ISSN 1507-3858
e-ISSN 2450-0003

Edyta Pieszczyk, Jan Werewka
AGH Akademia Górniczo-Hutnicza w Krakowie
e-mails: edyta.pieszczyk@wp.pl; werewka@agh.edu.pl

ANALIZA PRZYCZYN PROBLEMÓW JAKOŚCI OPROGRAMOWANIA NA PODSTAWIE ANKIET OSÓB UCZESTNICZĄCYCH W PROCESIE WYTWARZANIA SYSTEMÓW INFORMATYCZNYCH

ANALYSIS OF THE REASONS FOR SOFTWARE QUALITY PROBLEMS BASED ON SURVEY OF PERSONS INVOLVED IN THE PROCESS OF DEVELOPING OF IT SYSTEMS

DOI: 10.15611/ie.2015.3.06

Streszczenie: Poziom niepowodzeń projektów informatycznych jest wysoki i nie zmienia się od wielu lat mimo znacznej poprawy procesów wytwarzania oprogramowania. Jedyną z oczywistych przyczyn tego stanu jest skracanie czasu dostawy systemów informatycznych w projektach oraz szybko zmieniająca się technologia. W związku z tym zapewnienie jakości w procesie wytwarzania oprogramowania stanowi kluczowy problem w kontekście poprawnego działania systemu informatycznego. W celu wskazania przyczyn problemów istotne jest poznanie punktu widzenia członków zespołów uczestniczących w procesie wytwarzania/rozwijania oprogramowania. Uzyskanie ich opinii było możliwe dzięki przeprowadzeniu interaktywnej ankiety, opracowanej według metody 5M. Zgodnie z nią na wstępie zidentyfikowano kategorie przyczyn za pomocą narzędzia do zarządzania jakością, jakim jest diagram Ishikawy. Następnie przeprowadzono analizę z wykorzystaniem zasady Pareto, co pozwoliło na wyodrębnienie najbardziej istotnych przyczyn występowania błędów w procesie wytwarzania oprogramowania. W efekcie szczegółowa analiza zaobserwowanych wyników oraz wnioski z tej analizy przyczynią się do poprawy procesów wytwarzania oprogramowania, a także polepszenia relacji między obszarami zarządzania projektami informatycznymi i wytwarzania oprogramowania.

Słowa kluczowe: zarządzanie projektami, wytwarzanie oprogramowania, jakość oprogramowania, diagram przyczynowo-skutkowy, Pareto.

Summary: The level of failure of IT projects is high and has not changed over many years despite improvement in software development processes. One of the obvious reasons for this is shortening the delivery time of information systems in the projects and rapidly changing technology. Quality assurance in software development process is a key issue in ensuring the proper operation of an IT system. In order to know the causes of the problems, it is

important to know viewpoints of members of the teams developing software. These views were obtained by carrying out the survey. At the beginning categories of reasons are identified using Ishikawa diagram, a tool for quality management. In the next step an analysis was performed using the Pareto principle. The analysis allowed the identification of the most important causes of errors in the software development process. A detailed analysis of the observed results and appropriate conclusions of this analysis will help improve the processes of software development and collaboration between project management and software development processes.

Keywords: project mangement, software development, software quality, cause and effect diagram, Pareto.

1. Wstęp

Rozwój technologii informatycznych sprawił, że w skali globalnej stymulują one rozwój większości dziedzin gospodarczych. Systemy informatyczne w coraz większym stopniu są również odpowiedzialne za nasze zdrowie i bezpieczeństwo, dlatego tak ważną kwestię stanowi ich jakość.

Na zagadnienie jakości oprogramowania można popatrzeć z dwóch różnych perspektyw. Pierwsza dotyczy poziomu kierownictwa wykonawców i odbiorców oprogramowania, zaś druga dotyczy jego wykonawców. Autorzy artykułu postanowili się odnieść do tych zagadnień z obu perspektyw. Jako wstępną motywację przyjęto zlecanie przez firmy ze Stanów Zjednoczonych firmom ulokowanym w Polsce prac i projektów budowy systemów informatycznych. Przykładowo wiele takich projektów wykonywanych jest w Krakowie – regionie, który jest największym centrum outsourcingu w Europie (zajmuje on dziewiąte miejsce na świecie). Według Tholons [Tholons 2015] w pierwszej setce mieszczą się również Warszawa (na pozycji 30.) oraz Wrocław (na pozycji 62.). Duża część prac wykonywanych w tych centrach dotyczy rozwoju oprogramowania. Ciekawe jest zagadnienie, jak bardzo zróżnicowane jest podejście do oceny jakości projektów informatycznych przez amerykańskie firmy będące odbiorcami oprogramowania i podejście zespołów wykonawczych z Polski.

Ujęcie zagadnienia z podziałem na pracę zespołów w siedzibie firmy (*on-site*) oraz pracę zdalną (*offshore*) było rozpatrywane w różnych pracach. Na przykład w Indiach [Kumar, Thangavelu 2013] był rozwijany model SOSPRM (*Software Outsourcing Service Provider Relationship Model*) dotyczący współpracy zespołów w siedzibie firmy i zespołów zdalnych. W wyniku przeprowadzonych prac stwierdzono, że w projektach globalnego rozwoju oprogramowania GSD (*Global Software Development*) istotne jest dzielenie się wiedzą, zaufanie, zaangażowanie i przepływ wymagań. Z kolei w opracowaniu ze Sri Lanki [Jayathilake i in. 2011] rozpatrywano kwestię weryfikacji jakości oprogramowania przez wydzierżawiane (outsourcing) firmy. W wyniku tych analiz rozpoznano główne przyczyny problemów związanych z jakością oprogramowania.

W niniejszym artykule zastosowano metodę, w której rozpatrywano podejście zespołów kierujących projektami powiązanych z klientami z jednej strony oraz zespołów wytwarzających oprogramowanie z drugiej strony. W celu zidentyfikowania różnicy w tych podejściach przeprowadzono wstępną analizę tego zagadnienia przez wykonanie kilku etapów prac opisanych w kolejnych punktach artykułu. Dokonano analizy oceny wykonania projektów informatycznych przez amerykańskie firmy konsultingowe. Następnie, dzięki przeprowadzonym dyskusjom z ekspertami i badaniom literaturowym, wyznaczono obszerną listę problemów związanych z jakością oprogramowania. W kolejnym etapie dokonano kategoryzacji tej listy, wykorzystując diagram przyczynowo-skutkowy Ishikawy. Z tej wstępnie uporządkowanej listy opracowano ankietę przeznaczoną dla członków zespołów rozwijających oprogramowanie. Uzyskano dzięki niej informację dotyczącą oceny problemów jakości oprogramowania. Wykorzystanie zasady Pareto w kolejnym kroku pozwoliło na wyodrębnienie i zastosowanie działań zapobiegawczych w stosunku do przyczyn generujących największą liczbę błędów, a tym samym powodujących największe koszty. Na bazie uzyskanych wyników wyciągnięto wstępne wnioski dotyczące poprawy procesów zarządzania i wytwarzania dostosowanych do potrzeb i możliwości organizacji wytwarzającej oprogramowanie. Przedstawione podejście ma nieskomplikowaną strukturę i może być stosowane w różnych firmach informatycznych, przy uwzględnieniu ich specyfiki i przy założeniu przeprowadzenia podobnej ankiety w ich siedzibach.

2. Problemy z projektami informatycznymi

W metodykach zarządzania projektami jakość produktów i procesów odgrywa jedną z głównych ról. Okazuje się jednak, że pojawiają się duże problemy związane z zarządzaniem projektami informatycznymi i nie są one rozwiązywane w sposób zadowalający. Świadczą o tym dane amerykańskiej grupy konsultingowej Standish Group, która sporządza jedną z najbardziej znanych statystyk dotyczących sukcesów bądź niepowodzeń projektów informatycznych, określanych jako raporty chaosu (*chaos reports*). Raport dotyczy wykonania projektów branży IT w Stanach Zjednoczonych. Statystyki te koncentrują się na sukcesie projektu bazującym na popularnym dla projektów trójkącie ograniczeń, który mówi, że projekt zakończył się sukcesem, jeżeli został wykonany:

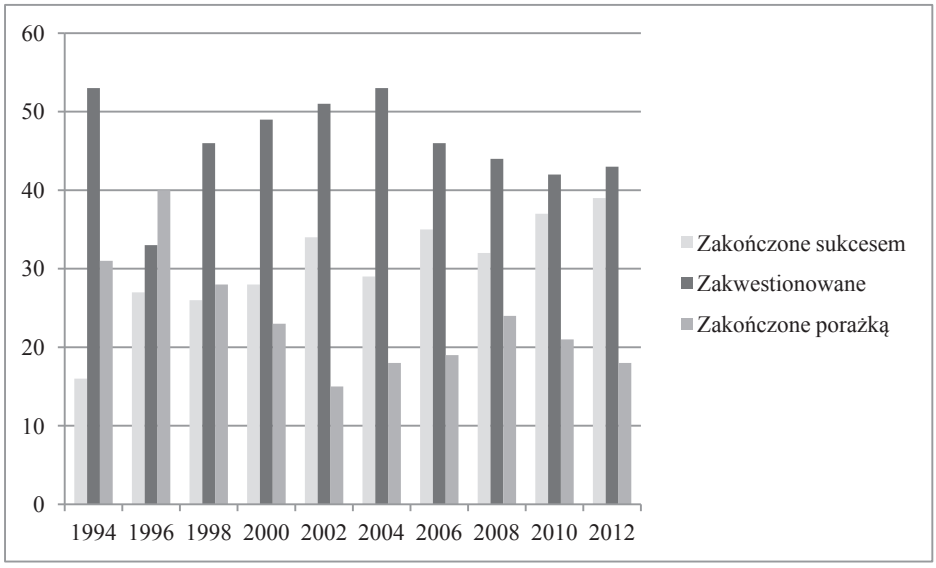
- na czas (harmonogram),
- przy zaplanowanym budżecie (koszty),
- z wymaganymi funkcjami i własnościami (zakres).

W raporcie Standish Group [The Standish Group 2014] wyróżnia się trzy typy zakończenia projektów:

- Projekt odniósł sukces (*successful*), jeżeli został wykonany w zaplanowanym czasie, a jego koszty nie przekroczyły zaplanowanego budżetu, ponadto zostały spełnione wszystkie własności i funkcje zdefiniowane na początku projektu.

- Projekt zakwestionowany (*challenged*), jeżeli został zakończony i produkty dostawy są przydatne w działalności operacyjnej, ale projekt nie spełnił przynajmniej jednego z warunków, którymi są dochowanie czasu realizacji, budżetu wykonania lub wyspecyfikowanych na początku własności i funkcji.
- Projekt zakończył się niepowodzeniem (*failed*), jeżeli został przerwany lub nie wykorzystano produktów powstałych w wyniku realizacji projektu.

Wyniki tych badań w ostatnich dwudziestu latach [The Standish Group 2013] wykazują, że duża część projektów jest kwestionowana lub kończy się niepowodzeniem, a w ostatnich kilku latach sytuacja ta nie poprawiła się (rys. 1).



Rys. 1. Efektywność projektów informatycznych według danych Standish Group Inc.

Źródło: opracowanie własne na podstawie [The Standish Group 2013].

W raporcie wskazano również, że w ostatnim czasie nastąpiło pogorszenie wyników projektów w porównaniu z rokiem 2004. Jest to o tyle dziwne, że równocześnie zwiększyła się liczba ekspertów w dziedzinie zarządzania projektami, organizowane są liczne profesjonalne szkolenia oraz powstały dedykowane narzędzia. Dodatkowo pełniejsza jest informacja o metodykach zarządzania, wypracowano również lepsze zasady współpracy między interesariuszami projektu. Z drugiej strony, ze względu na dynamiczny rozwój sektora informatycznego, stosowane są coraz nowsze (nie zawsze dobrze rozpoznane rozwiązania), zwiększyła się złożoność systemów, a czas dostawy produktów się skrócił.

Zasady podawania statystyk Standish Group budzą wątpliwości [Eveleens, Verhoef 2010]. Dotyczą one sposobu pojmowania sukcesu projektu, który nie zawsze jest mierzony przez podane trzy wskaźniki. Niekiedy bowiem o sukcesie projektu

mogą w znacznym stopniu decydować inne wskaźniki, takie jak: zadowolenie klienta, użyteczność, zysk, jakość dostarczanych produktów czy też poziom ryzyka.

Mimo różnych wątpliwości wobec podawanych statystyk raporty te w branży informatycznej stanowią punkt odniesienia do oceny wyników projektów. Można mieć różne wątpliwości wobec podanych statystyk, lecz widoczne jest, że projekty informatyczne należą do przedsięwzięć, dla których stopień ryzyka i niepowodzenia jest bardzo wysoki.

Jako przyczyny niepowodzeń projektów podawane są [The Standish Group 2014]: niekompletne wymagania (13,1%), brak zaangażowania klienta (12,4%), brak zasobów (10,6%), nierealne oczekiwania (9,9%), brak wsparcia ze strony kierownictwa (9,3%), zmiana wymagań i specyfikacji (8,7%), brak planowania (8,1%), projekt stał się niepotrzebny (7,5%), brak zarządzania IT (6,2%), technologiczny analfabetyzm (4,3%) i inne (9,9%).

Jako przyczyny kwestionowania projektów wymienia się [The Standish Group 2014]: brak zaangażowania klienta (12,8%), niekompletne wymagania i specyfikacje (12,3%), zmieniające się wymagania i specyfikacje (11,8%), brak zaangażowania kierownictwa (7,5%), brak kompetencji technologicznych (7,0%), brak zasobów (6,4%), nierealne oczekiwania (5,9%), niejasne cele (5,3%), nierealne ramy czasowe (4,3%), nowe technologie (3,7%) i inne (23,0%).

Powstaje pytanie, jakie są podstawowe trudności w realizacji projektów informatycznych w porównaniu z innymi branżami. Otóż: do najbardziej podstawowych należą następujące:

- Oprogramowanie jest trudne do przedstawienia i trudno wyjaśnić jego wymagania.
- Rzadko dwa takie same systemy oprogramowania są budowane wielokrotnie (zwykle nie ma analogii do podobnych, już funkcjonujących systemów).
- Przeprowadzenie projektu informatycznego jest działaniem złożonym i obciążonym dużym ryzykiem (wynikającym np. ze zmiany technologii w trakcie trwania projektu).
- Przy tworzeniu oprogramowania obserwuje się chęć do wprowadzania zmian powiązaną z problemami z dokładnym definiowaniem wymagań.

Firmy informatyczne próbują opanować sytuację przez definiowanie architektury korporacyjnej i wprowadzanie ładu architektonicznego [Werewka, Jamróz, Pitulej 2014].

Raporty Standish Group rozpatrują bardziej stronę odbiorców, to znaczy klientów, dla których to oprogramowanie jest dostarczone. W niniejszym artykule przedstawiony zostanie punkt widzenia dostawców, a w szczególności członków zespołów rozwijających oprogramowanie.

Ciekawą analizę przedsięwzięć informatycznych zawiera opracowanie [Wachnik 2013], przedstawiające sytuację na rynku polskim. Z opracowania tego wynika, że firmy głównie adaptują funkcjonujące systemy i nie realizują nowych, dużych lub bardzo dużych projektów. Taka sytuacja prowadzi do tego, że przedsięwzięcia są mniej ryzykowne, lecz prawdopodobnie są również mniej innowacyjne.

W publikacji [Lavallée, Robillard 2012] dokonano przeglądu wpływu poprawy procesów rozwoju oprogramowania (SPI – *Software Process Improvements*) na osoby i zespoły rozwijające oprogramowanie. Przeprowadzone badania pozwolą na identyfikację przeszkód i szans związanych z osiągnięciem celów. W pracy analizowane są te czynniki pod względem pozytywnego i negatywnego wpływu na członków oraz zespoły rozwijające oprogramowanie.

3. Dobór metody badawczej

W niniejszym artykule przyjęto arbitralnie, że wyznacznikiem problemów związanych z jakością oprogramowania z punktu widzenia zarządzania projektami są raporty Standish Group. Przy takim założeniu należało się skupić na analizie problemów związanych z jakością oprogramowania w zespołach wytwarzających oprogramowanie.

W tym celu starano się zidentyfikować możliwie najobszerniejszą listę problemów związanych z jakością oprogramowania, zgłaszanych przez zespoły programistyczne. Ważne było, by nie sugerować się przy tym raportami Standish Group lub innymi raportami opracowanymi z punktu widzenia kierowników projektów. Dlatego skoncentrowano się głównie na problemach jakości oprogramowania zgłaszanych na stronach internetowych w Polsce lub przez osoby związane z procesem wytwarzania oprogramowania.

Zagadnienie jakości oprogramowania rozpatrywano z punktu widzenia zarządzania projektami i procesów wytwarzania oprogramowania. Podobne zagadnienia badano [Werewka 2015], analizując atrybuty jakości na poziomach architektury korporacyjnej i architektury oprogramowania.

Przy identyfikacji przyczyn problemów w procesie wytwarzania oprogramowania należy posłużyć się właściwymi narzędziami. W niniejszym opracowaniu do zidentyfikowania i kategoryzacji przyczyn posłużono się diagramem Ishikawy, który jest uznawany za jeden z siedmiu podstawowych narzędzi zarządzania jakością [Ishikawa 1986; Mach, Guaqueta 2001]. Diagram przyczynowo-skutkowy, zwany również diagramem Ishikawy lub diagramem rybiej ości, opracowany został przez japońskiego profesora Kaoru Ishikawę (1915-1989), uznawanego w Stanach Zjednoczonych za jednego z bardziej znanych innowatorów w dziedzinie zarządzania jakością. Proces budowy diagramu, tj. zbierania informacji stanowiących źródło diagramu, zgodnie z założeniami jego twórcy, wymaga pracy grupowej, określanej jako burza mózgów.

Nie bez znaczenia jest również graficzna forma prezentacji diagramu, wpływająca na jego przejrzystość i czytelność [Hamrol, Mantura 2012]. Taka forma przekazu ułatwia identyfikację i określenie wpływu czynników różnego rodzaju na ten sam efekt końcowy oraz zidentyfikowanie wzajemnych powiązań czynników wywołujących określony problem. Budowa diagramu Ishikawy składa się z czterech etapów; są nimi: identyfikacja problemu, określenie przyczyn głównych, uszczegółowienie przyczyn,

analiza wyników. W identyfikacji przyczyn głównych zastosowanie ma podejście bazujące na predefiniowanych strukturach. Metoda 5M [Hamrol 2005] zakłada, że przyczyny powinny być poszukiwane w jednym z pięciu głównych obszarów 5M, tj.: *manpower* (człowiek), *method* (metoda), *machine* (maszyna), *material* (materiał), *managment* (zarządzanie). Istotną kwestią związaną z tworzeniem diagramu Ishikawy jest zaangażowanie osób z doświadczeniem, jednoznacznie rozumiejących cel analizy, specjalizujących się w różnych dziedzinach związanych z rozpatrywanym problemem.

Ważne są cel, sposób, wiarygodność i koszt przeprowadzania ankiet. Relacje między firmami zlecającymi a firmami outsourcingowymi ulokowanymi w Polsce mają duże znaczenie gospodarcze. Można sobie wyobrazić projekt rządowy, którego celem jest poprawa konkurencji firm outsourcingowych w Polsce. Jest to istotne zagadnienie, silnie zależne od warunków ekonomicznych w kraju, w krajach prowadzących działalność outsourcingową oraz w krajach zlecających pracę firmom outsourcingowym. Przeprowadzenie ankiet na poziomie krajowym lub regionu jest bardzo kosztowne i musi je poprzedzać seria mniejszych analiz. W opracowaniu przeprowadzono badania bez jakiegokolwiek wsparcia finansowego, mające na celu wychwycenie podstawowych zależności, bazując na wynikach uzyskanych od wystarczająco reprezentatywnej grupy osób. Wyniki badań mogą być przydatne zespołom wytwarzającym oprogramowanie i badaczom tego obszaru. Podejście proponowane w artykule jest zbliżone do prac nad długiem technicznym w zwinnych metodykach rozwoju oprogramowania [Holvitie, Leppanen, Hyrynsalmi 2014] bazujących na ankietach przeprowadzonych w przemyśle. W wyniku analizy ankiet w artykule zwrócono uwagę na istotny wpływ architektury i struktury implementacyjnej na dług techniczny.

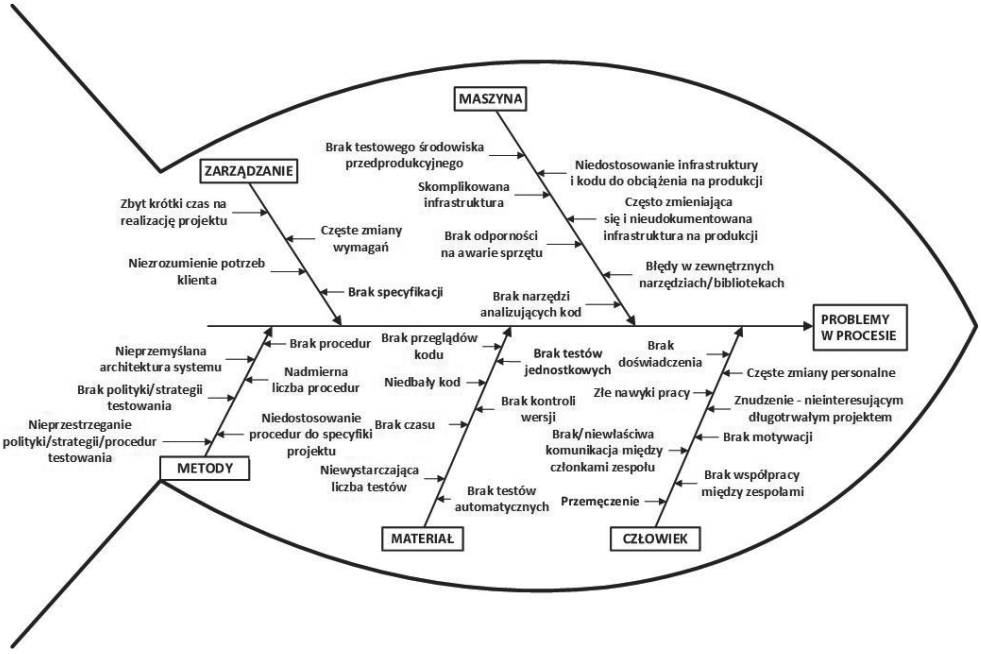
Liczba zidentyfikowanych przyczyn oddziałujących na jakość oprogramowania jest duża, dlatego zasadne stało się określenie, które czynniki mają na tę jakość największy wpływ. W pracy zastosowano zasadę Pareto, pozwalającą na wyselekcjonowanie najistotniejszych czynników odpowiedzialnych za występowanie większości problemów, a w konsekwencji − na podjęcie właściwych działań celem ich eliminacji oraz redukcji kosztów. Zasada Pareto sformułowana została przez włoskiego ekonomistę i socjologa Vilfredo Federico Damaso Pareto [*Vilfredo Pareto* 2015] w 1906 r. Podczas prowadzonych badań nad zamożnością społeczeństwa Pareto zaobserwował, że ok. 20% rodzin włoskich jest w posiadaniu 80% bogactw całego kraju. Z biegiem czasu odkrył, że prawidłowość ta ma zastosowanie nie tylko w ekonomii, ale również w prawie każdej dziedzinie życia. Zasada ta znana jest dzisiaj również pod nazwą 80/20, najogólniej mówiąc, stanowi ona, że zazwyczaj 20-30% przyczyn generuje 70-80% skutków [*Vilfredo Pareto* 2015]. Zasada Pareto wykorzystywana jest np. w niezależnej weryfikacji i walidacji oprogramowania (IV&V − *Independent Verification and Validation*) [Port i in. 2007] lub wykrywaniu komponentów narażonych na defekty [Oyetoyan, Conradi, Cruzes 2013].

Zaproponowana w opracowaniu metoda badawcza posłuży do wykrywania zależności występujących między procesami zarządzania projektami informatycznymi

a wytwarzania oprogramowania. Przedstawiona metodyka ma prostą strukturę, umożliwiającą jej zastosowanie na poziomie pojedynczej firmy w celu wykrycia zależności specyficznych dla określonej organizacji.

4. Identyfikacja i kategoryzacja przyczyn problemów w procesie wytwarzania oprogramowania

Analizę występowania przyczyn w procesie wytwarzania oprogramowania rozpoczęto od zidentyfikowania czynników wywołujących problemy w tym procesie oraz przyporządkowania ich do odpowiedniej kategorii czynników, zgodnie zasadami metody 5M.



Rys. 2. Diagram Ishikawy dotyczący problemów w procesie wytwarzania oprogramowania

Źródło: opracowanie własne.

Identyfikacja potencjalnych przyczyn problemów związanych z jakością oprogramowania polegała na przeprowadzeniu dyskusji i niesformalizowanych wywiadów z ekspertami uczestniczącymi w procesach wytwarzania oprogramowania. Ponadto dokonano przeglądu stron internetowych w celu identyfikacji dodatkowych przyczyn [Źródła błędów … 2010; *Dlaczego oprogramowanie*… 2014]. Również dodatkowo dokonano analizy stron związanych z testowaniem oprogramowania, np. dotyczących strategii testowania opisującej metody testowania stosowane w organizacji [Amber Team 2015]. W ten sposób uzyskano listę zawierającą 32 przyczyny. W następnym

kroku dokonano przydziału elementów tej listy do poszczególnych kategorii. Ostatecznie uzyskano wyniki, które zostały przedstawione w formie graficznej za pomocą diagramu Ishikawy (rys. 2).

Na rysunku 2 dwie kategorie mogą budzić wątpliwości w odniesieniu do wytwarzania oprogramowania, dlatego postanowiono je krótko wyjaśnić. Kategoria „materiał" odnosi się do komponentów i kodu oprogramowania, przypadków testowych, dokumentacji oraz innych elementów potrzebnych do uzyskania docelowego produktu, jakim jest system informatyczny. Kategoria „maszyna" dotyczy środowisk wytwarzania oprogramowania, środowisk testowych oraz docelowej infrastruktury wdrożeniowej.

Podsumowując, należy stwierdzić, że uzyskano wystarczająco długą i spójną listę przyczyn występowania problemów w procesie wytwarzania oprogramowania. Przeprowadzone ankiety miały wykazać, które czynniki odgrywają najbardziej istotną rolę.

5. Zasady przeprowadzania ankiety

Na podstawie zidentyfikowanych kategorii przyczyn i samych przyczyn sporządzono ankietę składającą się z pięciu pytań grupujących przyczyny występowania błędów według przynależności do danej kategorii (zgodnie z założeniami metody 5M). Dodatkowo ankietowani mogli zgłosić dodatkowe przyczyny, nieujęte w dostarczonej liście.

W interaktywnym badaniu ankietowym (interankiety.Pl), przeprowadzonym w maju 2015 r., wzięło udział 55 osób czynnie uczestniczących w procesie wytwarzania oprogramowania, z czego największy odsetek stanowili programiści oraz testerzy, najmniejszy zaś − analitycy i kadra zarządzająca. Ze względu na anonimowość ankiety nie jest możliwe dokładne sprecyzowanie procentowego udziału poszczególnych osób z podziałem na stanowiska. Ankieta była przygotowana w takiej formie, że każdy uczestnik mógł wypełnić ją tylko raz. Składała się z pięciu pytań wielokrotnego wyboru oraz dodatkowej opcji umożliwiającej wpisanie własnej odpowiedzi (tzw. pytanie otwarte).

W celu uzyskania obiektywnego wyniku badanie przeprowadzone zostało wśród osób z pięciu niezależnych od siebie firm z branży IT. Firmy te wykonują zlecane projekty i charakteryzują się różną dojrzałością procesów wytwarzania oprogramowania. W tabeli 1 zawarto wyniki ankiet przeprowadzonych ze względu na kategorie przyczyn problemów.

Tabela 1. Ilościowe zestawienie przyczyn problemów w procesie wytwarzania oprogramowania z uwzględnieniem procentowego udziału w ogólnej liczbie wystąpień

Kategorie przyczyn	Zidentyfikowane przyczyny	Liczba wystąpień	Procentowy udział w ogólnej liczbie wystąpień
1	2	3	4
Człowiek	Brak doświadczenia	41	11,26
	Brak motywacji	2	0,55

Tabela 1, cd.

1	2	3	4
	Częste zmiany personalne	3	0,82
	Brak współpracy między zespołami	19	5,22
	Złe nawyki pracy	2	0,55
	Przemęczenie	2	0,55
	Brak/niewłaściwa komunikacja między członkami zespołu	8	2,20
	Znudzenie nieinteresującym, długotrwałym projektem	1	0,27
Zarządzanie	Zbyt krótki czas na realizację projektu	44	12,09
	Brak specyfikacji	7	1,92
	Częste zmiany wymagań	20	5,49
	Niezrozumienie potrzeb klienta	3	0,82
Metody	Brak polityki/strategii testowania	39	10,71
	Nadmierna liczba procedur	1	0,27
	Nieprzemyślana architektura systemu	22	6,04
	Brak procedur	5	1,37
	Niedostosowanie procedur do specyfiki projektu	1	0,27
	Nieprzestrzeganie polityki/strategii/procedur testowania	1	0,27
Materiał	Niedbały kod	43	11,81
	Brak przeglądów kodu	5	1,37
	Niewystarczająca liczba testów	8	2,20
	Brak testów automatycznych	4	1,10
	Brak testów jednostkowych	8	2,20
	Brak kontroli wersji	3	0,82
	Brak czasu	1	0,27
Maszyna	Brak testowego środowiska przedprodukcyjnego	39	10,71
	Skomplikowana infrastruktura	4	1,10
	Niedostosowanie infrastruktury i kodu do obciążenia na produkcji	13	3,57
	Brak odporności na awarie sprzętu	1	0,27
	Często zmieniająca się i nieudokumentowana infrastruktura produkcji	7	1,92
	Błędy w zewnętrznych narzędziach/bibliotekach	3	0,82
	Brak narzędzi analizujących kod	4	1,10
	SUMA	364	100,00

Źródło: opracowanie własne na podstawie przeprowadzonych ankiet.

Podział na kategorie ułatwił wyznaczenie różnych rodzajów problemów występujących w procesie wytwarzania oprogramowania. Ankietowani koncentrowali się w pierwszej kolejności na kategoriach, a następnie – na problemach wewnątrz kategorii.

6. Zastosowanie zasady Pareto przy analizie ankiet

W opracowaniu było istotne, czy na podstawie przeprowadzonych ankiet można wyciągnąć wnioski i ocenić różnice opisanych dwóch podejść. Analiza danych obejmowała takie etapy, jak [Diagram Pareto 2015]:

1. Zebranie danych w celu zidentyfikowania przyczyn wpływających na ogólną wartość analizowanego zjawiska.
2. Zebranie danych dotyczących wartości generowanej przez każdą zidentyfikowaną przyczynę.
3. Uporządkowanie przyczyn problemu według częstotliwości występowania.
4. Obliczenie wartości skumulowanej dla każdej zidentyfikowanej przyczyny.
5. Przedstawienie wyników skumulowanych na wykresie kolumnowym.
6. W celu łatwiejszego zidentyfikowania przyczyn mających 70-, 80-procentowy wkład w zaistniały problem narysowano krzywą Lorenza, łączącą punkty odpowiadające sumie skumulowanych wartości poszczególnych przyczyn.
7. Analiza wykresu, wyciągnięcie wniosków oraz podjęcie ewentualnych działań korygujących.

Zgodnie z założeniami zasady Pareto działania korygujące w pierwszej kolejności powinny się skupić na grupie najwyższego ryzyka, gdyż oddziałując tylko na 20--30% ogólnej liczby przyczyn, a tym samym ponosząc relatywnie niewielkie koszty, jesteśmy w stanie wyeliminować 70-80% negatywnych skutków.

W odniesieniu do pozostałych przyczyn należy przeprowadzić dokładną analizę, gdyż często okazuje się, że działania korygujące nie znajdują uzasadnienia ekonomicznego.

Przyjmując wstępnie, że poszczególne kategorie przyczyn są jednakowo ważne, postanowiono wyznaczyć najbardziej istotne czynniki wpływające na jakość oprogramowania. W tym celu uporządkowano wartości uzyskane z ankiet w sposób malejący oraz obliczono skumulowany procent wystąpień (tab. 2).

Tabela 2. Zestawienie skumulowanego procentu wystąpień przyczyn problemów w procesie wytwarzania oprogramowania

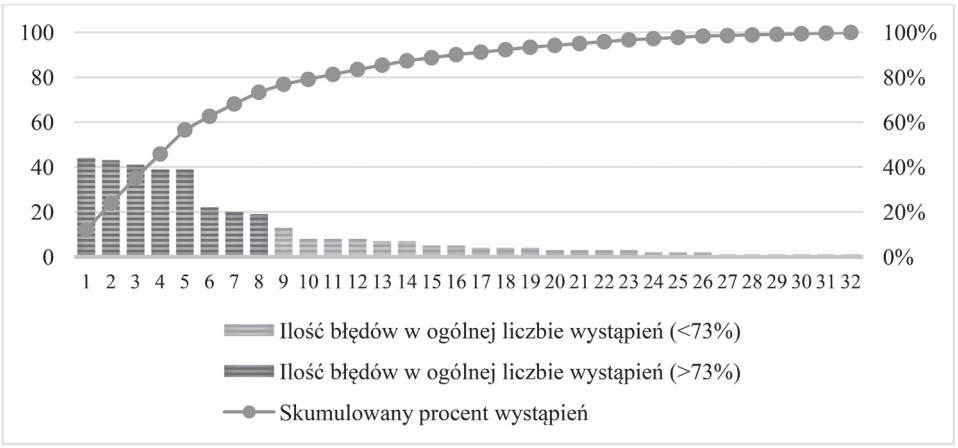
Lp.	Zidentyfikowane przyczyny	Liczba wystąpień	Procentowy udział w ogólnej liczbie wystąpień	Skumulowany procent wystąpień
1	2	3	4	5
1	Zbyt krótki czas na realizację projektu	44	12,09	12,09
2	Niedbały kod	43	11,81	23,90
3	Brak doświadczenia	41	11,26	35,16

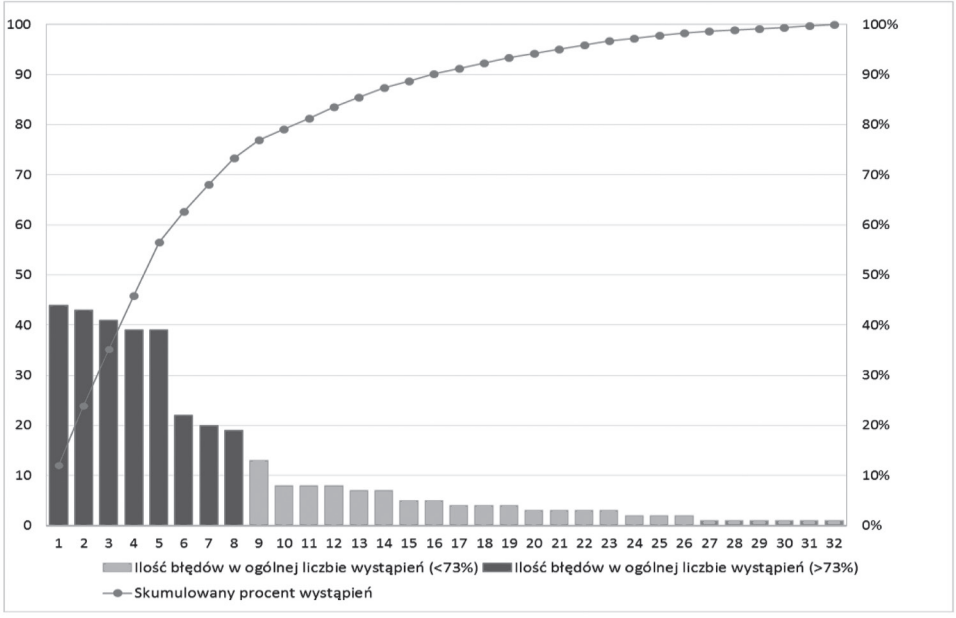
Tabela 2, cd.

1	2	3	4	5
4	Brak polityki/strategii testowania	39	10,71	45,88
5	Brak testowego środowiska przedprodukcyjnego	39	10,71	56,59
6	Nieprzemyślana architektura systemu	22	6,04	62,64
7	Częste zmiany wymagań	20	5,49	68,13
8	Brak współpracy między zespołami	19	5,22	73,35
9	Niedostosowanie infrastruktury i kodu do obciążenia na produkcji	13	3,57	76,92
10	Niewystarczająca liczba testów	8	2,20	79,12
11	Brak testów jednostkowych	8	2,20	81,32
12	Brak/niewłaściwa komunikacja między członkami zespołu	8	2,20	83,52
13	Brak specyfikacji	7	1,92	85,44
14	Często zmieniająca się i nieudokumentowana infrastruktura produkcji	7	1,92	87,36
15	Brak procedur	5	1,37	88,74
16	Brak przeglądów kodu	5	1,37	90,11
17	Brak testów automatycznych	4	1,10	91,21
18	Skomplikowana infrastruktura	4	1,10	92,31
19	Brak narzędzi analizujących kod	4	1,10	93,41
20	Niezrozumienie potrzeb klienta	3	0,82	94,23
21	Częste zmiany personalne	3	0,82	95,05
22	Błędy w zewnętrznych narzędziach/ bibliotekach	3	0,82	95,88
23	Brak kontroli wersji	3	0,82	96,70
24	Przemęczenie	2	0,55	97,25
25	Brak motywacji	2	0,55	97,80
26	Złe nawyki pracy	2	0,55	98,35
27	Niedostosowanie procedur do specyfiki projektu	1	0,27	98,63
28	Znudzenie nieinteresującym, długotrwałym projektem	1	0,27	98,90
29	Nadmierna liczba procedur	1	0,27	99,18
30	Nieprzestrzeganie polityki/strategii/ procedur testowania	1	0,27	99,45
31	Brak czasu	1	0,27	99,73
32	Brak odporności na awarie sprzętu	1	0,27	100,00

Źródło: opracowanie własne na podstawie przeprowadzonych ankiet.

Na podstawie tak przeprowadzonej analizy sporządzono wykres Pareto. Następnie w celu lepszego zobrazowania przyczyn odpowiedzialnych za występowanie 70-80% błędów wykres uszczegółowiono tzw. krzywą Lorenza, powstałą wskutek połączenia ze sobą punktów stanowiących skumulowany procentowy udział wyszczególnionych przyczyn.





Rys. 3. Analiza Pareto-Lorenza dotycząca przyczyn błędów w procesie wytwarzania oprogramowania

Źródło: opracowanie własne na podstawie przeprowadzonych ankiet.

Zgodnie z zasadą Pareto analiza obliczeń wykazała, że 8 z 32 wyszczególnionych przyczyn odpowiada za występowanie ok 73% błędów. Stanowi to ok. 25% wszystkich zidentyfikowanych przyczyn, do których zalicza się:

- zbyt krótki czas na realizację projektu − 12,1% (nierealne ramy czasowe – 4,3%),
- niedbały kod − 11,8% (brak kompetencji technologicznych – 7,0%),
- brak doświadczenia − 11,3% (brak zasobów – 6,4%),
- brak polityki/strategii testowania − 10,7% (niejasne cele – 5,3%),
- brak testowego środowiska przedprodukcyjnego − 10,7% (nowe technologie − 3,7%),
- nieprzemyślaną architekturę systemu − 6,0% (niekompletne wymagania i specyfikacje − 12,3%),
- częste zmiany wymagań − 5,5% (zmieniające się wymagania i specyfikacje – 11,8%),
- brak współpracy między zespołami − 5,2% (brak zaangażowania klienta − 12,8%).

W nawiasach podano skojarzone przez autorów przyczyny kwestionowania projektu podane w raporcie Standish Group [The Standish Group 2014]. Wynika z tego, że obserwuje się rozbieżność między wynikami raportu Standish Group i przeprowadzonej ankiety. Różnice wynikają z innego punktu widzenia problemów związanych z projektem.

Mimo że wszystkie wyszczególnione przyczyny są bardzo istotne w procesie wytwarzania oprogramowania, ze względu na fakt, że każda z tych przyczyn może generować błędy, na wskazanych powyżej ośmiu czynnikach należy skupić największą uwagę. Pozwoli to zapobiec występowaniu błędów w procesie wytwarzania oprogramowania oraz ograniczyć koszty związane z ewentualnym naprawieniem ujawnionych błędów.

7. Zagadnienie poprawy jakości oprogramowania

W celu sprostania opisanym problemom należy, uwzględniając specyfikę przemysłu informatycznego, wypracować i zastosować odpowiednie metody zarządzania projektami, zastosować odpowiednie metody wytwarzania oprogramowania pozwalające na częstą weryfikację jego jakości, a także dokonać poprawy procesów zarządzania i wytwarzania dostosowanych do potrzeb i możliwości organizacji.

W tabeli 3 zawarto propozycje działań zmierzających do zminimalizowania występowania błędów w procesie wytwarzania oprogramowania dla ośmiu przyczyn wyszczególnionych z zastosowaniem zasady Pareto. Zaproponowane działania zapobiegawcze to wynik własnych przemyśleń i dyskusji z ekspertami z dziedziny wytwarzania oprogramowania. Propozycje zawarte w tabeli 3 przedstawiają możliwe rozwiązania, a każde przedsiębiorstwo informatyczne może mieć w tym względzie własne zalecenia.

Tabela 3. Zestawienie działań eliminujących osiem głównych przyczyn problemów w procesie wytwarzania oprogramowania

Kategorie przyczyn	Zidentyfikowane przyczyny	Proponowane działania zapobiegawcze
1	2	3
Zarządzanie	Zbyt krótki czas na realizację projektu	Jednym z czynników zmniejszających ryzyko wystąpienia błędów spowodowanych zbyt krótkim czasem przeznaczonym na realizację projektu jest właściwy dobór kadry zarządzającej projektem, a w szczególności project managera (PM), który powinien mieć duże doświadczenie w realizacji tego typu projektów. Jeśli PM nie ma stosownej wiedzy, powinien mieć możliwość uzyskania opinii eksperckiej. Mimo wszystko pojęcie czasu w realizacji projektów w branży IT jest skomplikowanym zagadnieniem, z którym czasem nie są sobie w stanie poradzić nawet najlepsi specjaliści.
	Częste zmiany wymagań	Dobrą praktyką, mającą na celu zapobieganie występowania błędów wynikających z częstych zmian wymagań, jest zalecenie, aby wszelkie zmiany wymagań były uzgadniane i akceptowane przez uczestniczące w projekcie strony oraz sporządzane w formie pisemnej. Ma to również na celu zapobieganie powstaniu błędów związanych z niezrozumieniem potrzeb klienta, z ich nieprecyzyjnością czy też nieuzasadnionymi zmianami.
Materiał	Niedbały kod	Dobry kod powinien charakteryzować się przede wszystkim czytelnością, prostotą i łatwością zrozumienia, tzw. utrzymywalnością i przenaszalnością. Błędy w procesie wytwarzania oprogramowania, powodowane niedbale napisanym kodem, mogą być niwelowane przez szkolenia deweloperów, standaryzację, właściwe warunki pracy oraz brak pośpiechu w realizacji projektu.
Człowiek	Brak doświadczenia	Błędy spowodowane niedoświadczeniem osób uczestniczących w realizacji projektu można zminimalizować przez wprowadzenie szkoleń dziedzinowych. Dobrą praktyką może być również wyznaczenie opiekuna, którego zadaniem będzie stopniowe wdrażanie mniej doświadczonych pracowników w realizowane zadania.
	Brak współpracy między zespołami	Minimalizacja błędów spowodowanych tym czynnikiem może się odbywać przez zlokalizowanie zespołów relatywnie blisko siebie tak, aby możliwa była ich integracja i swobodna komunikacja. Ważne są również rola kierowników poszczególnych zespołów i ich umiejętności komunikacyjne i zarządcze.
Maszyna	Brak testowego środowiska przedprodukcyjnego	Ideą utworzenia testowego środowiska przedprodukcyjnego jest możliwość testowania rozwiązań utworzonych w środowisku programistycznym. Jego brak może

Tabela 3, cd.

1	2	3
		skutkować występowaniem błędów w środowisku produkcyjnym. W celu ograniczenia występowania błędów generowanych brakiem testowego środowiska przedprodukcyjnego postuluje się jego utworzenie z uwzględnieniem zalecenia, aby środowisko to było jak najbardziej zbliżone do środowiska produkcyjnego.
Metody	Brak polityki/strategii testowania	Eliminacja błędów związanych z brakiem polityki/strategii testowania będzie polegać na jej utworzeniu z jednoczesnym dostosowaniem do typu projektów realizowanych przez organizację.
	Nieprzemyślana architektura systemu	Rozwiązaniem, które powinno być zastosowane w celu wyeliminowania błędów związanych z nieprzemyślaną architekturą systemu, jest zatrudnienie architekta systemu wyróżniającego się dużym doświadczeniem zawodowym w realizacji dziedzinowych projektów.

Źródło: opracowanie własne.

Tabela 3 zawiera propozycje działań zapobiegawczych w firmie ousourcingowej. Jeżeli wiemy, jakie występują problemy związane z jakością wytwarzania oprogramowania, to należy się zastanowić, jak nawiązać do problemów zleceniodawcy opisanych w raportach Standish Group. Na podstawie danych można określić wstępnie, że największymi trzema problemami do wspólnego przedyskutowania są: niedbały kod − 11,8% (brak kompetencji technologicznych − 7,0%), brak doświadczenia − 11,3% (brak zasobów − 6,4%), nieprzemyślana architektura systemu 6,0% (niekompletne wymagania i specyfikacje − 12,3%). Oczywiście, te rozważania mają charakter zgubny, jednakże wskazują na ogólne podejście do zagadnienia.

W niniejszym punkcie wskazano, że oprócz poprawy jakości wybranych procesów wytwarzania oprogramowania warto znaleźć najsilniejsze korelacje z problemami firmy zlecającej w celu rozwiązania najbardziej pilnych wspólnych problemów.

8. Zakończenie

Głównym celem niniejszego artykułu było wyznaczanie przyczyn problemów w procesie wytwarzania oprogramowania i porównanie ich z problemami na poziomie zarządzania projektami. Taki przykład zależności występuje, gdy zespoły wykonawcze są zespołami wydzierżawionymi – pojawia się tu zagadnienie podziału na pracę zespołów w siedzibie firmy (*on-site*) oraz pracę zdalną *(offshore*).

Zastosowano systematyczne podejście, bazujące na uznanych narzędziach, jakimi są diagram przyczynowo-skutkowy i zasada Pareto. Wskazany cel osiągnięto na podstawie badania ankietowego przeprowadzonego na grupie osób czynnie uczestniczących w procesie wytwarzania oprogramowania. Zastosowanie diagramu

przyczynowo-skutkowego umożliwiło zidentyfikowanie 32 przyczyn problemów w procesie wytwarzania oprogramowania, natomiast zastosowanie zasady Pareto pozwoliło na usystematyzowanie wyszczególnionych przyczyn ze względu na częstotliwość ich występowania. Dzięki obiektywnej analizie wyodrębniono osiem kluczowych przyczyn, stanowiących 25% wszystkich przyczyn, generujących aż 73% błędów. Uzyskane wyniki analizy stanowiły następnie punkt wyjścia do kontynuowania prac, których celem była dalsza identyfikacja i analiza problemów oraz zaproponowanie konkretnych działań zapobiegawczych, pozwalających na ograniczenie występowania problemów związanych z jakością oprogramowania. Przedstawiona metoda może być dostosowana do potrzeb firm po uwzględnieniu ich specyfiki.

Prezentowane w literaturze zagadnienia związane z problemami realizacyjnymi projektów informatycznych uwzględniają punkt widzenia odbiorców. W artykule przedstawiono w tym samym kontekście punkt widzenia dostawców oprogramowania, jakimi są zespoły rozwijające oprogramowanie, a także punkt widzenia wspólny dla odbiorców i dostawców oprogramowania. W tym obszarze trudno znaleźć publikowane prace naukowe.

Literatura

Amber Team, 2015, *Zarządzanie testami – AmberPlace*, Retrieved February 11, 2016, from http://amberplace.amberteam.pl/Zarz%C4%85dzanie_testami (11.02.2016).

Diagram Pareto, 2015, http://www.governica.com/Diagram_Pareto (11.02.2016).

Dlaczego oprogramowanie zawiera defekty?, 2014, http://www.testowanie.net/testowanie/dlaczego-oprogramowanie-zawiera-defekty/ (1.010.2014).

Eveleens J.L., Verhoef C., 2010, *The rise and fall of the Chaos report figures*, IEEE Software, 27(1), 30-36, http://doi.org/10.1109/MS.2009.154.

Hamrol A., 2005, *Zarządzanie jakością z przykładami*, Warszawa, Wydawnictwo Naukowe PWN.

Hamrol A., Mantura W., 2012, *Zarządzanie jakością: teoria i praktyka*, Wydawnictwo Naukowe PWN, Warszawa.

Holvitie J., Leppanen V., Hyrynsalmi S., 2014, *Technical Debt and the Effect of Agile Software Development Practices on It − An Industry Practitioner Survey*, Sixth International Workshop on Managing Technical Debt (MTD), s. 35-42, http://doi.org/10.1109/MTD.2014.8.

Ishikawa K., 1986, *Guide to Quality Control*, 2nd edition, Quality Resources.

Jayathilake D., Yaggahavita H., Senanayake U., Elvitigala C., Sriyananda D., 2011, *A scalable product quality verifier framework for a outsourcing supplier*, 2011 IEEE International Conference on Computer Applications and Industrial Electronics (ICCAIE), s. 390-395, http://doi.org/10.1109/ICCAIE.2011.6162166.

Kumar S.A., Thangavelu A.K., 2013, *Factors affecting the outcome of global software development projects: An empirical study*, 2013 International Conference on Computer Communication and Informatics (ICCCI), s. 1-10, http://doi.org/10.1109/ICCCI.2013.6466113.

Lavallée M., Robillard P.N., 2012, *The impacts of software process improvement on developers: A systematic review*, 2012 34th International Conference on Software Engineering (ICSE), s. 113-122, http://doi.org/10.1109/ICSE.2012.6227201.

Mach P., Guaqueta J., 2001, *Utilization of the seven Ishikawa tools (old tools) in the six sigma strategy*, 24th International Spring Seminar on Electronics Technology: Concurrent Engineering in Electronic Packaging, s. 51-55, http://doi.org/10.1109/ISSE.2001.931009.

Oyetoyan T.D., Conradi R., Cruzes D.S., 2013, *A Comparison of Different Defect Measures to Identify Defect-Prone Components*, 2013 Joint Conference of the 23rd International Workshop on Software Measurement and the 2013 Eighth International Conference on Software Process and Product Measurement (IWSM-MENSURA), s. 181-190, http://doi.org/10.1109/IWSM-Mensura.2013.34.

Port D., Kazman R., Nakao H., Katahira M., 2007, *Practicing What is Preached: 80-20 Rules for Strategic IV&V Assessment,* IEEE International Conference on Exploring Quantifiable IT Yields, 2007. EQUITY '07, s. 45-54, http://doi.org/10.1109/EQUITY.2007.13.

The Standish Group, 2013, *CHAOS Manifesto 2013, Think Big, Act Small*, www.standishgroup.com.

The Standish Group, 2014, *CHAOS, 2014, The Standish Group Report, Project Smart*, www.standishgroup.com.

Tholons, 2015, *2015 Top 100 Outsourcing Destinations*, www.THOLONS.com.

Tholons_Whitepaper_December_2014.pdf. (n.d.), http://www.tholons.com/nl_pdf/Tholons_Whitepaper_December_2014.pdf (11.02.2016).

Vilfredo Pareto, *Wikipedia, wolna encyklopedia*, https://pl.wikipedia.org/w/index.php?title=Vilfredo_Pareto&oldid=42091738 (16.03.2015).

Wachnik B., 2013, *Analiza przedsięwzięć informatycznych w modelach budowania wartości przedsiębiorstw. Podsumowanie badań z lat 2011-2012*, Informatyka Ekonomiczna, nr 4(30), Wydawnictwo Uniwersytetu Ekonomicznego we Wrocławiu.

Werewka J., 2015, *Investigation of enterprise architecture and software architecture in relation to quality attributes in military applications*, Fast Tracked Vehicles, 37(2), http://www.obrum.gliwice.pl/spg/215/01_Werewka_ang.pdf.

Werewka J., Jamróz K., Pitulej D., 2014, *Developing Lean Architecture Governance at a Software Developing Company Applying ArchiMate* M*otivation and Business Layers,* [w:] Kozielski S., Mrozek D., Kasprowski P., Małysiak-Mrozek B., Kostrzewa D. (red.), *Beyond Databases, Architectures, and Structures*, vol. 424, s. 492-503, Springer International Publishing, http://link.springer.com/10.1007/978-3-319-06932-6_48.

Źródła błędów w projektach programistycznych, http://blog.wsoczynski.pl/2010/05/02/zrodla-bledow-w-projektach-programistycznych/(2.05.2010).