie już realnie rzadko używana.

Kolejny krok to uzupełnienie zbioru danych o dane testowe. Przykładową wartością dla przykładu opisanego powyżej jest PS0000 czyli przypadek negatywny, nie będący częścią zdefiniowanych identyfikatorów pracowników. W zbiorze danych uwzględnione powinny być również dane testowe, w których w miejsce pierwszej litery P wstawi się inny znak, w tym wartość liczbowa. Ilość takich przypadków testowych będzie dążyła do nieskończoności, dlatego należy poprzez analizę ograniczyć je do niezbędnego minimum.

Gdy znane są już dane testowe, należy określić źródło ich pochodzenia. Dane mogą być skopiowane z aktualnie używanego systemu. Stanowią wtedy w pełni rzeczywiste odwzorowanie danych produkcyjnych. Mogą również zostać wygenerowane według zdefiniowanego wzorca zarówno dla przypadków pozytywnych jak i negatywnych. Następnie należy określić minimalną, maksymalną i optymalną ilość danych testowych.

Tak przygotowane dane stanowią solidny fundament w dalszych testach oprogramowania, bez względu na poziom. Nadają się zarówno do testów jednostkowych, jak i dla późniejszych testów akceptacyjnych.

6. Typy danych

Dane występujące w systemach informatycznych można podzielić na dane:

- pozytywne,
- negatywne.

Pozytywne dane testowe, są to dane, które powinny być akceptowane w systemach informatycznych, stanowią one jednocześnie podstawę do testów oprogramowania. System po podaniu tych danych nie powinien zwrócić komunikatu błędu. Dane te tworzą zbiór skończony.

Negatywne dane testowe są to dane, które nie powinny być akceptowane w systemach informatycznych, które stanowią jednocześnie podstawę do testów oprogramowania. System po podaniu tych danych powinien zwrócić komunikat błędu. Dane te tworzą zbiór skończony.

Ze zbioru danych można wyselekcjonować dane optymalne, które znamiennie skrócą czas testów. Dane optymalne są wartościami alfanumerycznymi, których skończona ilość pomaga w szybkim przeprowadzaniu testów. Ze względu na ich specyfikę wyniki badań wykazały, że dane optymalne są w stanie wychwycić do 80% najważniejszych błędów dla danego typu pola tekstowego.

7. Walidacja danych

Bardzo ważną kwestią dotyczącą wyboru danych testowych jest ich poziom walidacji w systemach informatycznych. W jednym systemie dana może być akceptowana, w innym może być odrzucana. Dane optymalne są jedynie pewną sugestią i powinny być dopasowane do systemów informatycznych.

W systemach informatycznych istnieją zasady, które muszą być brane pod uwagę przy wprowadzaniu danych do pola. Dla tzw. „pól wymaganych" konieczne jest wprowadzenie danej testowej będącej ciągiem znaków. Pozostawienie pustego pola może powodować zwrócenie przez system komunikatu błędu. Odrębne zagadnienie stanowi pojemność pola, czyli ilość znaków jakie można wprowadzić do danego pola. Każde pole ma swoje ograniczenia, dlatego ilość możliwych znaków wprowadzonych jest zawsze skończona. Przekroczenie pojemności pola powinno skutkować zwróceniem komunikatu błędu. Istnieje jednak niebezpieczeństwo, ponieważ część danych w systemach informatycznych nie ma ograniczeń długości, np. hasło, wtedy muszą być stosowane ograniczenia lokalne na poziomie każdego systemu informatycznego. Ważne jest, aby powiadomienie o ograniczeniach dla użytkowników systemu było przekazywane w sposób użyteczny w rozumieniu użyteczności, czyli w sposób jak najbardziej przyjazny użytkownikowi.

Należy uwzględnić sposób podawania danych przez użytkowników. Istnieją różne metody walidacji danych i jest niezbędne, aby pokryć je przypadkami testowymi podczas testowania systemów. Niektóre systemy walidują i automatycznie zmieniają błędnie podane dane, np. kiedy użytkownik wklei w pole tekstowe zmienną ze spacją na końcu docelowo jest ona automatycznie wycinana.

8. Analiza danych – najpopularniejsze dane w systemach informatycznych

Podczas analizy najpopularniejszych danych występujących w systemach informatycznych testom poddano 1000 różnorakich aplikacji, zarówno produkcyjno-biurowych dostępnych ograniczonej ilości użytkowników jak i aplikacji internetowych dostępnych dla wszystkich użytkowników globalnej sieci. Wśród przetestowanych systemów znalazły się miedzy innymi:

- polskie portale: Onet.pl, WP.pl, gazeta.pl,
- polskie portale społecznościowe: nasza-klasa.pl, grono.net,
- aplikacje księgowe: Symfonia, RAKS, CDN,
- aplikacje do zarządzania zasobami: SAP ERP,
- aplikacje do prowadzenia firm: WAPRO,
- aplikacje do zarządzania klientami: ChromeCRM, Imagine.CRM..

Analiza zawężała się do aplikacji dedykowanych pojedynczym użytkownikom operujących w Polsce, dlatego też dane można traktować jako specyficzne jedynie dla Polski.

Analiza wykazała, że najpopularniejsze dane w systemach informatycznych to:

- polskie imię żeńskie,
- polskie imię męskie,
- polskie nazwisko męskie,
- polskie nazwisko żeńskie,
- polskie domeny internetowe,
- polski adres e-mail,
- login do poczty,
- hasło,
- polskie miasto,
- ulica,
- kod pocztowy,
- gmina,
- powiat,
- województwo,
- państwo,
- wykształcenie,
- numer telefonu stacjonarnego,
- numery telefonu komórkowego,
- PESEL,
- REGON,
- IBAN,
- NIP,
- alfanumeryczne ciągi znaków,
- nazwa/login do systemu,
- data urodzenia,
- data (inna).

Dane te są zbiorem nieuporządkowanym tak więc pozycja na liście nie informuje o tym jak popularna jest dana. Z uwagi na złożoność i obszerność takiej analizy danych w artykule zostanie przybliżona metodyka badań na przykładzie nazwisk i numerów PESEL.

8.1. Analiza nazwisk w Polsce

Analiza nazwisk w Polsce prowadzi do następujących wniosków:

- najpopularniejszym nazwiskiem jest NOWAK,
- najkrótszym nazwiskiem jest AK,
- najdłuższym nazwiskiem jest ACHMISTROWICZ-WACHMISTROWICZ,
- pierwsze nazwisko w alfabetycznym spisie to AAB,
- ostatnie nazwisko w alfabetycznym spisie to ŻYŻYŃSKI.

Przy analizie nazwisk zagranicznych, które nie są regulowane tak restrykcyjnymi prawnymi zasadami jak w Polsce mogą pojawić się egzemplarze, które znacząco zwiększą zbiór danych testowych. Przykładowe, najdłuższe nazwisko zagraniczne, które przy okazji nadaje się również do książki rekordów Guinessa to:

Wolfeschlegelsteinhausenbergerdorffvoralternwarengewissenhaftschaferswesenchafewarenwholgepflegeundsorgfaltigkeitbeschutzenvonangereifenduchihrraubgiriigfeindewelchevorralternzwolftausendjahresvorandieerscheinenbanderersteerdeemmeshedrraumschiffgebrauchlichtalsseinursprungvonkraftgestartseinlangefahrthinzwischensternartigraumaufdersuchenachdiesternwelshegehabtbewohnbarplanetenkreisedrehensichundwohinderneurassevanverstandigmenshlichkeitkonntevortpflanzenundsicherfreunanlebenslamdlichfreudeundruhemitnichteinfurchtvorangreifenvonandererintlligentgeschopfsvonhinzwischensternartigraum.

Choć powyższa dana testowa wydaje się absurdalna, powinno się ją uwzględniać. Niestety przy takiej danej większość znanych autorom aplikacji zwróci błąd, tak więc człowiek o takim nazwisku będzie miał problem z założeniem chociażby konta e-mail.

Pozostając jednak na polskim gruncie, długość nazwisk aktualnie to przedział od dwóch do dwudziestu siedmiu znaków. Pamiętając jednak o ludzkiej pomysłowości, warto ten zakres zwiększyć do 32 znaków. Wiedząc, że dane powinny być cyklicznie aktualizowane warto okresowo sprawdzać, czy nie pojawiają się dłuższe nazwiska.

8.1.1. Regulacje prawne

Zgodnie z ustawą o nazwiskach [4] zdefiniowano zasady odnoszące się do budowania nazwisk:

- nazwiska nie zawierają cyfr,
- nazwiska nie zawierają znaków innych niż litery oraz jeden myślnik „-”,
- nazwiska nie mogą składać się z więcej niż dwóch członów,
- nazwiska dwuczłonowe zawierają jedną spację lub dwie spacje (w zapisie formalnym obok myślnika),
- nazwiska w Polsce zaczyna się od wielkiej litery przy czym nazwisko pisane samymi dużymi literami również powinno być akceptowane.

Tak przeprowadzona analiza pomaga zrozumieć specyfikę danej testowej oraz pozwala przeanalizować wszystkie najbardziej prawdopodobne przypadki testowe.

8.1.2. Przypadki testowe

Jeżeli możliwych znaków w nazwisku jest 32 to ilość wszystkich pozytywnych przypadków testowych wynosi 32^{32}, czyli $1{,}46150164 \times 10^{48}$. Aby ograniczyć tą ilość do minimum należy wybrać z tego zbioru następujące przypadki testowe.

Pozytywny:

- prosty – NOWAK,
- optymalny – BAŹ – DĘTOWSKA,
- ekstremalny– AĄBCĆDEĘFGHIJKLŁMNŃOÓPRSŚTUWYZŹŻ.

Negatywny:

- prosty - puste pole (zero znaków),
- ekstremalny - „AĄBCĆDEĘFGHIJKLŁMNŃOÓPRSŚTUWYZŹŻ”,
- liczbowy - „0123456789”,
- znaki specjalne - „~`!@#$%^&*()_+={}[]:;”’|\|<,>.?/”,
- dwie spacje (jedna za drugą) - „ab ba”,
- dwa myślniki - „--” lub „aa-ab-ba”.

Minimalna ilość przypadków testowych wynosi 1. Zweryfikowanie poprawności implementacji danego pola formularza wymaga wykonania minimalnie jednego przypadku testowego.

Optymalna ilość przypadków testowych wynosi 10. Badania pokazują, że optymalną ilością przypadków testowych jest wykonanie 3 przypadków pozytywnych i 7 negatywnych. Każdy z nich może testować inną walidację dla danego pola. Przykładowo wartość liczbowa sprawdzi czy aplikacja przyjmie nazwisko złożone z cyfr.

Maksymalna ilość przypadków jest nieskończona. Uwzględniając, że ilość możliwych znaków jest skończona, ale ilość znaków możliwych do wprowadzenia jest nieskończona, również ilość przypadków zmierza do nieskończoności.

8.2. Analiza numerów PESEL

Numer PESEL (skrót od Powszechny Elektroniczny System Ewidencji Ludności) jest numerem unikalnym nadawanym przez ministra właściwego do spraw wewnętrznych, w całej Polsce nie ma dwóch takich samych numerów, więc jest to dana testowa, która nie powtarza się w systemach informatycznych. W każdym systemie, czy aplikacji może wystąpić tylko raz. Korzystając z reguł tworzenia takich numerów rozróżnia się numery będące zbiorem danych testowych optymalnych pozytywnych, optymalnych negatywnych, pozytywnych oraz negatywnych. Dane pozytywne tworzy się według reguł opisanych poniżej. Numer PESEL stanowi 11 cyfr:

- cyfry [1-6] to data urodzenia (rr-mm-dd) z określeniem stulecia urodzenia. Dla osób urodzonych w latach 1900 -1999 – miesiąc zapisywany jest w sposób naturalny, dla osób urodzonych w innych latach dodawane są do numeru miesiąca następujące wielkości:
  - dla lat 1800-1899 – 80,
  - dla lat 2000-2099 – 20.
- Cyfry [7-10] to numer serii z oznaczeniem płci,
- cyfra [10] to płeć, cyfry parzyste (0, 2, 4, 6, 8) oznaczają płeć żeńską, a cyfry nieparzyste (1, 3, 5, 7, 9) oznaczają płeć męską,
- cyfra [11] jest liczbą kontrolną

Przykładem danej pozytywnej jest numer: 44051401458. Warunkiem poprawności numeru PESEL jest zapis bez spacji i myślników.

Dane testowe optymalne negatywne będą wszystkimi pozostałymi danymi nie zawierającymi się w zbiorze danych optymalnych pozytywnych. Oto kilka przykładów danych testowych optymalnych negatywnych:

- aaaaaaaaaaa,
- 360413-66-359,
- 360413 66 359,
- 6041366359,

Podobnie dane testowe negatywne będą należały do zbioru przeciwnego danych pozytywnych.

W przypadku numerów PESEL danych testowych rzeczywistych nie można wygenerować ze względu na ochronę danych osobowych.

9. Podsumowanie

Adekwatne dane podnoszą efektywność testów, poprzez ograniczenie czasu potrzebnego na testowanie i wykrywanie w skończonej jednostce czasu więcej defektów niż techniki oparte na losowym doborze danych.

Dobre dane podnoszą skuteczność testów w poszukiwaniu defektów dlatego, że egzaminują obszary, w których człowiek popełnia najwięcej błędów. Mowa tu zarówno o programistach tworzących kod, jak i użytkownikach, których akcje są w aplikacji walidowane.

W przeciwieństwie do przypadków testowych można z całą pewnością powiedzieć, że ilość „przydatnych” danych jest skończona. Pozwala to na podkopanie fundamentów tak popularnej wśród testerów maksymy, że testowanie nigdy się nie kończy.

Dane testowe mogą być raz zdefiniowane i wielokrotnie używane w całym procesie tworzenia i testowania aplikacji. Co więcej dane te mogą być również używane w pracach rozwojowych nad innymi aplikacjami. Należy tylko pamiętać, aby danymi testowymi zarządzać, tak jak każdą inną dokumentacją czyli kontrolować wersję i zabezpieczać przed przypadkowym usunięciem.

Literatura

1. Anna Piaskowy, *„Badania i analiza najpopularniejszych danych występujących w systemach informatycznych”*, raport dla PARP 2009
2. Radosław Smilgin „*Dane testowe. Teoria i Praktyka”*, Warszawa 2009
3. *www.testerzy.pl*
4. *„Certyfikowany tester – Plan poziomu podstawowego”*, Stowarzyszenie Jakości Systemów Informatycznych
5. Ustawa z dnia 15 listopada 1956 r. *„o zmianie imion i nazwisk”*
6. *www.wikipedia.pl*
7. *„Weryfikacja I (a) testowanie”* materiały do wykładu inżynieria oprogramowania