INŻYNIERIA OPROGRAMOWANIA W PROCESACH INTEGRACJI SYSTEMÓW INFORMATYCZNYCH
Pod redakcją J. Górskiego, C. Orłowskiego, 2010 PWNT Gdańsk

PREDYKCJA DEFEKTÓW NA PODSTAWIE METRYK OPROGRAMOWANIA – IDENTYFIKACJA KLAS PROJEKTÓW*

Marian Jureczko[1], Lech Madeyski[2]

1. Wprowadzanie

Istnieje szereg prac naukowych w których wykorzystując różne rodzaje regresji bądź też metody sztucznej inteligencji konstruuje się modele predykcji defektów bazujące na metrykach oprogramowania. Są to takie modele, które na podstawie metryk charakteryzujących wytwarzany w projekcie programistycznym artefakt (zazwyczaj klasę lub plik), dokonują oszacowania prawdopodobieństwa wystąpienia w tym artefakcie defektu albo liczby istniejących w tym artefakcie defektów. Modele takie są bardzo przydatne w procesie testowania, kiedy w wyniku ograniczeń budżetowych nie jesteśmy w stanie przetestować wszystkich wytworzonych artefaktów. Wtedy testując tylko te artefakty, które według modelu predykcji cechują się największą liczbą defektów, możemy zaoszczędzić sporo czasu, a mimo to zidentyfikować większość znajdujących się w systemie defektów. Model predykcji defektów usprawnia również refaktoryzację, przez wskazanie tych klas, w przypadku których refaktoryzując uzyskamy największe korzyści. Klasy, które mają wartości metryk będące według modelu zazwyczaj powiązane ze sporą liczbą defektów należy uznać za klasy posiadające niepoprawną strukturę. Przekształcając je do postaci, dla której model nie będzie sygnalizował dużej liczby defektów, dokonujemy zazwyczaj dobrej refaktoryzacji.

Korzyści wynikające ze stosowania modeli predykcji defektów nie budzą większej wątpliwości, a mimo to, jak wynika z obserwacji poczynionych przez autorów na lokalnym rynku informatycznym, poziom ich stosowania w przemyśle jest znikomy. Podstawową przyczyną tego stanu rzeczy są problemy ze stosowaniem modelu opracowanego na podstawie jednego projektu do predykcji w drugim projekcie, co jest podstawowym scenariuszem stosowania takiego modelu z punktu widzenia projektów przemysłowych. Istnieją nieliczne próby wielokrotnego wykorzystywania tego samego modelu w różnych projektach [14,16,17,18]. Zupełnie brak natomiast ogólnych wytycznych pozwalających stwierdzić kiedy jakiś model jest modelem odpowiednim do badania konkretnego projektu. Niniejsza praca próbuje wypełnić tą lukę. Przedstawiono tu eksperyment w ramach którego postawiono hipotezę, że istnieją trzy klasy projektów: projekty przemysłowe, projekty otwarte (open-source), oraz projekty akademickie, które na tyle się między sobą różnią, że do badania projektów do nich

*Praca częściowo wspierana przez stypendium współfinansowane przez Unię Europejską w ramach EFS.
[1]Politechnika Wrocławska, Wydział Elektroniki, Instytut Informatyki Automatyki i Robotyki, e-mail: marian.jureczko@pwr.wroc.pl
[2]Politechnika Wrocławska, Wydział Informatyki i Zarządzania, Instytut Informatyki, e-mail: lech.madeyski@pwr.wroc.pl

należących trzeba stosować zupełnie inne modele predykcji defektów. W celu zweryfikowania tej hipotezy skonstruowano trzy różne modele, które następnie stosowano do predykcji defektów w projektach zarówno należących do klasy na podstawie której dany model konstruowano jak i nie należących do tej klasy. Następnie oceniano dokładność predykcji poszczególnych modeli w stosunku do poszczególnych projektów. Na podstawie uzyskanych ocen zweryfikowano, czy zaproponowany w hipotezie podział na klasy projektów jest uzasadniony z punktu widzenia modeli predykcji defektów.

2. Tło literaturowe

Istnieje wiele prac naukowych dotyczących konstrukcji modeli predykcji defektów w oparciu o metryki oprogramowania, m.in. [1,4,2,5]. Temat ponownego wykorzystywania modelu pojawia się jednak stosunkowo rzadko, tym bardziej warto więc omówić te prace, które ten temat podejmują.

Watanabe i in. [16] podjęli próbę zastosowania modelu skonstruowanego na podstawie systemu rozwijanego w Javie do predykcji defektów w projekcie rozwijanym w C++ i odwrotnie. Badane przez nich projekty były bardzo podobne co do rozmiaru i dziedziny. Były to dwa edytory tekstu: *Sakura Editor* i *jEdit*. Na podstawie przeprowadzonego eksperymentu doszli do wniosku, że ponowne wykorzystanie modelu jest możliwe, ale mimo tego, że badane projekty mają podobny rozmiar i podobny zestaw funkcjonalności, to dokładność predykcji nadal pozostawia wiele do życzenia.

Weyuker i in. [17] opracowali generyczny model predykcji defektów, który można by stosować w wielu różnych projektach. Swoje badania opierają jednak na zaledwie jednym projekcie (mającym za to aż 35 wydań), a uzyskanych modeli nie walidują przy pomocy innych projektów programistycznych. W związku z tym uzyskane wyniki (znajdowano od 50% do 92% defektów w 20% plików wskazanych przez model jako obarczonych największą liczbą defektów) nie mogą zostać uznane za zakończone sukcesem tj. opracowaniem generycznego modelu predykcji defektów. Rozwinięcie tych badań można znaleźć w [18]. Zastosowano tam wcześniej opracowane modele w 3 innych projektach uzyskując niemal równie dobre wyniki jak we wspomnianej wcześniej pracy. Autorzy jednak sami przyznają, że opracowany przez nich model nie jest uniwersalny. Nadaje się do stosowania w tylko jednej klasie projektów – dużych projektach przemysłowych. Autorzy nie weryfikowali skuteczności działania modelu w innych typach projektów.

Warto jeszcze wspomnieć o pracy [14], gdzie autorzy na podstawie pięciu projektów przemysłowych badali możliwość ponownego wykorzystywania modeli predykcji defektów. Na podstawie otrzymanych wyników stwierdzili, że ponowne wykorzystywanie jest możliwe jedynie w przypadku kolejnych wersji tego samego projektu, lub w przypadku podobnych projektów. Nie podano jednak ani formalnej definicji podobieństwa ani nie dowodzono istotności wyciągniętych wniosków.

3. Definicja eksperymentu

3.1. Akwizycja danych

W omawianym eksperymencie analizowano 5 projektów przemysłowych (w 24 wersjach), 14 projektów otwartych (w 40 wersjach) oraz 17 projektów studenckich (w 17 wersjach). Liczba wersji jest większa od liczby projektów dlatego, że dla niektórych badanych projektów analizowano więcej niż jedną wersję. Np. przebadano wersje 1.3, 1.4, 1.5, 1.6 oraz 1.7 projektu *Ant* organizacji Apache. Kolejne wersje są zawsze widocznymi dla klienta wydaniami projektu. Wszystkie badane projekty przemysłowe to zaawansowane systemy tworzone pod indywidualne

zamówienie klienta. Wszystkie zostały już wdrożone i są z powodzeniem stosowane przez klienta dla którego zostały wykonane. Badane projekty otwarte to: *Ant*, *Camel*, *ckjm*, *Forrest*, *ivy*, *log4j*, *Lucene*, *pBeans*, *POI*, *Synapse*, *Tomcat*, *Velocity*, *Xalan* oraz *Xerces*. Projekty studenckie były realizowane przez 4 lub 5 osobowe zespoły złożone ze studentów 4 i 5 roku jednolitych magisterskich studiów na kierunku informatyka. Tematyka projektów studenckich była różnorodna. Wszystkie badane projekty były implementowane w języku Java.

Dla każdej klasy wyliczano następujące metryki:

- zaproponowane przez Chidambera i Kemerera [4]: Weighted Methods per Class (WMC), Depth of Inheritance Tree (DIT), Number Of Children (NOC), Coupling Between Object classes (CBO), Response For a Class (RFC), Lack of Cohesion in Methods (LCOM),
- zaproponowaną przez Hendersona-Sellersa [7]: Lack of Cohesion in Methods (LCOM3),
- zaproponowane przez Martina [12]: Afferent Couplings (CA), Efferent Couplings (CE),
- zaproponowane przez Bansiya i Davisa [1]: Number of Public Methods (NPM), Data Access Metric (DAM), Measure Of Aggregation (MOA), Measure of Functional Abstraction (MFA), Cohesion Among Methods of class (CAM),
- zaproponowane przez Tanga [15]: Inheritance Coupling (IC), Coupling Between Methods (CBM), Average Method Complexity (AMC),
- opierające się na złożoności cyklomatycznej McCabe [13]: średnia (Avg_CC) oraz maksymalna (Max_CC) złożoność cyklomatyczna dla wszystkich metod klasy,
- oraz liczba linii kodu w klasie (LOC)

Definicje metryk, ze względu na ograniczony rozmiar niniejszej pracy, można znaleźć w [10].

3.2. Badane hipotezy

Aby zweryfikować, czy z punktu widzenia modeli predykcji defektów sensowne jest wprowadzenie klas projektów postawiono hipotezę, że istnieją trzy takie klasy: projekty przemysłowe, otwarte i studenckie. Hipotezę weryfikowano konstruując modele w oparciu o poszczególne klasy projektów. Przy pomocy tych modeli dokonywano predykcji defektów w poszczególnych wersjach projektów. Jakość uzyskanej predykcji pozwalała ocenić stopień dopasowania danego modelu do danej wersji projektu.

Niech funkcja E(*m*,*v*) będzie funkcją oceniającą stopień dopasowania modelu *m* do wersji projektu *v*. Funkcja ta wylicza predykcję liczby defektów dla każdej klasy należącej do wersji projektu *v* przy pomocy modelu *m*. Następnie szereguje klasy malejąco według uzyskanego oszacowania i sumuje liczbę rzeczywiście znalezionych błędów (rzeczywista liczba znalezionych błędów jest znana dla każdej wersji każdego projektu) w kolejnych klasach należących do *v* z zachowaniem wspomnianego wcześniej uszeregowania. Sumowanie zostaje przerwane wtedy, kiedy liczba zsumowanych rzeczywistych defektów przekroczy 80% liczby wszystkich rzeczywistych defektów w *v*. Oceną stopnia dopasowania modelu jest procent klas należących do *v*, które zostały poddane przeglądowi przed dotarciem do wspomnianej wcześniej granicy 80%. Im lepszy stopień dopasowania tym mniejszy jest uzyskany wynik. Innymi słowy funkcja ta wylicza jaki procent klas musi zostać poddany inspekcji, pod warunkiem przeglądu klas w kolejności zasugerowanej przez model *m*, aby znaleźć 80% defektów znajdujących się w badanej wersji projektu.

Model to kombinacja liniowa wartości metryk uzupełniona o wyraz wolny. Model budowany jest poprzez zastosowanie regresji krokowej postępującej. Zmiennymi niezależnymi w regresji są wartości metryk dla danej klasy (brano zawsze pod uwagę wszystkie klasy wchodzące w skład wszystkich wersji projektów na podstawie których budowano model), a zmienną zależną rzeczywista liczba defektów w tej klasie. Przed zastosowaniem regresji eliminowano te metryki, które były silnie ze sobą skorelowane, odrzucając zawsze tą która była słabiej skorelowana z rzeczywistą liczbą defektów. Dzięki zastosowaniu regresji krokowej do modelu trafiały tylko

te metryki, które niosły istotną informację o defektach. Przy okazji powodowało to, że różne modele wykorzystywały różne metryki. Co więcej, liczba wykorzystanych metryk również nie była stała dla wszystkich modeli.

Badane jest istnienie trzech klas projektów. W związku z tym eksperyment musi zostać powtórzony trzy razy. Aby móc za każdym razem stosować tą samą definicję eksperymentu przyjmijmy, że A, B i C to rozważane klasy projektów. Na potrzeby eksperymentu możemy przyjąć, że klasa projektów jest tożsama ze zbiorem wersji projektów. Zatem niech A będzie zbiorem wszystkich wersji projektów zawierających się w klasie, której istnienie badamy. Niech *a* będzie wersją projektu należącą do A. Niech m_X będzie modelem skonstruowanym na podstawie wszystkich wersji projektów należących do zbioru X. W szczególności istnieją m_A, m_B, m_C, $m_{B\cup C}$ ($B\cup C = \neg A$). Następnie dla każdego *a* wyliczono:

- $E(m_A,a)$ - zbiór uzyskanych wartości nazwijmy E_A,
- $E(m_B,a)$ - zbiór uzyskanych wartości nazwijmy E_B,
- $E(m_C,a)$ - zbiór uzyskanych wartości nazwijmy E_C,
- $E(m_{B\cup C},a)$ - zbiór uzyskanych wartości nazwijmy $E_{B\cup C}$.

Mając tak zdefiniowane zbiory można postawić następujące hipotezy zerowe:

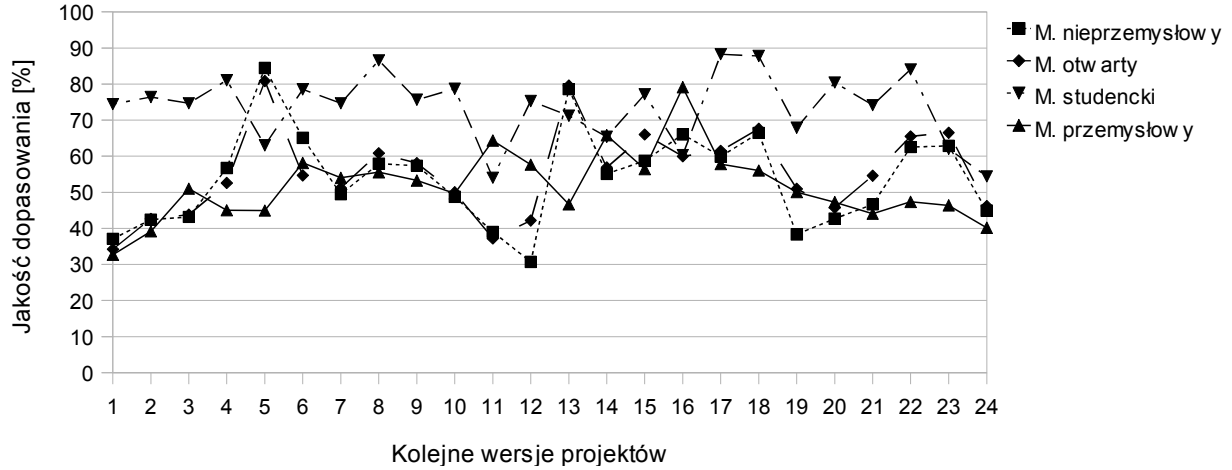
- $H_{0,A,B}$ – E_A oraz E_B pochodzą z tego samego rozkładu. Odrzucenie tej hipotezy będzie oznaczało, że E_A i E_B pochodzą z różnych rozkładów. Więc jeżeli dodatkowo średnia z E_A będzie mniejsza od średniej z E_B, to będzie to oznaczało, że stosowanie modelu zbudowanego na podstawie wersji projektów należących do B do dokonywania predykcji liczby defektów w wersji projektu należącej do A daje statystycznie gorsze rezultaty niż stosowanie modelu zbudowanego na podstawie wersji projektów należących do A.
- $H_{0,A,C}$ – E_A oraz E_C pochodzą z tego samego rozkładu.
- $H_{0,A,B\cup C}$ – E_A oraz $E_{B\cup C}$ pochodzą z tego samego rozkładu.

Z uwagi na to, że dla każdego elementu należącego do E_A można znaleźć dokładnie jeden odpowiadający mu element w każdym z pozostałych zbiorów (E_B, E_C, $E_{B\cup C}$) do weryfikowania hipotez można użyć testu dla prób zależnych. Przed testowaniem hipotezy sprawdzono normalność rozkładu przy pomocy testu Shapiro-Wilka oraz homogeniczność wariancji przy pomocy testu Levene'a [11]. Testy te pokazały, że do testowania każdej z hipotez można było zastosować test t dla prób zależnych.

4. Wyniki Eksperymentu

Eksperyment przeprowadzono trzy razy. Za każdym razem badano istnienie innej klasy projektów, kolejno były to projekty przemysłowe, projekty otwarte i projekty studenckie.

4.1. Klasa projektów przemysłowych



Rys. 1. Dopasowanie modeli do poszczególnych projektów

Tab. 1. Statystyki opisowe – dopasowanie modeli do projektów przemysłowych

	Model nieprzemysłowy	Model otwarty	Model studencki	Modele przemysłowe
Średnia	53,96	55,38	73,59	51,77
Odchyl. stand.	13,29	12,01	9,68	9,76

Tab. 2. Badanie normalności rozkładu – test Shapiro-Wilka

	Model nieprzemysłowy	Model otwarty	Model studencki	Modele przemysłowe
W	0,970 (p=0,650)	0,975 (p=0,794)	0,946 (p=0,220)	0,960 (p=0,443)

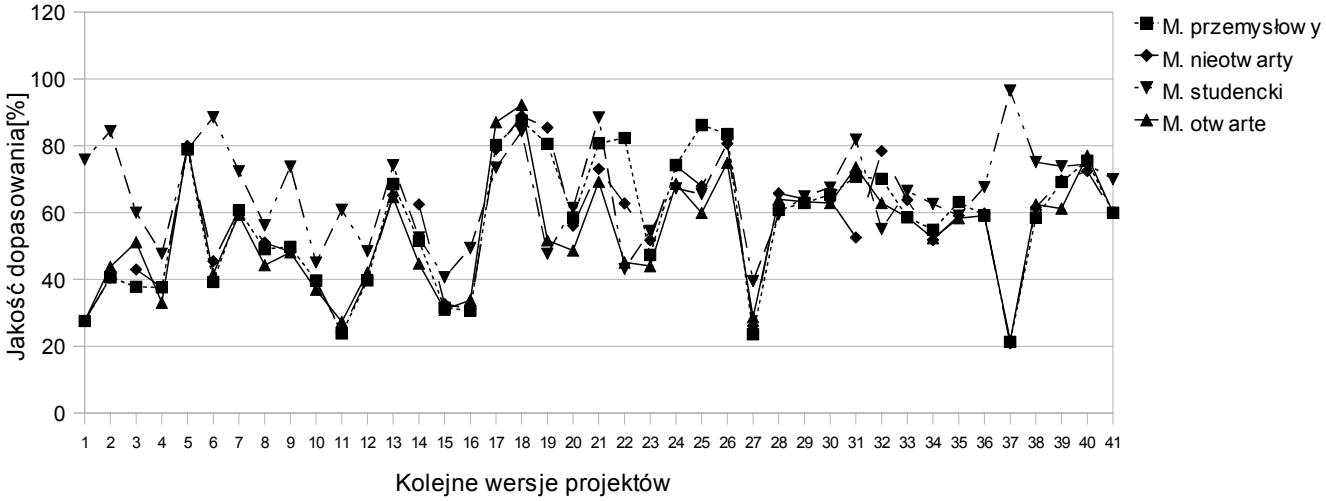
Tab. 3. Badanie homogeniczności wariancji pomiędzy wynikami stosowania modelu przemysłowego, a pozostałymi – test Levene'a

	Model nieprzemysłowy	Model otwarty	Model studencki
F(1,df), df=46	3,149	1,207	0,007
Prawd. tesowe	0,083	0,278	0,936

Tab. 4. Testowanie hipotez – test t dla prób zależnych

	$H_{0,przem,nprzem}$	$H_{0,przem,otw}$	$H_{0,przem,stud}$
t, df=23	0,708	1,223	7,320
Prawd. tesowe	0,486	0,234	**0,0000**

4.2. Klasa projektów otwartych



Rys. 2. Dopasowanie modeli do poszczególnych projektów

Tab. 5. Statystyki opisowe – dopasowanie modeli do projektów otwartych

	Model przemysłowy	Model nieotwarty	Model studencki	Modele otwarte
Średnia	57,11	56,57	65,59	54,08
Odchyl. stand.	19,12	17,67	14,22	16,85

Tab. 6. Badanie normalności rozkładu – test Shapiro-Wilka

	Model przemysłowy	Model nieotwarty	Model studencki	Modele otwarte
W	0,956 (p=0,116)	0,971 (p=0,380)	0,983 (p=0,768)	0,982 (p=0,734)

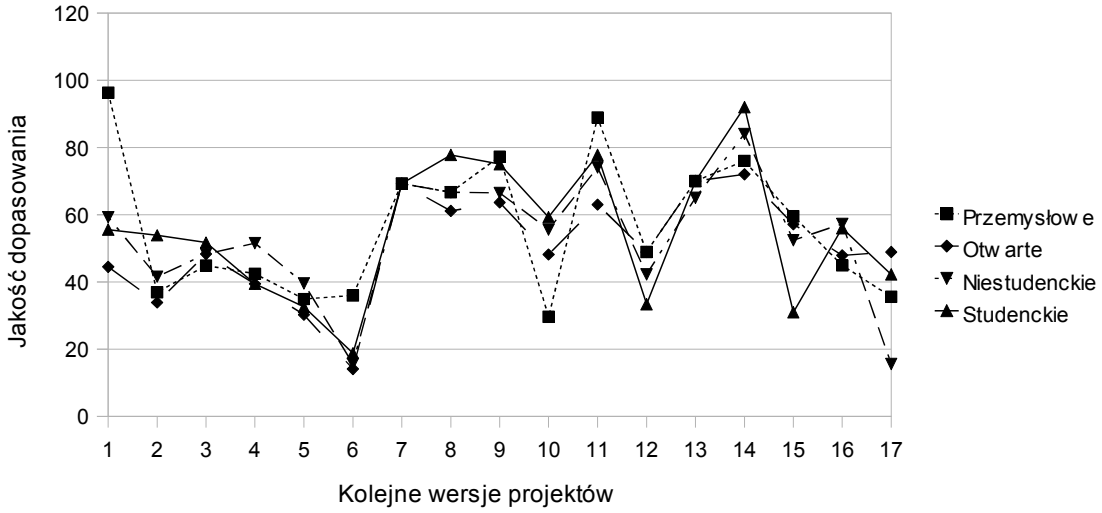
Tab. 7. Badanie homogeniczności wariancji pomiędzy wynikami stosowania modelu otwartego, a pozostałymi – test Levene'a

	Model przemysłowy	Model nieotwarty	Model studencki
F(1,df), df=80	0,858	0,093	1,338
Prawd. tesowe	0,357	0,761	0,251

Tab. 8. Testowanie hipotez – test t dla prób zależnych

	$H_{0,otw,przem}$	$H_{0,otw,notw}$	$H_{0,otw,stud}$
t, df=40	2,091	1,907	4,405
Prawd. tesowe	**0,043**	0,064	**0,0001**

4.3. Klasa projektów studenckich



Rys. 3. Dopasowanie modeli do pszczególnych projektów

Tab. 9. Statystyki opisowe – dopasowanie modeli do projektów studenckich

	Model przemysłowy	Model otwarty	Model niestudencki	Modele studenckie
Średnia	56,34	50,6	53,19	55,02
Odchyl. stand.	20,71	15,56	18,54	20,21

Tab. 10. Badanie normalności rozkładu – test Shapiro-Wilka

	Model przemysłowy	Model otwarty	Model niestudencki	Modele studenckie
W	0,921 (p=0,151)	0,946 (p=0,401)	0,942 (p=0,345)	0,973 (p=0,869)

Tab. 11. Badanie homogeniczności wariancji pomiędzy wynikami stosowania modelu studenckiego, a pozostałymi – test Levene'a

	Model przemysłowy	Model otwarty	Model niestudencki
F(1,df), df=32	0,271	1,343	0,303
Prawd. tesowe	0,606	0,255	0,586

Tab. 12. Testowanie hipotez – test t dla prób zależnych

	$H_{0,stud,przem}$	$H_{0,stud,otw}$	$H_{0,stud,nstud}$
t, df=16	0,312	-1,484	-0,696
Prawd. tesowe	0,759	0,157	0,496

Aby uniknąć wpływu wielokrotnego testowania na wynik testu, część statystyków zaleca stosowanie korekcji Bonferroniego, ewentualnie skorzystanie z nieco bardziej wyrafinowanych metod gdy jest to pomocne, by nie utracić istotnego rezultatu. Korekcja Bonferroniego zakłada, że tylko hipotezy dla których $p \leq \alpha_{Eksp}/n$ (gdzie $\alpha_{Eksp} = 0{,}05$ to poziom ufności dla całego eksperymentu, n to liczba porównań) są odrzucane. W naszym przypadku liczba porównań wynosi n = 9. Stąd poziom ufności dla każdego testu wynosi 0,0055. Konsekwencja zastosowania korekcji Bonferroniego jest taka, że nie możemy odrzucić hipotezy $H_{0,\ otw,przem}$.

5. Wnioski i plany dalszych badań

Uzyskane wyniki nie pozwalają na wyciągnięcie całkowicie jednoznacznych wniosków. W stosunku do żadnej z badanych klas projektów nie udało się odrzucić wszystkich trzech badanych hipotez. W związku z tym nie udało się formalnie wykazać istnienia żadnej z trzech klas projektów. Mimo to przeprowadzone eksperymenty ujawniły szereg ciekawych zależności o których warto wspomnieć. Okazało się że modele konstruowane na podstawie projektów studenckich były gorsze (różnica była statystycznie istotna) od prawie wszystkich pozostałych modeli podczas dokonywania predykcji dla projektów zarówno otwartych (zgodnie z tab. 5 średnie dopasowanie modeli studenckich to 65,59, a pozostałe od 54,08 do 57,11) jak i przemysłowych (zgodnie z tab. 1 studenckie 73,59, a pozostałe od 51,77 do 55,38). Co więcej, okazało się że model skonstruowany na podstawie projektów studenckich nie był lepszy niż inne modele podczas dokonywania predykcji dla projektów studenckich (tab. 9), czyli tych na podstawie których był utworzony. Można stąd wyciągnąć wniosek, że istnieje klasa projektów o charakterystyce bliskiej projektom studenckim, ale nie tak rozumiana jak to określono w tej pracy, a raczej jako klasa projektów obciążonych bardzo dużą losowością (wynikającą prawdopodobnie z niedojrzałości procesu wytwarzania oprogramowania i braku doświadczenia u programistów) i przez to nie nadającą się ani do badania przy pomocy modeli predykcji defektów, ani do konstruowania takich modeli.

Warto również zauważyć, że gdyby w eksperymencie przyjąć odrobinę mniej rygorystyczny poziom istotności, np. niekiedy przyjmowany $\alpha=0{,}10$ lub nawet tylko $\alpha=0{,}07$, to w przypadku klasy projektów otwartych wszystkie badane hipotezy zerowe zostałyby odrzucone (zgodnie z tab. 8 uzyskano następujące prawdopodobieństwa testowe: dla $H_{0,otw,przem}$ 0,043, dla $H_{0,otw,stud}$ 0,0001, a dla $H_{0,otw,nieotw}$ 0,064). Zastosowanie korekcji Bonferroniego oddala uzyskane wyniki od granicy za którą hipotezy zerowe zostałyby odrzucone, ale mimo to, uzyskane wyniki sugerują, że istnieje klasa projektów zbliżona do badanej tu klasy projektów otwartych.

Warto dodatkowo zwrócić uwagę na to (rys. 2), iż istnieje kilka projektów otwartych dla których stosowanie modelu skonstruowanego na podstawie projektów otwartych daje zdecydowanie gorsze wyniki. Istnieją więc wewnątrz badanej klasy projekty, które znacznie odbiegają od średniej. Jest więc bardzo prawdopodobne, że przedefiniowanie tej klasy przez odrzucenie najbardziej nietypowych projektów (tj. najbardziej odbiegających od średniej), pozwoliłoby na uzyskanie klasy, której istnienie można by udowodnić. Podobna sytuacja, ale na mniejszą skalę występuje w przypadku projektów przemysłowych (rys. 1). W związku z tymi obserwacjami planuje się przeprowadzenie dalszych badań które będą miały na celu wykrycie klas projektów w zgromadzonych danych na podstawie istniejących zależności pomiędzy wartościami metryk a defektami. Do tego celu planowane jest zastosowanie technik sztucznej inteligencji. Następnie, jeżeli uda się takie klasy zidentyfikować, to zostanie przeprowadzony dowód ich istnienia w sposób analogiczny jak w tej pracy.

W badaniach dotyczących modeli predykcji defektów w oprogramowaniu ważne jest aby udostępnić wyliczone wartości metryk tak, aby umożliwić replikację eksperymentu [6]. Prace nad serwisem w ramach którego zostaną udostępnione wartości metryk i informacje o defektach są już bardzo zaawansowane i niebawem będzie on dostępny pod adresem: http://purl.org/MarianJureczko/MetricsRepo. Warto tu również wspomnieć o planach dalszego

rozwoju narzędzi użytych do wyliczenia metryk (ckjm) oraz akwizycji danych o defektach (BugInfo). Narzędzie ckjm zostało rozbudowane i doczekało się już oficjalnego wydania, które dostępne jest pod adresem: http://gromit.iiar.pwr.wroc.pl/p_inf/ckjm. Dalszy rozwój tego narzędzia będzie się skupiał na usuwaniu ewentualnych defektów. Narzędzie BugInfo natomiast jest jeszcze cały czas intensywnie rozwijane, niemniej jego aktualną wersję można pobrać z: http://kenai.com/projects/buginfo.

6. Literatura

[1] Bansiya J. and Davis C. G.: A Hierarchical Model for Object-Oriented Design Quality Assessment. *IEEE Transactions on Software Engineering*, 28(1): 4-17, 2002.
[2] Basili V., Briand L., Melo W., 1996. A validation of object-oriented design metrics as quality indicators. *IEEE Transactions on Software Engineering* 22 (10), 751-761.
[3] Briand L. C., Daly J. W., Wust J.K: A unified framework for coupling measurement in object-oriented systems. *IEEE Transactions on Software Engineering* 25 (1), 91-121, 1999.
[4] Chidamber S. R. and Kemerer C. F.: A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, 20(6): 476–493, 1994.
[5] Fenton N. E. and Neil M.: A Critique of Software Defect Prediction Models. *IEEE Transactions on Software Engineering*, 25(5): 675-689, 1999.
[6] Forrest J. Shull F. J., Carver J. C., Vegas S., Juristo N.: The role of replications in Empirical Software Engineering. *Empirical Software Engineering* 13(2): 211-218, 2008.
[7] Henderson-Sellers B.: *Object-Oriented Metrics, measures of Complexity.* Prentice Hall, 1996.
[8] Jureczko M.: Use of software metrics for finding weak points of object oriented projects. *Proceedings of Metody i narzędzia wytwarzania oprogramowania* 133-144, 2007
[9] Jureczko M.: Ocena jakości obiektowo zorientowanego projektu programistycznego na podstawie metryk oprogramowania. *Inżynieria oprogramowania - metody wytwarzania i wybrane zastosowania*, PWN, 364-377, 2008
[10] Jureczko M., Spinellis D.: Using Object-Oriented Design Metrics to Predict Software Defects. *Models and methodology of system dependability.* Oficyna wydawnicza Politechniki Wrocławskiej, 69-81, 2010.
[11] Madeyski L.: Test-Driven Development - An Empirical Evaluation of Agile Practice. Springer, 2010.
[12] Martin R.: OO Design Quality Metrics - An Analysis of Dependencies. *Proceedings of Workshop Pragmatic and Theoretical Directions in Object-Oriented Software Metrics, OOPSLA'94*, 1994.
[13] McCabe T. J.: A complexity measure. *IEEE Transaction on Software Engineering*, 2(4): 308-320, 1976.
[14] Nagappan N., Ball T., Zeller A.: Mining Metrics to Predict Component Failures. *Proceedings of the 28th international conference on Software engineering*. 452-461, 2006.
[15] Tang M-H, Kao M-H and Chen M-H: An Empirical Study on Object-Oriented Metrics. *Proceedings of The Software Metrics Symposium* 242-249, 1999.
[16] Watanabe S., Kaiya H., Kaijiri K.: Adapting a Fault Prediction Model to Allow Inter Language Reuse. *Proc. of PROMISE'08*, 19-24, 2008.
[17] Weyuker E. J., Ostrand T. J., Bell R. M. : Adapting a Fault Prediction Model to Allow Widespread Usage. *Proceedings of PROMISE'06*, 1-5 2006.
[18] Weyuker E. J., Ostrand T. J., Bell R. M.: Do too many cooks spoil the broth? Using the number of developers to enhance defect prediction models. *Empirical Software Engineering* 13(5): 539-559, 2008.