testerzy.pl

KORZYŚCI WYNIKAJĄCE Z ZASTOSOWANIA TWIERDZENIA BANACHA O PUNKCIE STAŁYM W PROCESIE GENEROWANIA DANYCH TESTOWYCH

Artykułu opisuje strategię generowania danych testowych wykorzystującą twierdzenie genialnego polskiego matematyka Stefana Banacha o punkcie stałym oraz zestawienia ich z innymi technikami generowania danych testowych.

Wprowadzenie

Dane testowe to wartości, które wprowadza się do testowanego systemu jako dane wejściowe dla pewnej funkcjonalności w celu weryfikacji lub walidacji jej poprawności.

W literaturze istnieje podział danych testowych na dane wymagane oraz dane niewymagane. Autorzy pozycji [19, 20] piszą, że dane testowe wymagane to takie, których podanie jest niezbędne w procesie użytkowania aplikacji. W praktyce łatwo można zauważyć, że aplikacje komputerowe odmiennie reagują na błędnie wprowadzane dane lub po prostu zawierają różne rodzaje błędów, które uniemożliwiają prawidłowe funkcjonowanie aplikacji tuż po wprowadzeniu danych do formularza, bez względu na to, czy te dane są wymagane czy nie. Dla każdej aplikacji istnieją takie zestawy danych, które z dużym prawdopodobieństwem zostaną wprowadzone podczas użytkowania aplikacji oraz takie, których użytkownik prawdopodobnie nie wprowadzi przez cały cykl życia oprogramowania [20]. Niezależnie od tego, jakie dane testowe wprowadzane są jako dane wejściowe, aplikacja musi zareagować na nie pozytywnie, a to oznacza, że dane zostaną zapisane, przetworzone, przeniesione lub aplikacja zwróci komunikat o niepoprawnych danych, który dodatkowo poinformuje użytkownika, które pole zawiera niedozwolone dane lub jaka funkcja dostała niepoprawne dane wejściowe. Istnieje w praktyce wiele technik, które pozwalają ułatwić wybór danych tak, aby były one zgodne z danymi rzeczywistymi. W pozycji [21] podane są techniki pozyskiwania danych testowych jak:

- *Testowanie dziedziny,*
- *Podział na klasy równoważności,*
- *Analiza wartości brzegowych (można analizować wartości brzegowe wejściowe oraz wyjściowe),*
- *Testowanie porycia kodu (testowanie pokrycia instrukcji, testowanie okrycia warunków logicznych),*
- *Zastosowanie tablic decyzyjnych,*
- *Zastosowanie metody pair-wise testing [24],*
- *Zastosowanie metody n-wise testing,*
- *Wykorzystanie ogólnych lub własnych matematycznych modeli funkcjonalności w systemach informatycznych [23, 25, 26].*
- *Zastosowanie strategii ewolucyjnych w procesie generowania danych testowych.*

Wspomniane strategie ewolucyjne są o tyle ciekawe, że odznaczają się takimi cechami jak *[1,2,3,4,7,8,9,10,11,12,14,15,16,17,18]*:

- *Losowość ale taka, której kierunek jest kontrolowany za pomocą funkcji dopasowania oraz procesów ewolucyjnych (krzyżowanie). Istnieją generatory danych testowych, ale dzieje się to w sposób całkowicie niekontrolowany, tzn. że nie istnieje kryterium dopasowania danych do problemu*

Istnieje w testowaniu oprogramowania takie pojęcie jak strategia testowa, czyli pewien ciąg czynności pozyskiwania informacji o przypadkach testowych, czyli również na temat danych testowych. Strategie te mogą być mieszane z technikami pozyskiwania danych wejściowych. Wg Farley`a, Humble`a oraz Romana [6, 22] dane testowe powinny w szczególności odznaczać się takimi cechami jak:

- *wydajność (może być rozumiana jako czas wykonania testu),*
- *zależność od systemu (jakość zależy od aspektów technologicznych),*
- *zupełność (stopień, w jakim te dane mają wartość dla wszystkich atrybutów w systemie),*
- *aktualność,*
- *wiarygodność,*
- *spójność (np. niesprzeczność pomiędzy danymi testowymi),*
- *precyzyjność (dotyczy testów na liczbach, w aplikacjach które wymagają dużej dokładności np. przeliczanie miar czy jednostek).*

Istnieje pewna teoria zwana teorią **No Free Lunch** [13], która mówi, że **nie istnieje jedna uniwersalna strategia, która rozwiązałaby wszystkie postawione problemy z tą samą jakością rozwiązania.** Warto wobec tego szukać nowych strategii oraz dzielić postawione problemy na klasy. Celem artykułu jest zaproponowanie niespotykanej dotąd strategii generowania danych testowych wykorzystując twierdzenie Banacha o punkcie stałym i wykazanie, że takie podejście jest tak samo wartościowe, jak wspomniane wcześniej techniki generowania danych testowych. Pisząc artykuł autor zakłada, że czytelnik zna podstawowe pojęcia związane z analizą matematyczną, przestrzeniami metrycznymi, przestrzeniami unormowanymi oraz funkcjami wielu zmiennych w przestrzeniach wielowymiarowych.

Sformułowanie problemu

Testowany program, a konkretnie funkcjonalność w programie musi zadziałać bezawaryjnie dla każdego wektora danych, które są argumentami testowanej funkcji [22]. Wobec tego nie można ograniczyć się tylko do wyboru danych testowych, które leżą w dziedzinie testowanej funkcjonalności. Należy przetestować również zachowanie programu po podaniu danych wykraczających poza dziedzinę testowanych funkcjonalności. W analizie funkcjonalnej istnieje twierdzenie Banacha o punkcie stałym, które można wykorzystać do zaprojektowania algorytmu, który generuje dane testowe w oparciu o to twierdzenie. Ciekawą cechą tego twierdzenia jest to, że dzięki zastosowaniu ciągu iterowanego oraz założeń twierdzenia istnieje możliwość zbliżania się do pewnego punktu, który można obrać jako cel. Dane dostarczone w wyniku iteracji mogą znajdować się w dziedzinie testowanej funkcji, ale nie muszą, a dzięki temu zaprojektowanie strategii w oparciu o twierdzenie Banacha o punkcie stałym jest użyteczną czynnością z punktu widzenia testowania oprogramowania.

Kilka słów o Stefanie Banachu

Ten rozdział powstał, aby uszanować jednego z najlepszych na świecie matematyków, a na dodatek Polaka, który wniósł olbrzymi wkład w rozwój tej dziedziny, głównie analizy funkcjonalnej. Twierdzenie, które będzie opisane w dalszych rozdziałach, ma olbrzymie zastosowanie w rozwiązaniu trudnych do zrozumienia problemów matematyki wyższej. Okazało się jednak, że to kluczowe w wielu problemach twierdzenie znalazło zastosowanie również w testowaniu oprogramowania.

Stefan Banach wychowywał się w rodzinie zastępczej (właścicielki pralni – Franciszki Płowej i jej córki, Marii Puchalskiej). Znał osobiście tylko swojego ojca i czasami się z nim spotykał. Zgodnie z obietnicą daną matce ojciec łożył na jego utrzymanie. Od dzieciństwa wykazywał nieprzeciętne zdolności

matematyczne i lingwistyczne. […] Matematykę studiował jako samouk. W latach 1911–1913 zaliczył egzaminem częściowym (tzw. półdyplom) dwa lata studiów na Wydziale Inżynierii Lądowej Politechniki Lwowskiej. Po wybuchu I wojny światowej pracował jako nadzorca przy budowie dróg. Nie został wcielony do armii z powodu leworęczności i wady wzroku. Po powrocie do Krakowa zarabiał na życie korepetycjami. Nadal studiował sam. […]

W 1916 dr Hugo Steinhaus zainteresował się przypadkowo spotkanym Banachem. Przechodząc Plantami w Krakowie usłyszał dwóch młodych ludzi rozmawiających o o całce Lebesgue'a. Jednym z nich był właśnie Banach. Spotkanie zaowocowało wspólną publikacją i wieloletnią współpracą. W 1920 dzięki wstawiennictwu Steinhausa Banach otrzymał asystenturę (do 1922) w Katedrze Matematyki na Wydziale Mechanicznym Politechniki Lwowskiej u prof. Antoniego Łomnickiego. W 1920 (nie mając dyplomu ukończenia studiów) doktoryzował się na Uniwersytecie Jana Kazimierza we Lwowie na podstawie rozprawy, której główne wyniki zostały potem opublikowane w pracy: Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales (Fundamenta Mathematicae, III, 1922), w której zawarł podstawowe twierdzenia analizy funkcjonalnej, nowej dyscypliny matematyki (o czym jeszcze nie wiedział). W 1922 habilitował się na Uniwersytecie Jana Kazimierza (decyzja Rady Wydziału z 30 czerwca) i 22 lipca tego roku otrzymał nominację na profesora nadzwyczajnego, a w 1927 na profesora zwyczajnego tego uniwersytetu. W 1924 został członkiem PAU. W latach 1922–1939 kierował jednym z zakładów w Instytucie Matematycznym Uniwersytetu Jana Kazimierza, rozwijając działalność naukowo-badawczą. Stał się wkrótce największym autorytetem w analizie funkcjonalnej. [27]

Podstawowe wiadomości dotyczące twierdzenia Banach o punkcie stałym

W tym rozdziale przedstawione będą podstawowe wiadomości związane z twierdzeniem Banacha o punkcie stałym. Należy wobec tego podać definicje:

Definicja 1. [19] Odwzorowanie $f: X \to X$ w przestrzeni metrycznej $< X, d >$ nazywamy odwzorowaniem zwężającym, gdy istnieje taka stała L, spełniająca nierówność $0 < L < 1$, że dla każdych $x, y \in X$ zachodzi nierówność

$$d\big(f(x), f(y)\big) \leq Ld(x, y), \quad (1)$$

oraz

Definicja 2. [19] Niech $< X, d_X >, < Y, d_Y >$ będą przestrzeniami metrycznym. Odwzorowanie $f: X \to Y$ nazywamy izometrią jeśli dla każdych $x, y \in X$ spełniony jest warunek:

$$d_Y\big(f(x), f(y)\big) = d_X(x, y). \quad (2)$$

Wspomniane w tytule rozdziału twierdzenie ma postać:

Twierdzenie 1 (Banacha o punkcie stałym). [19] Jeżeli $f: X \to X$ jest odwzorowaniem zwężającym w przestrzeni metrycznej zupełnej $< X, d >$, to ma ono dokładnie jeden punkt stały , tj. taki $x \in X$, że $f(x) = x$. Jeżeli $x_1 \in X$ jest dowolnym punktem przestrzeni X oraz $x_{n+1} = f(x_n)$ dla $n = 1,2,3, \dots$, to ciąg (x_n) jest zbieżny do punktu x oraz

$$d(x_n, x) \leq \frac{L^{n-1}}{1-L} d(x_1, x_2). \text{ (3)}$$

Ciąg (x_n) nazywamy ciągiem kolejnych iteracji odwzorowania f. Wartość $d(x_n, x)$ przedstawia błąd popełniany przez zastąpienie rozwiązania równania $f(x) = x$ przybliżonym rozwiązaniem x_n.

Z wyżej podanych definicji oraz twierdzenia wyciągnąć można pewien ważny wniosek:

Wniosek 1. Niech $(R^n, ||\cdot||)$ będzie przestrzenią unormowaną zupełną. Niech metryka w zbiorze R^n będzie generowana prze normę oraz niech odwzorowanie $f: R^n \to R^n$ dane będzie wzorem:

$$f(x) = \frac{1}{K}[(K-1)x + p], \text{ (4)}$$

gdzie $K > 1, p \in R^n$. Wtedy odwzorowanie f ma jeden punkt stały p.

Dowód. Podstawiając punkt p do równania (4), otrzymujemy wyrażenie:

$$f(p) = \frac{1}{K}[(K-1)p + p] = \frac{Kp}{K} = p,$$

z którego widać, że p jest punktem stałym. Należy jeszcze wykazać, że odwzorowanie f jest zwężające. Niech $x, y \in R^n$. Wtedy

$$\|f(x) - f(y)\| = \left\|\frac{1}{K}[(K-1)x + p] - \frac{1}{K}[(K-1)y + p]\right\| =$$
$$\frac{1}{K}\|[(K-1)x + p] - [(K-1)y + p]\| = \frac{1}{K}\|[(K-1)x - [(K-1)]\| = \frac{K-1}{K}\|x - y\|.$$

Ponieważ $\frac{K-1}{K} < 1$, odwzorowanie f jest zwężające, co należało udowodnić.

Wniosek 1 można w testowaniu oprogramowania interpretować w ten sposób, że w odpowiednich zbiorach danych można wykorzystać wzór (4) do generowania danych testowych wg schematu:

1. Generowanie rozpoczyna się od dowolnego punktu x w przestrzeni R^n,
2. W wyniku kolejnych iteracji otrzymywane są punkty o współrzędnych, które nabierają cech opisanych w poprzednim I rozdziale, jeśli punkt p jest celem, do którego należy dojść w wyniku generowania ciągu danych testowych,
3. W wyniku iteracji dostarczony zostaje ciąg, którego pierwszym elementem jest x i który jest zbieżny do punktu p.

Takie podejście jest pewną strategią, która może być wykorzystana w testowaniu w problemie analizy wartości brzegowych czy testowaniu dziedziny. Istnieją systemy, które są bardzo wrażliwe na dane, które są w nich przetwarzane. Dlatego zbliżanie się do „pewnego punktu" jest w tym przypadku dobrą strategią testową. W sytuacji testowania funkcjonalności, której dziedzina jest sumą zbiorów rozłącznych lub nie jest zbiorem spójnym, wniosek 1 również będzie miał dobre zastosowanie.

Zastosowanie twierdzenia Banacha o punkcie stałym w problemie generowania danych testowych

Załóżmy, że istnieje pewien zaprogramowany czujnik z funkcjonalnością f, która jest bardzo wrażliwa na dane wejściowe, tzn. że w zależności od dostarczonego wektora danych wejściowych system może się zachować na różne sposoby i w każdym przypadku musi zwrócić poprawny wynik. Wektory wejściowe są punktami przestrzeni R^4 z normą naturalną, a wartościami są liczby rzeczywiste. Niech naszym celem będzie punkt $p = [p_1, p_2, p_3, p_4] = [3{,}6{,}2{,}8]$, natomiast generowanie ciągu danych rozpocznie się od punktu $X_1 = [x_1, x_2, x_3, x_4] = [-10, -5{,}3{,}0]$. Przyjmijmy stałą $K = 3$. Potrzebny jest jeszcze jeden parament n, który definiuje ilość iteracji dla strategii. W oparciu o wniosek 1 strategia generowania danych testowych działa wg algorytmu:

Algorytm A1$(\boldsymbol{n}, \boldsymbol{x}, \boldsymbol{p})$ (Generator ciągu danych testowych):

1. Oblicz $f(x), f^2(x), \ldots, f^n(x)$ i zapisz je w pamięci;
2. Zatrzymaj się po n iteracjach i zwróć zapisane wektory;

gdzie $f^2(x) = f\big(f(x)\big)$, itd.

Po zaimplementowaniu algorytmu A1 z parametrem $n = 20$ otrzymano wynik:

X_1	-10,0000	-5,0000	3,0000	0,0000
X_2	-5,6667	-1,3333	2,6667	2,6667
X_3	-2,7778	1,1111	2,4444	4,4444
X_4	-0,8519	2,7407	2,2963	5,6296
X_5	0,4321	3,8272	2,1975	6,4198
X_6	1,2881	4,5514	2,1317	6,9465
X_7	1,8587	5,0343	2,0878	7,2977
X_8	2,2391	5,3562	2,0585	7,5318
X_9	2,4928	5,5708	2,0390	7,6879
X_{10}	2,6618	5,7139	2,0260	7,7919
X_{11}	2,7746	5,8092	2,0173	7,8613
X_{12}	2,8497	5,8728	2,0116	7,9075
X_{13}	2,8998	5,9152	2,0077	7,9383
X_{14}	2,9332	5,9435	2,0051	7,9589
X_{15}	2,9555	5,9623	2,0034	7,9726
X_{16}	2,9703	5,9749	2,0023	7,9817
X_{17}	2,9802	5,9833	2,0015	7,9878
X_{18}	2,9868	5,9888	2,0010	7,9919
X_{19}	2,9912	5,9926	2,0007	7,9946
X_{20}	2,9941	5,9950	2,0005	7,9964

Tabela 1. Wyniki działania algorytmu Banach_GEN po 20-stu iteracjach.

Analizując wyniki łatwo zauważyć, że algorytm bardzo szybko zbliża się do celu, jakim jest osiągnięcie punktu p. Takie podejście z pewnością znajdzie zastosowanie w takich problemach optymalizacji testów np. w analizie wartości brzegowych w urządzeniach, które są bardzo wrażliwe na dostarczane dane.

Testowanie czujników za pomocą twierdzenia Banacha o punkcie stałym

Zanim zostanie podany sposób testowania dowolnego czujnika, należy opisać ten czujnik pewnym modelem matematycznym. Załóżmy teraz, że istnieje czujnik z pewnym systemem informatycznym z funkcją g, która przyjmuje na wejściu wektor n - wymiarowy $x = [x_1, \dots, x_n] \subset R^n$ z metryką euklidesową. Funkcjonalność g zaczyna zwracać wynik jeżeli $x_1 \geq a_1, \dots, x_n \geq a_n$. W przeciwnym przypadku na wyjściu funkcji g nie pojawia się żadna wartość. Należy wygenerować ciągi danych wejściowych, które pozwolą sprawdzić, czy funkcjonalność nie zacznie zwracać wyniku jeśli na wejściu pojawią się wartości $x_1 < a_1$ lub $x_2 < a_2$ lub$\dots, x_n < a_n$. Z punktu widzenia optymalizacji testowania (doboru danych testowych) wystarczy wygenerować n ciągów, w których każdy będzie zbieżny od pewnego punktu posiadającego **na co najmniej jednej współrzędnej** wartość, która nie powoduje uruchomienia funkcjonalności g. Do rozwiązania postawionego problemu można również wykorzystać wniosek 1 z poprzedniego rozdziału, poprzez zastosowanie następujących funkcji, który wygenerują dane testowe:

$$f_1([x_1, \dots, x_n]) = \frac{1}{K}\big[(K-1)[x_1, \dots, x_n] + [a_1, 0, \dots, 0]\big], (6)$$
$$f_2([x_1, \dots, x_n]) = \frac{1}{K}\big[(K-1)[x_1, \dots, x_n] + [0, a_2, \dots, 0]\big], (7)$$
$$\dots$$
$$f_n([x_1, \dots, x_n]) = \frac{1}{K}\big[(K-1)[x_1, \dots, x_n] + [0, 0, \dots, a_n]\big]. (8)$$

Dla przykładu weźmy przestrzeń R^3. Niech dany będzie czujnik, który zaczyna zwracać wartość, jeśli $x_1 \geq 3$, $x_2 \geq 0$, $x_3 \geq -1$. Niech $K = 3$, funkcje generujące dane testowe rozpoczynają iteracje od punktu $[-5, -6, -4]$ będą dane wzorami:

$$f_1([x_1, x_2, x_3]) = \frac{1}{3}\big[2[x_1, x_2, x_3] + [3,0,0]\big],$$
$$f_2([x_1, x_2, x_3]) = \frac{1}{3}\big[2[x_1, x_2, x_3] + [4,0,0]\big],$$
$$f_3([x_1, x_2, x_3]) = \frac{1}{3}\big[2[x_1, x_2, x_3] + [4,1,-1]\big].$$

Niżej przedstawione tabele prezentują wyniki zwrócony przez zastosowanie algorytmu A1, dla wyżej opisanej sytuacji:

X_1	**-5,000**	-6,000	-4,000
X_2	**-2,333**	-4,000	-2,667
X_3	**-0,556**	-1,667	-1,778
X_4	**0,630**	-0,111	-1,185
X_5	**1,420**	0,926	-0,790
X_6	**1,947**	1,617	-0,527
X_7	**2,298**	2,078	-0,351

X_8	**2,532**	2,385	-0,234
X_9	**2,688**	2,590	-0,156
X_{10}	**2,792**	2,727	-0,104
X_{11}	**2,861**	2,818	-0,069
X_{12}	**2,908**	2,879	-0,046
X_{13}	**2,938**	2,919	-0,031
X_{14}	**2,959**	2,946	-0,021
X_{15}	**2,973**	2,964	-0,014
X_{16}	**2,982**	2,976	-0,009
X_{17}	**2,988**	2,984	-0,006
X_{18}	**2,992**	2,989	-0,004
X_{19}	**2,995**	2,993	-0,003
X_{20}	**2,996**	2,995	-0,002

Tabela 2. Wyniki działania algorytmu Banach_GEN po 20-stu iteracjach testujące podejście pod lewostronny brzeg pierwszej współrzędnej.

X_1	-5,000	**-6,000**	-4,000
X_2	-2,000	**-4,000**	-2,667
X_3	0,000	**-2,667**	-1,778
X_4	1,333	**-1,778**	-1,185
X_5	2,222	**-1,185**	-0,790
X_6	2,815	**-0,790**	-0,527
X_7	3,210	**-0,527**	-0,351
X_8	3,473	**-0,351**	-0,234
X_9	3,649	**-0,234**	-0,156
X_{10}	3,766	**-0,156**	-0,104
X_{11}	3,844	**-0,104**	-0,069
X_{12}	3,896	**-0,069**	-0,046
X_{13}	3,931	**-0,046**	-0,031
X_{14}	3,954	**-0,031**	-0,021
X_{15}	3,969	**-0,021**	-0,014
X_{16}	3,979	**-0,014**	-0,009
X_{17}	3,986	**-0,009**	-0,006
X_{18}	3,991	**-0,006**	-0,004
X_{19}	3,994	**-0,004**	-0,003
X_{20}	3,996	**-0,003**	-0,002

Tabela 3. Wyniki działania algorytmu Banach_GEN po 20-stu iteracjach testujące podejście pod lewostronny brzeg drugiej współrzędnej.

X_1	-5,000	-6,000	**-4,000**
X_2	-2,000	-3,667	**-3,000**
X_3	0,000	-2,111	**-2,333**
X_4	1,333	-1,074	**-1,889**
X_5	2,222	-0,383	**-1,593**
X_6	2,815	0,078	**-1,395**
X_7	3,210	0,385	**-1,263**
X_8	3,473	0,590	**-1,176**
X_9	3,649	0,727	**-1,117**
X_{10}	3,766	0,818	**-1,078**
X_{11}	3,844	0,879	**-1,052**
X_{12}	3,896	0,919	**-1,035**
X_{13}	3,931	0,946	**-1,023**
X_{14}	3,954	0,964	**-1,015**
X_{15}	3,969	0,976	**-1,010**
X_{16}	3,979	0,984	**-1,007**
X_{17}	3,986	0,989	**-1,005**
X_{18}	3,991	0,993	**-1,003**
X_{19}	3,994	0,995	**-1,002**
X_{20}	3,996	0,997	**-1,001**

Tabela 4. Wyniki działania algorytmu Banach_GEN po 20-stu iteracjach testujące podejście pod lewostronny brzeg trzeciej współrzędnej.

Idea wygenerowanych wyników jest taka, że tylko jedna współrzędna wektora wejściowego nie spełnia warunków aktywacji czujnika. Takie podejście nazywane jest testowaniem negatywnym i często jest to bardzo ważny rodzaj testów.

Podsumowanie

Głównym celem napisania artykułu było krótkie zaprezentowanie informacji na temat znanych powszechnie technik generowania danych testowych oraz zaproponowanie zastosowania wniosku, który wynika z twierdzenia Banacha o punkcie stałym w przestrzeni R^n. Podczas głębszej analizy artykułu czytelnik z pewnością zda sobie sprawę z tego, że testowanie jak i dobieranie danych testowych nie jest trywialną czynnością.

„Poprzeczka jakości" systemów informatycznych wzrasta z biegiem lat. Proces dobierania danych testowych, które spełniają wszystkie warunki, które opisane są w początkowym rozdziale pozwoli na oszczędność czasu oraz pozwoli potwierdzić jakość testowanego produktu. Strategie testowe są bardzo różne. Zgodnie z teorią No Free Lunch z pewnością znajdzie się klasa problemów, dla których opisane w artykule podejście generowania danych testowych znajdzie zastosowanie. Coraz częściej w środowisku testerskim dyskutuje się na temat automatyzacji testowania (nie jest to to samo oczywiście, co automatyzacja wykonania testu). Zaproponowane podejście może być postrzegane jak automatyzacja generowania ciągu danych testowych.

LITERATURA

1. S.G. Ahmed. *Automatic generation of basis test paths using variable length genetic algorithm,* Journal Information Processing Letters, Volume 114 Issue 6, June, 2014, pp. 304-316.
2. R. Alavi, S. Lofti. *The New Approach for Software Testing Using a Genetic Algorithm Based on Clustering Initial Test Instances*, International Conference on Computer and Software Modeling 2011, IPCSIT vol.14 (2011).
3. E. Alba, F. Chicano. *Observations in using parallel and sequential evolutionary algorithms for automatic software testing*, Computers & Operations Research, Vol. 35 Issue 10, October 2008, pp. 3163-3183.
4. Aleti, L. Grunske*. Test data generation with a Kalman filter-based adaptive genetic algorithm,* The Journal of Systems and Software Vol. 103, May 2015 pp. 343-352.
5. M. Alshraideh, B.A. Mahafzah, S. Al-Sharaeh. *A multiple-population genetic algorithm for branch coverage test data generation*, Software Quality Journal, Vol. 19, Volume 19, Issue 3 September 2011, pp. 489-513.
6. D. Farley, J. Humble. *Ciągłe dostarczanie oprogramowania*, Helion 2015.
7. D. Gong, T. Tian, X. Yao. *Grouping target paths for evolutionary generation of test data in paralel*, The Journal of Systems and Software, Vol. 85, Issue 11, November 2012, pp. 2531–2540.
8. D. Gong, Y. Zhang. *Generating test data for both path coverage and fault detection using genetic algorithms* , Frontiers of Computer Science, December 2013, Vol. 7, Issue 6, pp. 822-837.
9. D. Gong, Y. Zhang. *Generating test data for both path coverage and fault detection using genetic algorithms: multi-path case*, Frontiers of Computer Science , October 2014, Vol. 8, Issue 5, pp. 726-740.
10. M.J. Harrold, R.. Pargas, R. R. Peck. *Test-Data generation using genetics algorithms,* Journal of Software Testing, Verification and Reliability 1999.
11. Hermadi, C. Lokan, R. Sarker. *Dynamic stopping criteria for search-based test data generation for path testing* , Information and Software Technology, April 2014 Vol. 56, pp. 395-407.
12. J. Hudec, E. Gramatova. *An Efficient functional test generation method for processors using genetic algorithms*, Journal of Electrical Engineering, July 2015, Vol. 66, Issue. 4, pp. 186-193.
13. Y. Ho, D. L. Pepyne. *Simple Explantation of No Free Lunch Theorem of Optimization*.
14. Decision and Control,. Precedings of the 40th IEEE Conference, Dezember 2001, Vol. 5, pp. 4409-4414.
15. N. Khurana, R.S. Chillar. *Test Case Generation and Optimization using UML Models and Genetic Algorithm*, Procedia Computer Science, August 2015, Vol. 57, pp. 966-1004.
16. H. Kim, P.R. Srivastava. *Application of Genetic Algorithm in Software Testing*, International Journal of Software Engineering and Its Applications, October 2009, Vol. 3, Issue 4, pp. 87-96.
17. R. Krishnamoorthi, A. Sahaaya, S.A. Mary. *Regression Test Suite Prioritization using Genetic Algorithms,* International Journal of Hybrid Information Technology, July 2009 Vol.2, Issue .3, July.
18. C. Mao. *Harmony search-based test data generation for branch coverage in software structural testing*, Neural Computing and Applications, September 2013, Vol. 25, Issue 1, pp.199-216. Springer-Verlag London 2013.
19. M. Mirzaaghaei, F. Pastore, M. Pezze. *Automatic test case evolution,* Software testing, Verification and Reliability, April 2014, Vol. 24, Issue 5, pp.386-411.
20. H. Musielak, J. Musielak. *Analiza matematyuczna. Tom I, Część 1: Ciągi, szeregi i funkcje.* Wydawnictwo Naukowe Uniwersytetu Im. Adama Mickiewicza w Poznaniu, 1992.
    a. Piaskowy, R. Smilgin. *Dane Testowe, teoria i praktyka*, Helion 2011.
    b. Roman, *Testowanie i jakość oprogramowania*, PWN 2015.
21. D. Rutkowska, M. Piliński, L. Rutkowski. Sieci neuronowe, algorytmy genetyczne i systemy rozmyte, PWN W-wa 1997.

22. D. Warchoł, M. Żukowicz. *Testing education: test case prioritization using matrices,* General and Professional Education, May 2015, Vol. 1, Issue 8, pp. 57-62.
23. M. Żukowicz. *Edukacja testowania: Narzędzie All-pairs Testing w procesie optymalizacji testów konfiguracji – zastosowanie narzędzia w systemie B2B OPTIbud,* General and Professional Education, Dezember 2015, Vol. 4, Issue 12, pp. 99-106.
24. M. Żukowicz. *Edukacyjne i ekonomiczne aspekty zastosowania cyklu Hamiltona w projektowaniu i testowaniu oprogramowania*, General and Professional Education, numer Dezember 2014, Vol. 4, Issue 13, pp. 95-102.
25. M. Żukowicz, O pewnych problemach analizy wartości brzegowych, Internet:, http://testerzy.pl/materialy/index.php?file=analiza-wartosci-brzegowych.pdf
26. https://pl.wikipedia.org/wiki/Stefan_Banach

Autor

Marek Żukowicz jest absolwentem matematyki na Uniwersytecie Rzeszowskim. Obecnie pracuje jako tester. Jego zainteresowania skupiają się wokół testowania, matematyki, zastosowania algorytmów ewolucyjnych oraz zastosowania matematyki w procesie testowania. Interesuje się również muzyką, grą na akordeonach oraz na perkusji.