XXIII Seminarium

ZASTOSOWANIE KOMPUTERÓW W NAUCE I TECHNICE' 2013
Oddział Gdański PTETiS
Referat nr 19

OCENA WKŁADU PROGRAMISTY W KOŃCOWY KOD ŹRÓDŁOWY PROGRAMU

Jerzy KACZMAREK[1], Michał WRÓBEL[2]

1. Wydział Elektroniki, Telekomunikacji i Informatyki, Politechnika Gdańska
   tel: (58) 347 26 82 fax: (58) 347 27 27 e-mail: jkacz@eti.pg.gda.pl
2. Wydział Elektroniki, Telekomunikacji i Informatyki, Politechnika Gdańska
   tel: (58) 347 29 89 fax: (58) 347 27 27 e-mail: wrobel@eti.pg.gda.pl

Streszczenie: Ocena pracy programistów jest zadaniem złożonym. Kierownicy projektów biorą pod uwagę takie czynniki jak jakość tworzonego kodu, zdolność do rozwiązywania problemów technicznych i biznesowych oraz produktywność. Mierzenie produktywności poszczególnych programistów jest jednak zadaniem skomplikowanym. W artykule zostanie przedstawiona nowa metoda oceny wkładu programisty w końcowy kod źródłowy programu. Na podstawie przechowywanej w repozytorium historii wykonanych operacji i zdefiniowanych w metodzie wag określany jest udział poszczególnych programistów w wytworzeniu oprogramowania. W artykule zostaną również przedstawione wyniki walidacji metody przeprowadzone dla dwóch projektów Open Source.

Słowa kluczowe: szacowanie nakładu, repozytorium kodu

1. WSTĘP

Obecnie coraz większa część ludzi, zwłaszcza w krajach rozwiniętych, pracuje umysłowo. W Polsce w 2010 w ten sposób pracowało już ponad 40% ludzi aktywnych zawodowo [1]. W odróżnieniu od pracy fizycznej, gdzie jej efekty można ocenić w sposób jednoznaczny, ocena nakładu pracy, wydajności, czy zaangażowania pracownika umysłowego jest znacznie bardziej skomplikowana. Programiści bez wątpienia są pracownikami umysłowymi, a wytwarzany przez nich kod źródłowy programów może być traktowany jako wirtualny produkt.

W inżynierii oprogramowania na przestrzeni lat zostało opracowanych wiele metod pozwalających na szacowanie nakładów koniecznych do wytworzenia oprogramowania, takich jak metoda FPA (*Function Point Analysis*) [2], COCOMO (*Constructive Cost Model*) i ich pochodne [3]. Jako podstawę wyliczeń przyjmują one ilość wierszy kodu, głównych wymagań funkcjonalnych, czy ilości klas w programowaniu obiektowym [4].

Powstały również metody pozwalające na ocenę wysiłku poszczególnych programistów i ich wkładu w produkt końcowy. Najprostsze podejście zakłada raportowanie wykonanych prac przez każdego programistę. Podobnie jak subiektywna ocena kierownika projektu, taka metoda jest mało miarodajna, gdyż w prosty sposób wyniki mogą być niewiarygodne. Jednak pomimo swych wad znajduje zastosowanie, i stanowi np. integralną część metody *Personal Software Process* [5].

Podejście hybrydowe, pozwala na zautomatyzowanie procesu raportowania postępów prac, poprzez integrację z narzędziami programistycznymi. W zdefiniowanych momentach pracy, np. przy kompilacji kodu źródłowego, czy wysłaniu kodu do repozytorium odpowiednie narzędzia pobierają od programisty informacje o czasie poświęconym na realizację zadanie, napotkane trudności, czy nakład wykonanej pracy. Jednak metoda taka nadal opiera się na wiarygodności ocenianego programisty, który może nierzetelnie udzielać odpowiedzi. Dodatkowo wymuszane przerwy w pracy przeznaczone na uzupełnianie raportu może rozpraszać, a nawet frustrować pracownika [6].

Najbardziej obiektywne informacje można uzyskać za pomocą metod automatycznych. Pozwalają one na zbieranie danych w sposób niezależny od programistów. Wśród metoda automatycznych mierzenie produktywności programistów można wyróżnić dwie grupy: analityczne i statystyczne. Narzędzie z pierwszej grupy pozwalają na analizowanie zachowania programisty w trakcie pracy. Najczęściej poprzez integrację ze środowiskiem programistycznym (*IDE*, ang. *Integrated Development Envirnoment*), czy nawet systemem operacyjnym, możliwe jest gromadzenie danych o czasie poświęconym przez programistę na realizację poszczególnych fragmentów kodu, na zapoznawanie się z dokumentacją lub jej tworzeniem, czy projektowanie systemu.

Metody statystyczne działają natomiast na bazie wytworzonych przez programistów danych. W głównej mierze są to kody źródłowe, dokumentacje i inne dokumenty projektowe. Analiza takich danych pozwala na określenie wkładu danego programisty w produkt końcowy.

W artykule zostanie przedstawiona nowa metoda mierzenia wkładu poszczególnych programistów w końcowy kod źródłowy programu. Ocena jest automatycznie wyliczana na podstawie informacji przechowywanych w repozytoriach kodu. Brane są pod uwagę operacje dodawania, modyfikowania i usuwania wierszy kodu. W celu odzwierciedlenia rzeczywistej pracochłonności dla poszczególnych operacji zostały zdefiniowane odpowiednie wagi, zależne od typu operacji

2. BADANIA WKŁADU PROGRAMISTY

Pojęcie wkładu programisty w produkt nie jest pojęciem ściśle zdefiniowanym w dziedzinie Inżynierii Oprogramowania. W języku polskim często jest mylone z pojęciem nakłady pracy. W języku angielskim, który jest podstawowym językiem informatyki, nakład pracy tłumaczony jest jako *effort*, natomiast wkład w produkt jako *contribution*. Nakład pracy rozumiany jest jako całość wykonanych prac, które w sposób bezpośredni i pośredni przyczyniły się do powstania produktu. W tym przypadku brane pod uwagę powinny być wszystkie wykonane prace zarówno organizacyjne, projektowe, jak również praca poświęcona nauce i rozwiązywaniu napotkanych problemów.

Natomiast na potrzeby niniejszego artykułu wkład programisty w końcowy kod źródłowy programu zdefiniowany będzie jako udział operacji dodawania, modyfikowania i usuwania wierszy kodu wykonanych przez poszczególnych programistów w stosunku do wszystkich operacji dokonanych w projekcie przez wszystkich programistów.

2.1. Systemy kontroli wersji

W celu zautomatyzowania procesu obliczania wkładu programisty zostaną wykorzystane informacje pobrane z systemów kontroli wersji (ang. *Software versioning and revision control system*). Podczas tworzenie programów komputerowych w większych zespołach roboczych konieczne jest zastosowanie między innymi narzędzi wspomagających wymianę kodów źródłowych. W praktyce zdecydowana większość projektów informatycznych korzysta z systemów kontroli wersji. Pozwalają one nie tylko na współdzielenie kodu pomiędzy programistami, ale przede wszystkim na przechowywanie szczegółowej historii zmian wprowadzonych w kodzie. Dzięki temu możliwe jest odtworzenie dowolnej wersji programu. Dodatkowo przetrzymywane są informacje o historii każdego pliku, czy strukturze projektu. Na podstawie zgromadzonych w ten sposób danych można uzyskać informację, który programista wprowadził zmiany w pliku, na czym te zmiany polegały i kiedy zostały przeprowadzone. Taki zbiór informacji wraz z kodem źródłowym i oprogramowaniem do zarządzania nosi nazwę repozytorium kodu [7].

Podczas pracy nad programem z użyciem systemu kontroli wersji, programista pracuje na swojej lokalnej kopii kodu źródłowego. W momencie zakończenia pracy nad fragmentem kodu, klasą czy modułem programista zgłasza zmiany i wysyła je do repozytorium. Taka operacja nosi angielską nazwę *commit*. Zatwierdzona zmiana, która zostaje włączona do kodu źródłowego nazywana jest rewizją.

W repozytorium przetrzymywane są między innymi informacje o tym kto wprowadził poszczególne zmiany. Pozwala to np. na identyfikację osób odpowiedzialnych za wytworzenie poszczególnych modułów, klas, czy funkcji. Poziom szczegółowości przetrzymywanych informacji pozwala nawet na znalezienie pojedynczych wierszy kodu wstawionych lub zmodyfikowanych przez konkretnego programistę. Korzystając z danych zawartych w repozytorium kodu możliwe jest zatem określenie jaki wkład mają poszczególni programiści w końcowy produkt.

2.2. Możliwości oceny wkładu w projektach

Przyjęta wąska definicja wkładu programisty w końcowy kod źródłowy pozwala na automatyczne generowanie metryki tylko na podstawie repozytorium programu. Z drugiej jednak strony należy mieć na uwadze, że uzyskane wyniki nie mogą być podstawą do oceny przydatności pracowników. Ocena taka odzwierciedla jedynie część prac potrzebnych do wytworzenia oprogramowania.

Szczególnie duża różnica pomiędzy nakładem pracy i wkładem w produkt będzie występowała w obecnie popularnych metodykach zwinnych. W podejściu tradycyjnym, ścisły podział ról na analityków, projektantów, programistów i testerów pozwala na przeprowadzenie oceny każdej grupy w oddzielny sposób. W takich metodykach pomiar wkładu tylko na podstawie analizy kodu źródłowego może dać rzeczywisty pogląd na nakład prac programisty poświęcony na wytworzenie programu. Jednak w podejściu zwinnym nie ma tak ścisłego podziału ról. W jednej z najpopularniejszych metodyk tego typu – metodzie *Scrum*, wśród osób tworzących kod źródłowych nie wyróżnia się żadnych ról – wszyscy programiści tworzą jeden *Zespół Scrum*. W związku z tym poza kodowaniem zajmują się również analizą, projektowaniem, tworzeniem dokumentacji i testowaniem. Dlatego nie może być podstawą oceny przydatności członka zespołu tylko udział jego kodu w gotowym programie.

Inna popularna metodyka lekka – *eXtreme Programming* wprowadza tzw. programowanie w parach. Polega to na tym, że nad jednym fragmentem kodu, w jednym czasie, przy jednym komputerze pracuje jednocześnie dwóch programistów, jeden pisze a drugi mu pomaga. Pomimo, iż takie podejście wydaje się na pierwszy rzut oka mało efektywne, badania pokazuję, że nie tylko wytworzony kod jest lepszej jakości, ale produkt powstaje szybciej [8]. W takim podejściu nie ma możliwości prawidłowej interpretacji zawartości repozytorium, gdyż każdy kod powstaje w parach, a dodatkowo skład pary programistycznej może się codziennie zmieniać.

Należy wziąć pod uwagę również inne aspekty związane z wytwarzaniem oprogramowania przy ocenie pracowników na podstawie liczby wprowadzony, zmodyfikowanych czy usuniętych wierszy kodu. Na przykład doświadczeni programiści z reguły piszą bardziej zwięzły kod niż początkujący [9]. Ponadto w przypadku tworzenie interfejsów użytkowych często wykorzystywane są programy, które generują kod źródłowy na podstawie przygotowanych formatek. W ten sposób dla prostego interfejsu można w krótkim czasie wygenerować kilkaset wierszy kodu.

W związku z pojawiającymi się ograniczeniami metod oceny wkładu programisty mogą one być stosowane jako uzupełnienie oceny pracowników.

3. PROPONOWANA METODA POMIARU WKŁADU

Opracowana została metoda oceny wkładu programistów w końcowy kod źródłowy programu na podstawie danych przechowywanych w repozytoriach kodu. Do oceny wybrane zostały następujące operacja wykonywane przez programistów podczas tworzenie programu:

- dodanie wiersza kodu
- zmodyfikowanie wiersza kodu
- usunięcie wiersza kodu

Na szczególną uwagę zasługuje operacja modyfikacji wiersza kodu. Może to być poprawienie literówki, dodanie komentarza, ale również usunięcie poważnego błędu, czy zoptymalizowanie kodu.

Jednakże próba oszacowania stopnia skomplikowania wprowadzonej modyfikacji prowadziłaby do znacznego zwiększenie złożoności mechanizmu oceny. Podobnie operacja usunięcia wiersza kodu może dotyczyć zarówno usunięcie pustego wiersza, jak i optymalizacji kodu.

W tabeli 1 przedstawiono proponowane wagi dla poszczególnego typu operacji.

Tabela 1. Ważone operacje programistów

Symbol	Opis	Waga
a	dodanie *a* wierszy kodu	3
m	zmodyfikowanie *m* wierszy kodu	2
d	usunięcie *d* wierszy kodu	1

Na podstawie zdefiniowanych operacji i ich wag przyjęto, iż wkład programisty *x* w końcowy kod źródłowy programu jest ważoną sumą liczby wszystkich dodanych, zmodyfikowanych i usuniętych wierszy kodu przez programistę *x*, w stosunku do ważonej sumy wszystkich dodanych, zmodyfikowanych i usuniętych wierszy kodu we wszystkich rewizjach wprowadzonych przez wszystkich programistów.

Wkład ten opisuje następujący wzór.

$$W_x = \frac{\sum_{i=1}^{n}\left(3a_{ix} + 2m_{ix} + d_{ix}\right)}{\sum_{i=1}^{n}\left(3a_i + 2m_i + d_i\right)} \cdot 100$$

gdzie:
n – ilość rewizji kodu
a_{ix} – ilość dodanych wierszy kodu przez programistę x w i-tej rewizji
m_{ix} – ilość zmodyfikowanych wierszy kodu przez programistę x w i-tej rewizji
d_{ix} – ilość usuniętych wierszy kodu przez programistę x w i-tej rewizji
a_i – ilość dodanych wierszy kodu w i-tej rewizji
m_i – ilość zmodyfikowanych wierszy kodu w i-tej rewizji
d_i – ilość usuniętych wierszy kodu w i-tej rewizji

Miara wkładu programisty w końcowy kod źródłowy jest mierzona w procentach.

4. WALIDACJA

W celu walidacji zaproponowanej metody został przeprowadzony eksperyment na dwóch publicznych repozytoriach projektów Open Source, przechowywanych w serwisie GitHub. Do celów badania został opracowany przez autorów skrypt w języku *Perl*, który pobiera zawartość repozytorium i oblicza metrykę wkładu w końcowy kod źródłowy dla każdego programisty.

Do badania zostały celowo wybrane repozytoria o granicznych rozmiarach, odpowiednio 2,5MB i 560MB. Pierwszym wybranym repozytorium był projekt *jQuery*, w ramach którego tworzona jest jedna z najpopularniejszy bibliotek języka JavaScript ułatwiająca tworzenie złożonych i interaktywnych stron WWW. W momencie przeprowadzania badania do repozytorium zostało zaakceptowanych 5367 zgłoszeń, dokonanych przez ponad 200 deweloperów.

Do drugiego badania wykorzystano kopię głównego repozytorium organizacji *Mozilla*, składającego się z 354.460 rewizji, które zostały utworzone przez ponad 3.500 programistów. W poniższych tabelach zostali przedstawieni najbardziej aktywni deweloperzy, wraz z obliczonym wkładem w końcowy kod źródłowy projektu. Ponadto dane te zostały rozszerzone o informację zawierająca ilości zgłoszonych przez programistę rewizji. Dane osobowe programistów, które zostały zastąpione kolejnymi numerami, są dostępne w publicznych repozytoriach kodu na serwerach serwisu GitHub.

Projekt jQuery został zapoczątkowany w 2006 roku przez Johna Resiga. Jak widać na podstawie rezultatów badania zaprezentowanych w tabeli 2 jest on w dalszym ciągu najbardziej aktywną osobą w projekcie. Zgodnie z proponowaną metodą jego wkład w końcowy kod źródłowy programu jest szacowany na 36%. Co ciekawe łączny wkład trzech najbardziej aktywnych programistów wynosi ponad 50%. Jest to dość częsta sytuacja w niewielkich projektach Open Source, gdzie trzon programistów stanowi kilka osób, a pozostałe zaangażowane osoby sporadycznie przesyłają poprawki do programu.

Tabela 2. Wkład programistów w projekcie *jQuery*

Programista	Wkład	Zatwierdzone zmiany	
		Ilość	Udział
John Resig	36,57%	1714	31,94%
Programista 1	11,66%	471	8,78%
Programista 2	7,95%	330	6,15%
Programista 3	5,85%	330	6,15%
Programista 4	5,48%	200	3,73%
Programista 5	4,31%	71	1,32%
Programista 6	3,96%	250	4,66%
Programista 7	3,24%	67	1,25%
Programista 8	2,81%	310	5,78%
Programista 9	2,67%	499	9,30%
Programista 10	2,61%	85	1,58%
Programista 11	2,22%	154	2,87%

W tabeli 2 umieszczono również informację o liczbie zaakceptowanych zmian przez poszczególnych programistów. Wartość ta również może być traktowana jako bardzo prosta metryka zaangażowania programistów w projekt. Jednak z analizy zebranych danych wynika iż nie jest ona skorelowana z metryką proponowaną przez autorów. Przykładowo programista oznaczony numerem 9, zgłosił ponad 9% wszystkich zatwierdzonych zmian, jednak jego wkład w końcowy kod źródłowy wynosi niewiele ponad 2,5%. Dla tego przypadku przeprowadzono ręczną analizę wprowadzonych zmian, z której wynika, iż głównie polegały one na refaktoryzacji kodu i drobnych modyfikacjach. Nie były natomiast dodawane większe fragmenty programu.

Odwrotna sytuacja ma natomiast miejsce w przypadku programisty nr 5, który zgłosił 71 zmian, co stanowi zaledwie 1,32% wszystkich zgłoszeń. Jednak jego wkład w końcowy kod źródłowy został oceniony na 4,31%. W odróżnieniu od programisty nr 9, wprowadzał on i modyfikował duże fragmenty kodu.

Drugie badanie zostało przeprowadzone na znacznie większym repozytorium projektu Mozilla. Zaakceptowanych w nim zostało ponad 60 razy więcej zmian niż w pierwszym przypadku. Analiza wkładu programistów została

przedstawiona w Tabeli 3. Z uwagi na znacznie większą liczbę zaangażowanych programistów wkład poszczególnych programistów jest znacznie bardziej rozłożony. Wkład zaledwie dziewięciu programistów został oszacowany na więcej niż 1%. Jednak dla porównania pod względem ilości rewizji tylko dwóch programistów zgłosiło więcej niż 1% wszystkich zmian (programista nr 11 i 12). Należy zwrócić uwagę, iż wkład tych programistów w końcowy kod źródłowy jest stosunkowo niski (odpowiednio 0,85% i 0,36%). Z drugiej strony wkład programisty nr 9, pomimo 74 wprowadzonych zmian (0,02%), oszacowano na aż 1,5%.

Tabela 3. Wkład programistów w repozytorium *Mozilla*

Programista	Wkład	Zatwierdzone zmiany	
		Ilość	Udział
Programista 1	3,12%	2099	0,59%
Programista 2	2,98%	384	0,11%
Programista 3	2,66%	2763	0,78%
Programista 4	1,93%	1240	0,35%
Programista 5	1,74%	1812	0,51%
Programista 6	1,72%	2681	0,76%
Programista 7	1,57%	1040	0,29%
Programista 8	1,53%	987	0,28%
Programista 9	1,52%	74	0,02%
Programista 10	0,99%	2825	0,80%
Programista 11	0,85%	7219	2,04%
Programista 12	0,36%	4335	1,22%
Programista 13	0,32%	3287	0,93%

O ile w badaniu proponowaną metodą niewielkiego projektu występowała korelacja pomiędzy obliczonym wkładem, a ilością zgłoszonych zmian, to przy większym projekcie taka korelacja nie występuje.

Na podstawie przeprowadzonych badań można stwierdzić, iż nie jest wiarygodna ocena pracowitości programistów na podstawie ilości zaakceptowanych do repozytorium zmian, ponieważ niektórzy deweloperzy preferują częste wysyłanie drobnych poprawek, inni znacznie rzadziej wysyłają większe fragmenty kodu.

Przedstawiona metoda oceny wkładu z uwzględnieniem proponowanych wag operacji w sposób poprawny odzwierciedla faktyczny udział poszczególnych programistów w powstanie końcowej wersji kodu źródłowego programu.

5. PODSUMOWANIE

Opisana w niniejszym artykule metoda oceny wkładu programistów w końcowy kod źródłowy programu może być pomocna do identyfikacji kluczowych osób w projekcie. Jej główną zaletą jest możliwość przeprowadzenia automatycznych obliczeń na podstawie danych przechowywanych w repozytoriach kodu. Dzięki temu jest ona całkowicie transparentna dla programistów i można być przeprowadzona również dla projektów już ukończonych.

W dalszej badaniach planowane jest zweryfikowanie przyjętych wag dla poszczególnych operacji za pomocą wywiadów z kierownikami zakończonych projektów, których repozytoria zostaną zbadane proponowaną metodą. Planowane jest również rozszerzenie metody o operacje związane z analizą i projektowaniem programów.

6. BIBLIOGRAFIA

1. Rocznik Statystyczny Pracy 2010, Główny Urząd Statystyczny, 2010.
2. Albrecht A.J., Gaffney J.E.: Software function, source lines of code, and development effort prediction: a software science validation, IEEE Transactions on Software Engineering, 6, 1983
3. Boehm B.W.: Software engineering economics, IEEE Transactions on Software Engineering, 1, 1984
4. Kaczmarek J., Kucharski M: Size and effort estimation for applications written in Java, Information and Software Technology, 46.9, 2004
5. Humphrey W.S.: Using a defined and measured personal software process, Software, IEEE, 13.3, 1996
6. Hochstein L., Basili V.R, Zelkowitz M.V., Hollingsworth J.K., Carver J: Combining self-reported and automatic data to improve programming effort measurement, ACM SIGSOFT Software Engineering Notes, 30.5, 2005.
7. Ruparelia N.B.: The history of version control, ACM SIGSOFT Software Engineering Notes, 35.1 2010
8. Cockburn A., Williams L.: The costs and benefits of pair programming, Extreme programming examined, 2000
9. Begel A., Simon B.: Novice software developers, all over again, Proceedings of the Fourth international Workshop on Computing Education Research, ACM, 2008.

SOFTWARE DEVELOPER CONTRIBUTION IN THE FINAL SOURCE CODE

Key-words: Software engineering, effort estimation

Estimation of the software developers effort is a complex task. Project managers take into account factors such as the quality of the created code, the ability to solve technical and business problems, as well as productivity. However measuring the productivity of individual developers, is very complicated. The article presents a new method of assessing the contribution of the developer in the final source code of the program, which operates on the data stored in code repositories. It will present the results of the method validation conducted on two open source projects.