Magazine

Zapewnianie jakości w projekcie informatycznym

Autor: Rafał Dobrosielski

O autorze:
Absolwent Politechniki Gdańskiej, wydziału Elektroniki Telekomunikacji i Informatyki, ukończył również studia podyplomowe z zakresu Inżynierii Oprogramowania i Zarządzania Projektami. Zawodowo, na co dzień, zajmuje się czynnie planowaniem, zapewnianiem i sprawdzaniem jakości w projektach informatycznych. Jest autorem szkoleń i warsztatów z zakresu zarządzania projektem informatycznym, zarządzania i zapewniania jakości produktów informatycznych oraz modeli procesów inżynierii oprogramowania.
Interesuje się analizą i modelowaniem procesów biznesowych w organizacjach jak również psychologią biznesu.
Email: rdobrosielski@paop.com.pl

Intermediate
Level

3
Magazine Number

Jakość w projekcie
Section in the magazine

Wprowadzenie
Co jest dobre, Fedrusie?
A co nie jest?
Czy powinniśmy kogoś pytać by się tego dowiedzieć?
(Platon, około 370 BC)

Zapewnianie jakości produktów na swoją historię i jak dowodzi powyższy cytat, sięga czasów bardzo odległych.

Przez wiele lat metody i systemy zapewniania jakości przeszły długą drogę i podlegały wielu zmianom. Ich największy rozwój przypada jednak na drugą połowę ubiegłego stulecia. W tym czasie obserwowaliśmy również ogromny wzrost konkurencji na rynku produktów informatycznych, a także wzrost oczekiwań w stosunku do ich jakości i szeroko rozumianej użyteczności.

Dziś od systemów informatycznych nie wymaga się już tylko prostej niezawodności czy „jakiejkolwiek" realizacji poszczególnych przepadków biznesowych za pomocą komputera.

W poprzednim numerze magazynu c0re, w artykule „Jakość produktów informatycznych", przedstawiony został model jakości systemu informatycznego, który podkreśla jej wielowymiarowość, a sukces projektu informatycznego coraz częściej mierzony jest osiąganiem oczekiwanych korzyści biznesowych przez organizację decydującą się zakupić i wdrożyć system informatyczny.

W związku z tym, rola i zadania procesów zapewniania jakości w projekcie informatycznym podlegały i podlegają nadal ciągłej ewolucji. To na nich spoczywa odpowiedzialność za zbudowanie modelu jakości (indywidualnie dla organizacji, projektu czy dla każdego z jego produktów i podproduktów), wytworzenie systemu konkurencyjnego, intuicyjnego, spełniającego wymagania szerokiego spektrum interesariuszy.

Coraz częściej członkowie zespołów projektowych, którzy są odpowiedzialni za jakość, uczestniczą w działaniach marketingowych, biorąc odpowiedzialność za prawidłowe odczytanie potrzeb klientów i optymalne określenie zakresu systemu – szacując wartość dodaną płynącą z jego wdrożenia, koszty i czas wytworzenia. Wszystko po to, aby trafić z produktem w okres największego zapotrzebowania na rynku, co przyniesie klientowi i dostawcy największe zyski. Z ostatnim zadaniem wiąże się również oszacowanie, czy produkt osiągnął już wystarczająco wysoką jakość, a dalsze prace nad jej podnoszeniem nie będą dla produktu destrukcyjne lub czy nie spowodują znacznych opóźnień we wdrożeniu, co z kolei może prowadzić do strat potencjalnych korzyści finansowych czy też utraty udziałów w rynku.

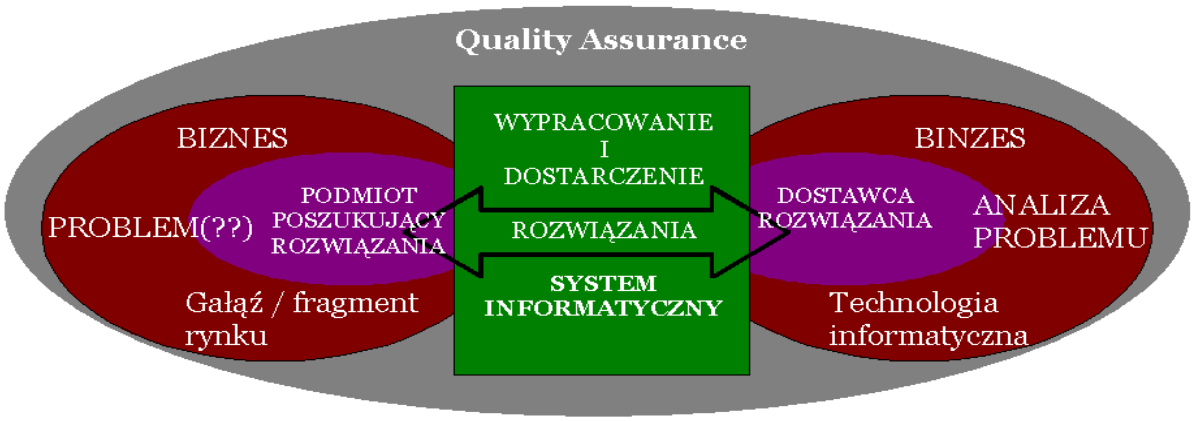
Powyższe rozważania dowodzą, że zapewnianie jakości w projekcie stało się już dziedziną interdyscyplinarną.

Kto odpowiada zatem za jakość systemu informatycznego?

Jednym z zasadniczych błędów w procesie zapewniania jakości jest (niestety) często spotykane w praktyce twierdzenie, że w ramach projektu informatycznego należy wyróżnić grupę osób odpowiedzialnych za jakość produktów czy procesów. Ten mit jest propagowany i uwieczniany poprzez powoływanie do życia osobnego zespołu, zwykle zwango Software Quality Assurance (SQA) i oddanie mu „przywileju" i odpowiedzialności za jakość.

W praktyce jakość jest i powinna być odpowiedzialnością każdego. Osiąganie wysokiej jakości musi być zintegrowane z wszystkimi czynnościami procesów zachodzących w projekcie, a nie stanowić osobną dyscyplinę. Takie podejście czyni każdego odpowiedzialnym za jakość produktów, nad którymi pracuje i za pieczołowite wykonywanie czynności w ramach procesów, w które jest zaangażowany [12].

Pomimo, że każdy ponosi odpowiedzialność za jakość, formalnie, za zapewnienie jakości w projekcie odpowiedzialny jest Kierownik Projektu. Dzięki sprawowaniu przez niego odpowiedniego nadzoru oraz



ustanawianiu odpowiednich priorytetów możemy mieć pewność, że jakość będzie odpowiednio zarządzana, mierzona i osiągana.

Zapewnianie jakości – podejście tradycyjne a podejście współczesne

Współczesne podejście do zapewniania jakości projektu informatycznego znacznie się zmieniło i nieustannie ewoluuje w kierunku osiągania zapewnienia satysfakcji klienta.

Początkowo projekty informatyczne rozumiały osiąganie jakości produktu informatycznego jako zbudowanie systemu zgodnego z przyjętymi uprzednio wymaganiami. Jakość produktu oceniana była głównie przez producenta czy też ekspertów - specjalistów z dziedziny informatyki - a wady traktowano jedynie jako odstępstwa od specyfikacji. Dążono do pozyskania i opracowania mierzalnych i obiektywnych kryteriów jakości.

Nadrzędną cechą jakości jest jej dynamika i subiektywizm. Dlatego obecnie jakość rozumie się jako stopień, w jakim system czy produkt informatyczny spełnia oczekiwania klienta. Najcenniejsze poglądy o jakości systemu informatycznego formułuje klient zamawiający system oraz jego użytkownicy.

Obecne systemy zapewniania jakości oparte są na analizie celów biznesowych, analizie wymagań, badaniach konkurencji i badaniach marketingowych oraz na ocenie zadowolenia klientów.

Tabela nr 1 przedstawia porównanie „tradycyjnego” i „nowoczesnego” podejścia do pojęcia jakości w informatyce.

Podejście „tradycyjne”	Podejście współczesne

• Jakość to
  ○ Zgodność produktu z wymaganiami
• Jakość ocenia
  ○ Producent, ekspert
• Co to jest wada?
  ○ Odstępstwo od specyfikacji
• Kryteria jakości
  ○ Obiektywne i mierzalne
  ○ Dotyczą atrybutów, charakterystyk i parametrów technicznych
• System zapewnienia jakości
  ○ Analiza wymagań, zdefiniowanie i kontrola procesu wytwarzania, standaryzacja, dokumentacja
	• Jakość to
	  ○ Stopień spełnienia oczekiwań użytkownika (klienta)
	• Jakość ocenia
	  ○ Klient, użytkownik
	• Co to jest wada?
	  ○ Niespełnienie oczekiwań użytkownika
	• Kryteria jakości
	  ○ Subiektywne oceny użytkownika
	  ○ Stopień satysfakcji użytkownika
	• System zapewnienia jakości
	  ○ Analiza celów biznesowych, analiza wymagań, badania konkurencji i marketingowe, ocena zadowolenia klientów

Tabela 1, Podejście „tradycyjne” a „nowoczesne” do pojęcia jakości w informatyce

Podejście tradycyjne nastawione było na wykrywanie defektów, natomiast zapewnianie jakości, często rozumiane jako testowanie, przypisywano specjalnie wydzielanym komórkom organizacyjnym. Dziś już wiadomo, że wytwarzanie systemów wysokiej jakości musi być okupione wysiłkiem niemalże całego przedsiębiorstwa. Procesy zapewniania jakości obejmują procesy wytwórcze, produkty (głównie cząstkowe), dokumentacje, ludzi itp. Większość wysiłku powinna być skierowana na zapobieganie defektom i wadom, a nie na wykrywanie ich i usuwanie.

Sprawdzanie jakości a zarządzanie jakością

Systemy jakości również dynamicznie zmieniają się, ewaluowały od tzw. systemów kontroli jakości, poprzez systemy zapewniania jakości do zarządzania nią.

Sprawdzanie jakości traktuje system jak black box, a jakość oprogramowania sprawdzana jest a posteriori. Cały system jakości zbudowany jest wokół idei testowania funkcjonalnego. Jego efektywność, przytaczana w [8], szacowana jest na 70% usuwanych defektów, zatem 30% trafia do klienta. Powoduje to wzrost kosztów pielęgnacji naprawczej, sięgających nawet do 25% kosztów wytworzenia. Jak wiemy, koszt naprawy pojedynczego błędu w „skończonym" systemie jest wysoki (reguła 1:10) a ogólny koszt sprawdzania jakości może sięgać do 50% kosztów wytworzenia. Najistotniejsze w tym podejściu jest to, że inne ważne atrybuty jakości są w tutaj ignorowane, gdyż po prostu nie można ich zbudować poprzez testowanie.

Zarządzanie jakością motywuje do szerokiego oceniania jakości wszystkich produktów cząstkowych, również w trakcie procesu ich wytwarzania. Procesy wytwórcze są zdefiniowane, opomiarowane, a ich realizacja również podlega sprawdzaniu. Procesy te mają zdefiniowaną strukturę, są podzielone na etapy z wynikającymi z nich produktami. Techniki oceny jakości (przeglądy, inspekcje) są w stanie wykryć 80-90% defektów. Dla poszczególnych etapów procesu, jak i dla całego przedsięwzięcia definiowane są zalecane metody i narzędzia, wprowadzane są pomiary procesu i produktu.

Standaryzacja struktury procesu i jego „uzbrojenia" narzędziowego oraz nadanie mu pewnego stopnia powtarzalności, pozwalają wykorzystywać te pomiary do szacowań kluczowych parametrów, takich jak koszt, czas realizacji, końcowa gęstość defektów itp. na podstawie danych historycznych. Umożliwia to kontrolę nad ryzykiem przedsięwzięcia poprzez podejmowanie akcji naprawczych, zanim sytuacja stanie się niebezpieczna.

Zarządzanie jakością w projekcie informatycznym

Cele procesu zarządzania jakością projektu

W projekcie informatycznym przed procesem zarządzania jakością stawiane są następujące cele:

- zidentyfikować i zdefiniować właściwe wskaźniki (metryki) akceptowalnej jakości,
- zidentyfikować, definiować i zaplanować odpowiednie pomiary, które będą użyte w procesie badania i oceny jakości,
- zidentyfikować i odpowiednio rozwiązać zagadnienia i problemy jakościowe tak szybko i efektywnie jak to możliwe.

Zarządzanie jakością jest realizowane poprzez wszystkie dyscypliny, czynności, fazy i iteracje modelu inżynierii oprogramowania. Oznacza to, że należy realizować, mierzyć i oceniać jednocześnie:

- jakość procesu – co jest nastawione na poprawne budowanie produktów,
- jakość produktu – co jest nastawione na zbudowanie poprawnego produktu.

Zarządzania jakością procesu

Jakość procesu odnosi się do stopnia, w jakim przyjęte, zaakceptowane procesy, uwzględniając pomiary i kryteria jakości, są wdrożone i stosowane w celu wytworzenia produktów.

Jak wiemy, wszystkie procesy składają się z czynności produkcyjnych i tzw. czynności wyższego rzędu, zarządczych.

Czynności produkcyjne mają rzeczywisty, namacalny wpływ na produkt końcowy. Czynności pozostałe mają

pośredni wpływ na końcowy produkt, wykonywane są podczas np. planowania, zarządzania i oszacowywaniu zadań.

Zadaniem i efektem mierzenia i zapewniania jakości procesu jest:

- właściwe, efektywne zarządzanie zasobami,
- właściwe zarządzanie ryzykiem i postępowanie z nim,
- zbieranie danych w celu usprawnienia procesów.

Z praktycznego punktu widzenia stosowanie w projekcie informatycznym sprawdzonych rynkowo modelów procesów i zasad oraz osiągniecie przez nie wysokiego poziomu wdrożenia przekłada się na jakość produktów, a ryzyko wyprodukowania produktów niskiej jakości maleje i jest skutecznie redukowane.

Zarządzania jakością produktów

Zarządzanie jakością produktów koncentruje się na wyprodukowaniu właściwych, spełniających wymagania interesariuszy, wszystkich produktów będących celem realizacji projektu informatycznego.

Powyżej uzgodniliśmy, że zapewnianie jakości musi odbywać się we wszystkich dyscyplinach inżynierii oprogramowania.

W ramach tego artykułu skupię się na dwóch, ramowych, kluczowych dyscyplinach – wymagań i testowania, które w tradycyjnym, pierwotnym podejściu stanowiły wejście i wyjście procesu produkcyjnego.

Elementem wielu definicji wysokiej jakości systemu informatycznego jest spełnienie wymagań stawianych systemowi czy oczekiwań interesariuszy/udziałowców.

„Udziałowcem jest niewątpliwie użytkownik końcowy, ale musimy uwzględniać także innych: kupującego, kontrahenta, administratorów, kierownika przedsięwzięcia, czy kogokolwiek innego, kto jest dostatecznie zainteresowany, albo kogo potrzeby muszą być zaspokojone poprzez przedsięwzięcie.”[12]

Dyscyplina wymagań - rozpoznawanie potrzeb udziałowców

Aby wytworzyć system ceniony przez jego odbiorców, zespół twórców musi rozumieć konkretne potrzeby udziałowców i cel biznesowy zleceniodawcy.

„Ważne jest, abyśmy aktywnie zbierali wszystkie typy żądań udziałowców (surowe dane wejściowe, służące wyrażaniu potrzeb udziałowców) w ciągu całego okresu trwania przedsięwzięcia. Na początku możemy korzystać z wywiadów, kwestionariuszy i spotkań roboczych, później będziemy zbierali żądania zmian, raporty o usterkach i żądania rozbudowy. Te żądania będą zwykle niejasne i wieloznaczne i często będą miały postać zapotrzebowania (na przykład „potrzebuję łatwiejszych sposobów dzielenia się moją wiedzą o stanie przedsięwzięcia”, „potrzebne mi zwiększenie mojej wydajności”, „musimy zwiększyć efektywność systemu”). Takie żądania udziałowców tworzą niezbędne warunki do zrozumienia ich rzeczywistych potrzeb i poznawania krytycznych uwag co do wymagań stawianych naszym produktom. A to z kolei stanowi ważną część układanki, która pozwoli nam rozpoznać wszystkie *dlaczego* i *co* dotyczące zachowań systemu.”[12]

W dalszych personalnych czy grupowych dyskusjach należy przekuć poszczególne zapotrzebowania na tzw. właściwości systemu – zachowania na wyższym poziomie, rozumiane jako „usługa, która ma być świadczona przez system, bezpośrednio spełniająca potrzebę użytkownika”[12]. W wyniku takiej dyskusji trzeba zmienić/przesunąć myślenie użytkownika od *co* do *jak.*

Dopiero dogłębne rozumienie potrzeb udziałowców i przekucie ich na właściwości systemu, pozwala nam na rozpoczęcie prac nad kolejnym poziomem szczegółowości tegoż opracowania w celu dokładnego przekazania twórcom tego, co system powinien robić (zamiast „Alu, oprogramuj system do automatycznego zawiadamiania o listach poczty elektronicznej”). Użyteczne staje się tu modelowanie biznesowe i systemowe, za pomocą którego będziemy mogli przetłumaczyć te potrzeby i właściwości na

specyfikację, którą będzie można projektować, implementować i testować. Tu również uaktywnia się proces zapewniania jakości, którego celem jest zbadanie min. spójności, mierzalności i testowalności wymagań.

Powyższe działania w ramach dyscypliny wymagań powinny nam pozwolić zrealizować następujące cele:

- „uzyskać i otrzymać zgodę klientów i innych udziałowców co do tego, **co** system powinien robić i **dlaczego**;
- pomóc twórcom systemu w lepszym zrozumieniu wymagań stawianych systemowi;
- określić granice systemu;
- stworzyć podstawy planowania technicznych treści iteracji (procesu wytwórczego);
- stworzyć podstawy estymacji kosztów i czasu budowy systemu;
- określić interfejs użytkownika dla systemu koncentrując się na potrzebach i celach użytkowników".

Należy przypomnieć, że istotne jest, aby zbierane wymagania, a następnie testy i poszczególne pomiary klasyfikować konsekwentnie do przyjętego modelu jakości i dokonywać ich analizy i syntezy poprzez pryzmat celu biznesowego zleceniodawcy.

Dla przykładu, model klasyfikujący atrybuty jakościowe wg RUP, jak i inne opisywane w literaturze modele zwykle jednakowo dokonują pierwotnego podziału na wymagania funkcjonalne i niefunkcjonalne.

Wymagania funkcjonalne

„Wymagania funkcjonalne służą do opisywania zachowań systemu przez formułowanie warunków dotyczących danych wejściowych, które mają być spełnione.

Wymagania niefunkcjonalne

Aby osiągnąć poziom jakości potrzebny użytkownikowi końcowemu, trzeba wyposażyć system w spory zestaw właściwości nie wymienionych w wymaganiach funkcjonalnych dotyczących tego systemu. Aby system takie właściwości uzyskał, musi spełniać jeszcze dodatkowe wymagania – nazywamy je niefunkcjonalnymi. W ostatecznym rozrachunku są one pod każdym względem równie ważne dla społeczności użytkowników końcowych, jak wymagania funkcjonalne.

Stosując metodę klasyfikacji FURPS, można wyróżnić następujące niefunkcjonalne atrybuty jakości:

- użyteczność – wymagania co do użyteczności dotyczą czynników ludzkich: estetyki, łatwości opanowania, łatwości używania oraz spójności interfejsu użytkownika, dokumentacji użytkownika i materiałów szkoleniowych,
- niezawodność – wymagania co do niezawodności dotyczą częstości błędów i ich dotkliwości, zdolności odnawiania, przewidywalności i dokładności,
- efektywność – wymagania co do efektywności narzucają warunki na wymagania funkcjonalne – na przykład wymaganie, które określa częstość transakcji, szybkość, dostępność, czas reakcji, czas odnowy albo zajętość pamięci, przy której dana czynność będzie musiała być wykonywana,
- łatwość wspierania (pielęgnowalność) – wymagania co do łatwości wspierania dotyczą testowalności, pielęgnowalności i innych cech niezbędnych, aby system zachował zdolność do prawidłowej pracy po oddaniu go do użytku. Te wymagania są unikatowe w tym sensie, że mogą dotyczyć nie tylko systemu, ale także procesu tworzenia systemu albo rozmaitych artefaktów procesów.

Wartość tych definicji polega na tym, że umożliwiają tworzenie szablonów albo list, ułatwiających formułowanie wymagań i ocenę kompletności, a także prawidłowe rozumienie wyników tej pracy. Mówiąc inaczej, jeżeli zbadamy wymagania z wszystkich tych kategorii i je zrozumiemy, to będziemy mieli pewność, że naprawdę podjęliśmy te wymagania, które mają największe znaczenie, zanim jeszcze zaczniemy poważnie inwestować w dane przedsięwzięcie.

„System, który nie spełnia wymagań niezawodności lub efektywności wynikających z innych wymagań, jest

tak samo zły jak system, który nie jest w stanie spełnić jakiegokolwiek zadanego wymagania funkcjonalnego.” [12]

Ocena jakości produktów

Dyscypliną odpowiedzialną za ocenę jakości produktów projektu informatycznego jest **testowanie**. Stosuje je się w różnych celach zależnych od osiągniętego poziomu zaawansowania projektu, jego produktów, włożonego i zaplanowanego wysiłku.
Zarządzanie, mierzenie i ocena jakości produktów i procesów powinna być przeprowadzana przez cały cykl życia projektu i modelu inżynierii oprogramowania.

Ocena jakości może być przeprowadzona kiedy wydarzy się jakieś istotne wydarzenie jak np. nowe, wymagające natychmiastowej reakcji zagadnienie ryzyka, nowe wymaganie lub kiedy zgodnie z przyjętym modelem i planem projektu ukończono w całości lub częściowo produkt podlegający testowaniu (zebrano wymagania, przygotowano przypadki testowe, dokumentację itp.).

Zadaniem testowania nie jest wyłącznie proste wyszukiwanie błędów. Proces oceny jakości powinien zapewnić:

- „wykrywanie i dokumentowanie niedostatków zgodnie z kryteriami wystarczająco dobrej jakości”;
- informowanie członków zespołu o postrzeganej jakości oprogramowania;
- potwierdzenie – konkretnymi dowodami – słuszności założeń poczynionych przy projektowaniu i specyfikowaniu wymagań;
- potwierdzenie, że produkt programowy działa zgodnie z projektem;
- potwierdzenie, że wymagania zostały odpowiednio zaimplementowane. [12]

Zwykle w projektach informatycznych testowanie pochłania od 30-50% łącznych kosztów tworzenia oprogramowania. Niestety, w wielu przypadkach przeważa opinia użytkowników mówiąca o tym, że oprogramowanie w ogólności nie jest dostatecznie przetestowane.

Ponieważ sposoby, poziomy i zakres testowania zależy od wielu czynników (np. typu projektu, zadań, jakie będą wykonywać wdrożone produkty programowe, technologii, użytych narzędzi), w niniejszym artykule nie zostaną przedstawione uniwersalne sposoby przeprowadzania czynności testowych. Poniżej omówione zostały najczęściej spotykane błędy i przyczyny niskiej efektywności procesu testowego, których eliminacja może prowadzić w praktyce do obniżania kosztów testowania i ryzyka wdrożenia oprogramowania niskiej jakości. Zostały one przedstawione w [11].

Przede wszystkim należy pamiętać, że testowanie nie jest pojedynczą czynnością czy etapem działalności, w którym ocenia się jakość. Aby wypełnić cel informowania na czas twórców o osiąganej jakości przez oprogramowanie, testowanie musi rozpoczynać się odpowiednio wcześnie i zarówno dotyczyć wczesnych prototypów, stabilności i sprawności architektury (jądra systemu), gdy jeszcze można ją poprawić jak i produktu końcowego, pozwalającego na ocenę jego gotowości do przekazania klientom.

Zbyt późne rozpoczęcie procesów testowych powoduje zatracenie najważniejszej korzyści płynącej z procesu testowego – szansy na uzyskanie informacji, czy mamy jeszcze czas, budżet i zasoby na to, aby jeszcze cokolwiek z tym produktem zrobić.

Testowalność nie jest uwzględniana w projekcie produktu, w wyniku czego często wielokrotnie wzrasta złożoność testowania, następują utrudnienia w automatyzacji lub niemożność wykonania niektórych testów.

Testowanie jest przeprowadzane bez jasno sformułowanych metod, zasad, wskutek czego wyniki zmieniają się w zależności od przedsięwzięcia, od przedsiębiorstwa, a efektywność testowania zależy głównie od umiejętności, pomysłowości i chwilowej dyspozycyjności członków zespołu testowego.

Narzędzia, od których zależy efektywność są albo używane nieskutecznie albo wcale i dlatego to co wymaga największego wysiłku najczęściej nie zostaje przetestowane. Testy rzadko są prowadzone automatycznie,

najczęściej bez użycia narzędzi, które pozwalają na skuteczne zarządzanie zbiorami danych testowych i wynikami testowania.

Nie jest możliwe całkowite przetestowanie oprogramowania, dlatego strategia testowania musi być dobierana zarówno metodycznie jak i doświadczalnie, a wiedza ta powinna podlegać zarządzaniu i propagowaniu w organizacji.

Bardzo często testowanie ograniczone jest do sprawdzania i oceniania atrybutów funkcjonalnych jakości, pomijając równie ważne atrybuty i kryteria użyteczności, efektywności czy niezawodności. Dla każdego z powyższych wymiarów jakości (czy wymagań) należy zdefiniować (w zależności od typu projektu) odpowiednio spriorytetyzowane przypadki testowe, które będą wykonywane na różnych poziomach procesu testowego [12].

Wyniki testowania powinny być jak najbardziej obiektywne (mimo subiektywności omawianych definicji jakości), poparte odpowiednimi pomiarami według uprzednio uzgodnionych, zatwierdzonych metryk odnoszących się zarówno do produktów jak i procesu.

Podsumowanie

Projekt informatyczny, zarządzanie nim i zarządzanie jakością to nie sekwencja poszczególnych sprawdzonych działań, ale skomplikowana sieć aktywności, które prowadzą do powstania wielu produktów. Transfer wysiłku osób odpowiedzialnych za zapewnianie jakości z testowania na poprawne zarządzanie procesem i czynnościami zazębia się z jakością produktu i bardzo często ogranicza ryzyko powstania produktu złej jakości. Od początku rozwoju informatyki organizacje poszukiwały i poszukują wciąż dobrego przepisu gwarantującego sukces podczas realizacji tego typu projektów. Kontynuując te rozważania, w następnym artykule z cyklu, postawione zostanie pytanie „Co to są modele procesów i czy są one nam potrzebne?

Literatura

[1] James Bach: „Good Enough Quality: Beyond the Buzzword”, IEEE Computer, Sierpień 1997
[2] Anna Bobkowska: „Inżynieria Oprogramowania”, Studium Podyplomowe Nowoczesne Metody Inżynierii Oprogramowania, edycja 2006-2007
[3] Grady Booch: „Leaving Kansas” IEEE Software. 15(1), Styczeń/Luty 1998
[4] Victor R. Basili: „The goal question metric approach”, (Advanced Computer Studies, Department of Computer Science, University Of Maryland)
[5] John Fodeh: „What's on Your Dashboard” Better Software, October 2007
[6] Janusz Górski: „Zarządzanie projektem informatycznym”, Studium Podyplomowe Nowoczesne Metody Inżynierii Oprogramowania, edycja 2006-2007
[7] Janusz Górski: „Inżynieria oprogramowania w projekcie informatycznym”, Zakład Nauczania Informatyki MIKOM, Warszawa 1999
[8] Jerzy Kaczmarek: „Metryki i zapewnianie jakości”, Studium Podyplomowe Nowoczesne Metody Inżynierii Oprogramowania, edycja 2006-2007
[9] Stephen H. Kan: „Metryki i modele w inżynierii jakości oprogramowania”, Wydawnictwo Naukowe PWN SA, 2006
[10] Adam Korczowski: „Zarządzanie ryzykiem w projektach informatycznych. Teoria i praktyka”, Wydawnictwo Helion 2010
[11] Per Kroll, Philippe Kruchten: „Rational Unified Process od strony praktycznej”, Wydawnictwo Naukowo-Techniczne, Warszawa 2007
[12] Philippe Kruchten: „Rational Unified Process od strony teoretycznej”, Wydawnictwo Naukowo-Techniczne, Warszawa 2007
[13] Zdzisław Szyjewski: „Metodyki zarządzania projektami informatycznymi”, Wyd. PLACET, Warszawa 2004