Quality - aplikacja wspomagająca systemowe zarządzanie jakością

Rafał Piotrowski (*rpiotrowski@matrix.pl)*
PJWSTK

Sławomir Babecki (slawomir.babecki@rcb-warsaw.raiffeisen.at)
PJWSTK

Ewa Stemposz (ewag@ipipan.waw.pl)
IPIPAN, PJWSTK

Kazimierz Subieta (subieta@ipipan.waw.pl)
IPIPAN, PJWSTK

Streszczenie

Artykuł - stanowiący element pracy magisterskiej [10] - dotyczy problematyki zarządzania jakością w projekcie informatycznym. Jakość produktu, w terminologii ISO 9000, jest rozumiana jako zbiór tych cech produktu, które decydują o zaspokojeniu potrzeb użytkownika - w kontekście projektu informatycznego, to cechy takie, jak np: poprawność, odpowiednio wysokie użyteczność, niezawodność, wydajność, itp. Zarządzanie jakością obejmuje zespół działań podejmowanych w trakcie realizacji projektu, zmierzających do uzyskania przez produkt pożądanej wysokiej jakości.

1. Wprowadzenie

Wysoka *jakość*, w terminologii ISO 9000, oznacza osiągnięcie przez produkt takiego stopnia doskonałości, że zaspokaja on stwierdzone lub przewidywane potrzeby użytkownika. Zgodnie z [2] istnieje kilka podstawowych własności, które powinny posiadać produkty informatyczne, by można było

przypisać im etykietę wysokiej jakości. Należą do nich: poprawność, użyteczność, niezawodność oraz wydajność.

Z pojęciem „jakość oprogramowania" nierozerwalnie wiąże się pojęcie *zapewnienie jakości*. Zapewnienie jakości jest rozumiane jako zespół działań zmierzających do wytworzenia u wszystkich zainteresowanych przekonania, że produkt posiada wymienione powyżej własności. Sposobem na realizację zapewnienia jakości jest *zarządzanie jakością*, które może być przeprowadzane na wiele sposobów. Ważne jest, by nie utożsamiać zarządzania jakością wyłącznie z testowaniem, które w wielu przypadkach odbywa się dopiero po zakończeniu wszystkich prac fazy implementacji, a nie po zakończeniu każdego z kolejnych etapów projektu. Ponadto, zarządzanie jakością nie powinno być powiązane jedynie z procesem produkcji – nie wolno zapominać o czynnościach pomocniczych, takich jak np: zakupy, kontrakty, polityka kadrowa czy finanse. Z kolei, *system jakości* to struktura organizacyjna, z jednoznacznym podziałem odpowiedzialności, określeniem procesów, procedur i zasobów, umożliwiająca wdrożenie zarządzania jakością.

Systemowe zarządzanie jakością można wdrażać w oparciu o model zarządzania jakością, jak np.: metodę TQM, zbiór norm ISO 9000, system SPICE, model CMM, metodykę BOOTSTRAP czy metodykę PRINCE2.

TQM (Total Quality Management), będąc uniwersalną metodą podnoszenia sprawności zespołów ludzkich poprzez zmianę sposobu myślenia, może być stosowana z powodzeniem zarówno w informatyce, jak i np. w budownictwie. TQM skupia uwagę na kliencie/odbiorcy, w oparciu o założenie, że jakość produktu lub usługi jest elementem wzajemnych relacji klient-dostawca. Obowiązkiem dostawcy jest nie tylko dostarczenie produktu (tu: systemu informatycznego), wypełniajacego oczekiwania klienta, ale także wyprzedzanie tych oczekiwań poprzez oferowanie nowych, innowacyjnych rozwiązań. TQM wymusza aktywność pracowników wszystkich szczebli organizacji, w której jest wdrażana.

ISO 9000 [9] to zbiór 17 różnych norm wprowadzonych w 1987 r., dotyczących realizacji zapewnienia jakości. 7 z nich to standardy bezpośrednio dotyczące problematyki zarządzania jakością w procesie wytwórczym oprogramowania i obejmujące: politykę, organizację, zakresy odpowiedzialności, przeglądy, inspekcje i audyty - wszelkie działania kontrolne służące osiągnięciu przez produkt informatyczny pożądanej wysokiej jakości.

CMM (Capability Maturity Model) - Model Dojrzałości Procesów Wytwórczych) [3],[9] - pozwala na wyspecyfikowanie zbioru cech określających możliwości firmy w zakresie podejmowania i prowadzenia z sukcesem przedsięwzięć informatycznych. CMM określa kryteria, których spełnienie jest niezbędne do osiągnięcia danego poziomu dojrzałości, np. poziom piąty – najwyższy - świadczy o tym, że zdolność do efektywnego i skutecznego

wytwarzania oprogramowania jest cechą organizacji, a nie ludzi, że procesy wytwórcze są zdefiniowane, obserwowane i systematycznie ulepszane.

Metodyka BOOTSTRAP [4] jest połączeniem doświadczeń pochodzących ze stosowania norm ISO serii 9000 i modelu CMM. **BOOTSTRAP** oparta została o szacowanie możliwości procesów realizowanych w przedsiębiorstwie, gdzie możliwość procesu rozumiana jest jako zdolność do osiągania założonych celów, w kontekście celów przedsiębiorstwa jako całości.

System SPICE [7] wyróżnia pięć podstawowych kategorii procesów, bazując na kategoriach procesów metodyki BOOTSTRAP. W przeciwieństwie do modelu CMM, SPICE nie przeprowadza oceny na poziomie całego przedsiębiorstwa i nie jest zorientowany na poprawę jakości pracy przedsiębiorstwa jako całości, ale na poprawę jakości pojedynczych procesów programowych (pośrednio wpływając na całość).

PRINCE2 (**PR**ojects **IN** **C**ontrolled **E**nvironment) to metodyka efektywnego zarządzania projektami, bazująca na podejściu procesowym - projekt to zbiór procesów, dla których określane są dane wejściowe, dane wyjściowe, cele do osiągnięcia i czynności niezbędne do wykonania w obrębie założonych celów. PRINCE2 wyróżnia sześć podstawowych procesów sterowanych przez zarząd przedsięwzięcia: formowanie projektu, inicjacja projektu, faza sterowania, faza zarządzania, zarządzanie dostawami oraz finalizacja projektu.

MS Project – przykład komercyjnego wdrożenia systemowego zarządzania jakością – może znaleźć zastosowanie w realizacji różnego rodzaju przedsięwzięć, niekoniecznie informatycznych. MS Project wspomaga zarządzanie przebiegiem realizacji projektu. Można tu rejestrować dowolną liczbę etapów o dowolnej strukturze z uwzględnieniem czasu wykonania czynności w obrębie każdego z etapów i zasobów niezbędnych do ich realizacji. Można monitorować status wykonania poszczególnych zadań oraz zajętość zasobów zaangażowanych w projekcie.

Zarządzanie jakością można także wdrażać przy pomocy dedykowanego oprogramowania. Quality, aplikacja do systemowego zarządzania jakością, jest przykładem zastosowania zaleceń zawartych w przedstawionych powyżej modelach.

W dalszej części artykułu przedstawiono: funkcjonalność i model pojęciowy oraz proponowaną architekturę dla aplikacji Quality.

2. Quality - aplikacja wspomagająca systemowe zarządzanie jakością

2.1. Cel

Quality, przeznaczona dla przedsiębiorstw konstruujących oprogramowanie, ma umożliwić prowadzenie z sukcesem przedsięwzięć polegających na budowaniu, rozwijaniu i dostarczaniu produktów informatycznych o wysokiej jakości, a w szczególności ułatwiać:

- przechowywanie informacji o realizowanych przedsięwzięciach informatycznych (projektach),
- realizowanie projektów informatycznych zgodnie z wybranymi przed realizacją wzorcami,
- zarządzanie zespołami projektowymi i czynnościami realizacyjnymi,
- kontrolowanie i monitorowanie przebiegu realizacji przedsięwzięć,
- tworzenie repozytorium ponownego użycia dla elementów wchodzących w skład wzorców projektowych,
- systematyczne doskonalenie procesu tworzenia oprogramowania.

2.2. Opis systemu

W zamyśle, Quality ma wspomagać zarządzanie jakością w procesie wytwarzania oprogramowania, co powinno umożliwić systematyzację związanych z tym prac i w efekcie zaowocować wzrostem jakości produktu końcowego. Wzorując się na omówionych powyżej modelach jakości, gdzie poziom dojrzałości przedsiębiorstwa jest charakteryzowany poprzez dojrzałość jego procesów wytwórczych, w Quality wprowadzono pojęcie wzorca projektu. Wzorzec projektu został tu oparty o tradycyjne podejście w realizacji projektów informatycznych, czyli o model kaskadowy. Na projekt składa się pewna liczba etapów. Każdy etap zawiera pewną liczbę zadań do wykonania; zadanie stanowi tu odpowiednik procesu istniejącego w omawianych powyżej modelach. Kierownik projektu, przed przystąpieniem do realizacji projektu, buduje model projektu, co w dużym uproszczeniu oznacza: dzieli całość prac na etapy, zadania, określa potrzebne w danym momencie zasoby i ustala harmonogram oraz budżet. W tym momencie może skorzystać z gotowych wzorców projektów, przechowywanych w repozytorium Quality. Może komponować je w dowolny sposób tworząc rozmaite warianty bazowego modelu kaskadowego, np.

kaskadowy z powrotami, z prototypowaniem, spiralny, itp. Stosując wzorce, kierownik projektu wie jakich zasobów, np. pracowników o pewnych umiejętnościach będzie potrzebował do wykonania konkretnych zadań. W trakcie realizacji projektu, członkowie zespołów projektowych są zobowiązani do rejestrowania przebiegu zadań projektowych bazując na wzorcach dla produktów (powiązanych z wzorcami odpowiednich zadań), oczywiście w założeniu, że model projektu skonstruowano w oparciu o nie, co nie zawsze musi mieć miejsce. Wzorcami dla produktów mogą być np. standardy na dokumenty (czy wręcz formularze dokumentów) oraz standardy na oprogramowanie. Kontroler jakości (osoba nie wchodząca w skład zespołu projektowego), a także kierownik projektu, niezależnie od siebie, kontrolują przebieg realizacji projektu na zgodność z założonym modelem zarówno w aspekcie wyników, jak również i w aspekcie parametrów krytycznych zawartych np. w harmonogramie. Wyniki przeprowadzanych kontroli są rejestrowane i służą zarówno do rozliczania zadań projektowych, jak i w przyszłości do podnoszenia poziomu dojrzałości procesów wytwórczych w przedsiębiorstwie jako całości. Ważną funkcją kontrolera jakości w systemie Quality jest definicja nowych i modyfikacja już istniejących wzorców projektów, w tym wzorców poszczególnych etapów, ich zadań oraz wytwarzanych przez nie produktów. Bardziej szczegółowe informacje na temat funkcjonalności zaprojektowanego systemu zostaną przedstawione w kolejnych punktach.

2.3. Użytkownicy systemu

Projekt systemu zakłada, iż użytkownikami mogliby tu być ci wszyscy pracownicy przedsiębiorstwa, którzy uczestniczą w realizacji projektów informatycznych, a w szczególności: kierownicy projektów, kontrolerzy jakości, członkowie zespołów projektowych, członkowie pionu finansowego.

2.4. Wymagania funkcjonalne

Quality, jako aplikacja wspomagająca systemowe zarządzanie jakością umożliwia:

Użytkownik	**Funkcjonalność**
Kierownik projektu	definiowanie modelu projektu informatycznego w oparciu o gotowe wzorce zawarte w repozytorium (kierownik projektu na dowolnym etapie realizacji, ma prawo do rezygnacji z opierania się o wybrane na początku wzorce)

	tworzenie harmonogramu i budżetu projektu w oparciu o zdefiniowany model
	zarządzanie zespołami projektowymi realizującymi przedsięwzięcia informatyczne
	ocenę czynności realizowanych w ramach projektu poprzez kierowników zespołów projektowych i kontrolerów jakości (ocena dotyczy zgodności z założonymi wcześniej wzorcami, jak również parametrami krytycznymi projektu)
	monitorowanie przebiegu realizacji oprogramowania
	monitorowanie zajętości zasobów ludzkich
	kontrolę odpowiedzialności za realizowane czynności
	zarządzanie zmianami w budżecie i harmonogramie poszczególnych projektów i etapów w nich zawartych
	zarządzanie kosztami realizacji projektów
	analizę ryzyka (definicja zagrożeń, czynności prewencyjnych oraz czynności naprawczych)
	pomiar oprogramowania
	definicję atrybutów jakości wykorzystywanych podczas pomiarów oprogramowania
Kontroler jakości	ocenę czynności realizowanych w ramach projektu poprzez kierowników zespołów projektowych i kontrolerów jakości (ocena dotyczy zgodności z założonymi wcześniej wzorcami, jak również parametrami krytycznymi projektu)
	pomiar oprogramowania
Członek pionu finansowego	zarządzanie zmianami w budżecie i harmonogramie poszczególnych projektów i etapów w nich zawartych
	definiowanie wzorców projektów (wzorzec projektu to struktura, na którą składają się etapy, zadania z uwzględnieniem potrzebnych zasobów, wzorce dokumentów, standardy, informacje dotyczące umiejętności pracowników niezbędnych do realizacji czynności w projekcie, itp.)
Członek zespołu projektowego	zarządzanie konfiguracją produktów stanowiących efekt prac na poszczególnych poziomach realizacji projektu takich, jak np.: dokumenty, kody programów, komponenty ponownego użycia

2.5. Model pojęciowy

Model pojęciowy (patrz rys. 1) dla Quality został opracowany przy użyciu obiektowego podejścia w modelowaniu danych z wykorzystaniem języka UML.

Do głównych klas modelu należą:

Projekt. Klasa specyfikuje projekty informatyczne prowadzone aktualnie lub w przeszłości przez przedsiębiorstwo. Lista właściwości projektu zawiera m. in.: nazwę projektu, opis, planowane i faktyczne daty rozpoczęcia i zakończenia realizacji projektu. Klasa Projekt powiązana jest z klasą Etap, Budżet, Pracownik.

Etap. Przechowuje informacje związane z etapami projektów. Przechowywane własności to m. in.: nazwa etapu, status, priorytet, zaplanowana data rozpoczęcia i zakończenia realizacji. Klasa Etap jest połączona związkiem agregacji z klasą Projekt i Zadanie. Istnieje asocjacja łącząca klasę Etap z klasą Pracownik, Budżet, Wzorzec.

Zadanie. Klasa Zadanie zawiera informacje związane z czynnościami realizowanymi w ramach konkretnego etapu projektu np.: opis czynności w języku naturalnym, produkty we/wy, zasoby potrzebne do realizacji (ludzie, sprzęt, narzędzia, czas i pieniądze), itp.

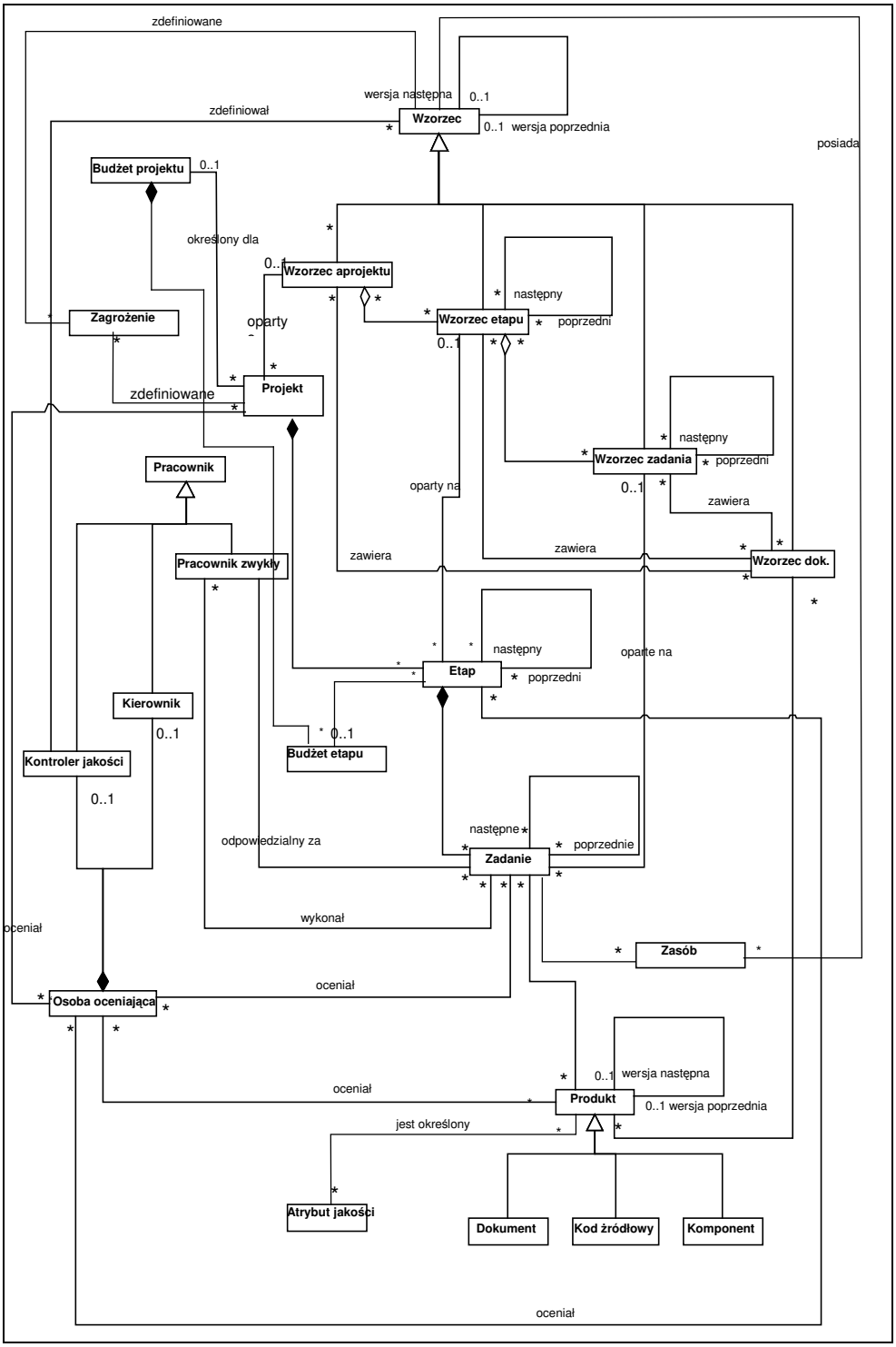
Pracownik. Ta klasa odpowiada za przechowywanie informacji o pracownikach, związanych z realizacją prowadzonych w przedsiębiorstwie projektów. Klasa Pracownik specjalizowana jest przez następujące podklasy: Pracownik zwykły, Kontroler jakości, Kierownik, Członek pionu finansowego. Asocjacja istniejąca pomiędzy klasami Pracownik i Zadanie umożliwia rejestrację informacji zarówno o przebiegu zadań przydzielonych w trakcie prac projektowych, jak i o osobach odpowiedzialnych za ich realizację.

Wzorzec. Specjalizację tej klasy stanowią: Wzorzec projektu, Wzorzec etapu, Wzorzec zadania, Wzorzec dokumentu. Informacje tu zawarte służą do oceny zadań i etapów, wchodzących w skład projektu.

Budżet. Klasa Budżet służy do przechowywania danych dotyczących budżetów poszczególnych projektów i ich etapów.

Produkt. Klasa Produkt specjalizowana jest przez następujące podklasy: Dokument, Kod źródłowy programu, Aplikacja oraz Komponent i przechowuje informacje związane z efektami realizacji zadań. Dzięki zastosowaniu struktury hierarchicznej w obrębie klasy Produkt, istnieje możliwość zbierania informacji odnoszących się do kolejnych wersji produktu (zarządzanie konfiguracją).

Zagrożenie. Klasa przechowuje informacje dotyczące zagrożeń zwiększających ryzyko niepowodzenia przedsięwzięcia np. miejsca przekroczenia terminów, budżetu itp.



Rysunek 1 Diagram klas

Pomiar. Klasa przechowuje informacje dotyczące dokonanych pomiarów oprogramowania w trakcie realizacji przedsięwzięcia.

Atrybuty jakości. Klasa przechowuje definicje atrybutów jakości wykorzystywanych podczas pomiaru oprogramowania.

2.6. Proponowana architektura dla Quality

Architektura proponowana dla Quality to architektura trójwarstwowa, na którą składają się: warstwa klienta, warstwa biznesowa i warstwa bazy danych.

W celu zwiększenia dostępności systemu Quality dla przyszłych użytkowników jako warstwę klienta wybrano przeglądarkę internetową. Zastosowanie przeglądarki internetowej zapewnia, że system mógłby być wykorzystywany zarówno w lokalnych sieciach tych przedsiębiorstw produkujących oprogramowanie, które zdecydowałyby się skorzystać z pomysłu, jak i poprzez sieć Internet.

W skład warstwy biznesowej wchodzi serwer WWW (np. Apache lub Microsoft Internet Information Server) oraz aplikacja, która może być napisana przy pomocy jednego z języków skryptowych uruchamianych po stronie serwera. Proponuje się wykorzystać język PHP 4.0.5 (dla serwera Apache) lub Active Server Pages (dla serwera MS IIS).

Dla implementacji warstwy bazy danych można zastosować albo Oracle 8i/9i albo Microsoft SQL Server 7.0/2000. Dostęp do danych z poziomu warstwy biznesowej mógłby odbywać się poprzez interfejs ADO w przypadku stosowania technologii ASP, a w przypadku języka PHP poprzez specjalne biblioteki wbudowane (ang. extensions).

Proponuje się scentralizowany model serwera, tzn. warstwa serwera bazy danych i biznesowa umiejscowiona na jednym serwerze działającym pod kontrolą systemu operacyjnego Microsoft Windows NT 4.0 lub Microsoft Windows 2000.

3. Podsumowanie

Prezentowany projekt nie dotyczy aplikacji, której głównym zadaniem jest wyłącznie archiwizacja informacji o przebiegu realizacji przedsięwzięcia informatycznego. Wykorzystanie Quality wymusza na użytkowniku pewną systematyczność poprzez dostarczenie mu możliwości używania gotowych wzorców projektów, co w ostateczności powinno wpłynąć na jakość produktu

końcowego. Ponadto, Quality pozwala na przeprowadzanie kontroli na zgodność z przyjętymi wzorcami, a także dostarcza możliwość „nauki na błędach", czyli ciągłego doskonalenia procesów wytwórczych - poprzez ulepszanie wzorców - dzięki wykorzystywaniu doświadczenia nabytego w trakcie realizowania kolejnych projektów.

Podsumowywując, aplikacja Quality, jest próbą ustanowienia standardów wytwarzania oprogramowania, które mogą być wykorzystywane w procesie realizacji przedsięwzięć informatycznych. Ponadto, stanowi też próbę wyszukania informacji, których pomiar i kontrola może usprawnić zarządzanie jakością. Na dzień dzisiejszy, Quality nie została wyposażona w narzędzia ułatwiające przeprowadzanie analizy danych przechowywanych w repozytorium. Wydaje się, że zbudowanie (lub wykorzystanie już istniejących na rynku) dedykowanych narzędzi analitycznych może stanowić element przyszłego rozwoju aplikacji

BIBLIOGRAFIA

[1]. B. Boehm. Software Engineering Economics, Prentice Hall, 1981

[2]. IEEE Standard Glossary of Software Engineering Terminology, ANSI/IEEE Std 610.12-1990

[3]. Górski J.: „Inżynieria oprogramowania w projekcie informatycznym", MIKOM Warszawa 2000

[4]. http://www.cern.ch/PTGroup/spi/process_ass/bootstrap.html

[5]. Capability Maturity Model for Software. http://www.sei.cmu.edu/cmm

[6]. C.Mazza, J. Fairclough, B.Melton, D. de Pablo, A. Scheffer, R. Stevens: "Software Engineering Standards", Prentice Hall, 1994

[7]. Byzia T.: "SPICE – standard oceny i doskonalenia procesów produkcji oprogramowania", w: Materiały Dziesiątej Jubileuszowej Górskiej Szkoły PTI, Szczyrk, 22-26.06.1998, str. 151-156

[8]. IEEE Standards Software Engineering Collection. IEEE-SA, 1999

[9]. Ian Sommerville, Pete Sawyer: „Requirements Engineering – A Good Practise Guide", John Wiley & Sons, 1999

[10]. Piotrowski R., Babecki S., "Systemowe zarządzanie jakością w cyklu wytwarzania oprogramowania" – praca magisterska