ZESZYTY NAUKOWE UNIWERSYTETU SZCZECIŃSKIEGO

NR 852 EKONOMICZNE PROBLEMY USŁUG NR 117 2015

GRAŻYNA HOŁODNIK-JANCZURA
Politechnika Wrocławska[1]

ITERACYJNY MODEL JAKOŚCI PRODUKTU W PODEJŚCIU AGILE

Streszczenie

W wyniku zmian zachodzących w zarządzaniu projektami informatycznymi, skupionych pod nazwą podejścia Agile, zaproponowano koncepcję definiowania iteracyjnego modelu jakości produktu. Analizując przebieg etapów testowania, sformułowano model przebiegu dwuetapowego TDD, włączającego testowanie akceptacyjne na początek cyklu wytwarzania. Zauważono przy tym możliwość połączenia głosu klienta/użytkownika ze stroną dewelopera, za pomocą zaproponowanego modelu jakości. Na przykładzie wybranych historyjek użytkownika pokazano sposób definiowania tego modelu.
Słowa kluczowe: charakterystyka jakości, kryterium akceptacji, testowanie.

Wprowadzenie

Jakość i koszt stanowią dzisiaj podstawę walki konkurencyjnej na rynku odbiorców i producentów. Szybkie dostarczenie tańszego i bardziej dopasowanego produktu do potrzeb klienta to dla wielu firm szansa na utrzymanie się lub wejście na rynek. Takie oczekiwania muszą się jednak wiązać z czasochłonnym wysiłkiem i zaangażowaniem organizacji w realizację inicjatywy wprowadzenia zmian, np. przejścia z tradycyjnego wytwarzania oprogramowania na Agile. Możliwość poprawy jakości oprogramowania, jako jeden z najważniejszych czynników satysfakcji klienta, stanowi tutaj przesłankę dla proponowanej koncepcji budowy iteracyjnego modelu jakości oprogramowania, od początku procesu wytwarzania.

[1] Wydział Informatyki i Zarządzania, Katedra Badań Operacyjnych, Finansów i Zastosowań Informatyki.

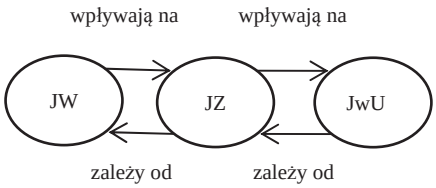
Potencjał poprawy jakości w podejściu zwinnym jest widoczny w zbiorze zasad Agile, jak i zasad Lean Software Development[2] (Poppendieck, s. 19–41), dla których nadrzędną zasadą jest: „wykonuj tylko prace wymagane, dążąc do maksymalnego wyeliminowania wyników, których nikt nie potrzebuje", ujętych w cztery filary podejścia Agile (agilemanifesto 2014):

1. Indywidualizm i interakcje ponad proces i narzędzia.
2. Działający kod ponad wyczerpującą dokumentację.
3. Współpraca z klientem ponad negocjacje umowy.
4. Odpowiedź na zmiany ponad wypełnienie planu.

Zasady Agile to podwyższanie jakości za pomocą zmiany stylu i organizacji pracy, a także zasad zarządzania. Wśród praktyk projakościowych istotne zmiany zauważa się w organizacji procesu testowania, stanowiące źródło proponowanego podejścia do budowy modelu jakości oprogramowania.

1. Zapewnianie jakości oprogramowania

Zapewnianie jakości wymaga skutecznych metod i narzędzi wspomagających. Według standardu jakości serii ISO/IEC 2500n jakość produktu jest opisywana za pomocą dwóch modeli: modelu jakości wewnętrznej (JW) i zewnętrznej (JZ) oraz modelu jakości w użyciu (JwU). Zachodzące między nimi relacje pozwalają na określenie wpływu atrybutów wewnętrznych na zewnętrzne oraz atrybutów jakości zewnętrznej na atrybuty jakości w użyciu. W przypadku planowania i badania jakości oprogramowania mogą one być wykorzystane do przewidywania jakości zewnętrznej na podstawie zmierzonych wartości atrybutów jakości wewnętrznej (rys. 1).



Rys. 1. Cykl relacji między atrybutami poszczególnych części modeli jakości SQuaRE

Źródło: opracowanie własne na podstawie (ISO/IEC TR 9126 – 2, s. 3).

[2] Lean Software Development posiada swoje źródło w koncepcji „odchudzonego wytwarzania produktu" autorstwa Taiichi Ohno i od lat stosowanego w systemie zarządzania w firmie Toyota pod nazwą „Toyota Production System". Wspólne elementy problemów wytwarzania, zarówno produktów materialnych, jak i niematerialnych, do jakich zaliczamy programy komputerowe, stały się podwaliną pomysłu adaptacji odchudzonego zarządzania również w projektach informatycznych.

W modelach jakości oprogramowania o strukturze hierarchicznej, które były podstawą modelu SQuaRE, jak np. model McCalla, Boehma, Boeinga, FURPS, ISO/IEC 9126, przyjęto zasadę określania jakości za pomocą złożonych charakterystyk oraz ich mierzalnych atrybutów, ponieważ jakość nie występuje jako pojedyncza właściwość. Jest wręcz odwrotnie, istnieje wiele czynników jakości, które mogą mieć różne znaczenie dla zainteresowanych jakością, a to pokazuje, jaka jest złożoność tego zagadnienia.

Model SQuaRE, który kompleksowo obejmuje różne aspekty jakości produktów informatycznych, w części dotyczącej jakości produktu informatycznego wskazuje, że miary zewnętrzne i wewnętrzne określają tę samą cechę produktu, ale z dwóch różnych perspektyw: klienta i dewelopera. Przykładowa wewnętrzna miara oczekiwanego czasu odpowiedzi może służyć do predykcji tego czasu, ale mierzonego z perspektywy zewnętrznej. Natomiast miary jakości w użyciu mierzone w relacji do rzeczywiście zakończonych zadań przedstawiają efekty uzyskane przez zastosowanie programu w konkretnym środowisku użytkownika, nazywanym kontekstem użycia.

1.1. Kryteria jakości – kryteria akceptacji klienta i użytkownika

Charakterystyki występujące w modelu jakości pełnią rolę kryteriów sprawdzanych podczas oceny jakości oprogramowania. Lista oczekiwań, od których zależy akceptacja programu przez klienta i użytkownika, najczęściej reprezentowanego przez tzw. głównego użytkownika, odpowiada kryteriom akceptacji. Można znaleźć różne, ale o podobnym znaczeniu definicje kryteriów jakości oprogramowania[3]. Dla podejścia Agile firma Microsoft definiuje kryteria akceptacji jako warunki, które produkt informatyczny musi spełnić, aby być zaakceptowanym przez użytkownika, klienta lub innego interesariusza projektu. Jest to zbiór wyrażeń z klarownym wynikiem sukcesu albo porażki, który specyfikuje zarówno wymagania funkcjonalne, jak i niefunkcjonalne odpowiednio do aktualnego etapu projektu (seguetech 2013).

Podkreśla się, że nie jest możliwe w przypadku dużych produktów mierzenie wszystkich charakterystyk wewnętrznych i zewnętrznych, jak i badanie wszystkich możliwych scenariuszy zadań użytkownika dla badania jakości w użyciu. Decydujące o dostosowaniu modelu jakości powinno być kryterium ważności jego elementów dla klienta, co może okazać się pomocne w przypadku poszukiwania kompromisu między niektórymi z nich, jak np. między wysoką jakością i niskim kosztem.

[3] Kryteria akceptacji wg metodyki PRINCE2 to uporządkowana według priorytetu lista kryteriów, które musi spełnić produkt, aby mógł być zaakceptowany przez klienta i interesariuszy projektu. Kryteria akceptacji wspomagają decyzje o odbiorze produktu, a także o zakończeniu projektu (Bentley 2003).

1.2. Weryfikacja i walidacja

Weryfikacja i walidacja to działania realizowane w ramach zapewniania jakości oprogramowania, polegające na sprawdzaniu programu różnorodnymi metodami i technikami: przeglądy techniczne, audyty, symulacje, przeglądy dokumentacji, danych, analizy algorytmów i testowanie. Walidacja to potwierdzenie, przez dostarczenie dowodu obiektywnego, że zostały spełnione wymagania dotyczące konkretnego użycia lub zastosowania (PN-EN ISO 9000:2001). Walidacja to też odpowiedź na pytanie, czy jest budowany potrzebny produkt. Natomiast weryfikacja, czy produkt jest budowany tak, jak potrzeba – to jest sprawdzenie, czy program odpowiada jego specyfikacji (BOE81, s. 37).

W procesie testowania, podstawowej części procesów weryfikacji i walidacji, wyróżnia się trzy poziomy: testowanie jednostkowe (komponentu, modułu), testowanie integracyjne (podsystemu, systemu), testowanie akceptacyjne. Przy tradycyjnym podejściu do wytwarzania oprogramowania przebiegają one sekwencyjnie, ale i też iteracyjnie, z możliwością powrotu do wcześniejszej części innego rodzaju testowania. W sekwencyjnym modelu V testowanie akceptacyjne jest realizowane przez użytkownika na końcu cyklu wytwarzania (Sommerville, s. 451).

2. Podejście Agile i testowanie

W dążeniu do osiągnięcia satysfakcji klienta w projektach Agile zauważa się istotną dbałość o techniczną doskonałość od samego początku projektu, zgodnie z zasadą „zrób to od razu dobrze". To podejście do jakości wyraża się za pomocą dwóch zasad: „buduj jakość" i „optymalizuj całość" (Poppendieck, s. 25–29, 38–40).

W Agile nie wyróżnia się faz projektu, takich jak specyfikacja wymagań, projektowanie, kodowanie i testowanie. Za pracę i wyniki każdej z faz odpowiada cały zespół. Na pytanie, czy testowaniem zajmuje się cały zespół, odpowiedź jest twierdząca, ponieważ testuje tester, programista, projektant, ale i użytkownik, który też należy do zespołu jako przedstawiciel biznesu, w roli tzw. właściciela produktu (WP). Współpraca zespołu z WP jest tutaj kluczowym elementem. Polega ona na nieprzerwanym dostępie zespołu do tej osoby i ma na celu zasypywanie bariery niewiedzy i nieporozumienia między biznesem i wykonawcą oprogramowania. Natomiast brak potrzeby szczegółowego dokumentowania wymagań pozwala na omawianie szczegółów i natychmiastowe przystępowanie do ich implementacji. Takie działanie wiąże się jednak z koniecznością wzajemnego zaufania, by bez pisemnych dowodów rozwiązywać problemy.

Właściciel produktu jest odpowiedzialny za definiowanie kryteriów akceptacji dla elementów rejestru produktu[4]. Są to warunki konieczne do uznania przez niego, że wymagania funkcjonalne i niefunkcjonalne są zrealizowane. WP w oparciu o te kryteria może pisać testy akceptacyjne lub angażować do tego ekspertów lub członków zespołu dewelopera. Bez kryteriów akceptacji zespół dewelopera nie będzie w pełni rozumiał danego elementu rejestru produktu i nie będzie mógł go przyjąć do realizacji. Jednakże to WP odpowiada za sprawdzenie, czy zostały spełnione kryteria akceptacji. Zespół może pomóc w stworzeniu odpowiedniego środowiska testowego, ale ostateczny werdykt wydaje właściciel produktu.

2.1. Dwuetapowe podejście TDD

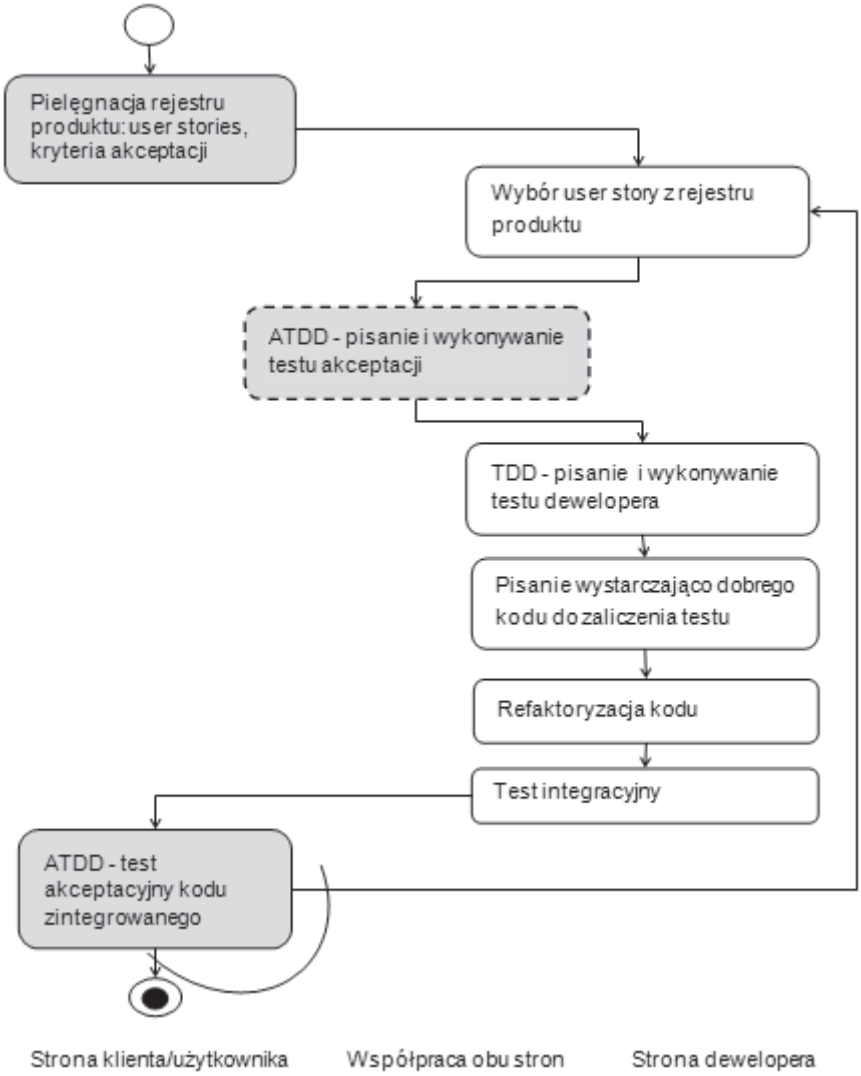
Proces pisania kodu i jego testowania w podejściu sterowanym testowaniem (*Test Driven Development*) jest istotnie różny od podejścia tradycyjnego – zmienia porządek w sekwencji działań „pisanie kodu" i „testowanie kodu" na odwrotną kolejność, najpierw testowanie, a po nim pisanie kodu. Programista pracuje w krótkich cyklach: identyfikacja i automatyzacja testu niepowodzenia, pisanie wystarczająco dobrego kodu do zaliczenia testu i czyszczenie kodu w sposób niezbędny przed rozpoczęciem powtórzenia cyklu dla nowego testu. Powtarzalność cyklu odbywa się bardziej co kilka minut niż co kilka godzin (Cohn, s. 156–158).

Testowanie powinno być zakończone decyzją o zaliczeniu testu, czyli stwierdzonej wystarczająco dobrej jakości kodu. Testowanie po stronie dewelopera polega na usuwaniu błędów technicznych. Natomiast jakość kodu to również jego odpowiedniość w odniesieniu do potrzeb klienta, badanych za pomocą kryteriów akceptacji. W podejściu TDD, kiedy testowanie wyprzedziło kodowanie, wystąpiła możliwość, ale i potrzeba testowania akceptacyjnego, również wyprzedzającego pisanie kodu produkcyjnego.

Pojawienie się testów akceptacyjnych na początku cyklu wytwarzania stało się potrzebne i możliwe w wyniku doświadczeń z podejściem sterowanym testami. Skoro możliwe jest wyprzedzenie pisania kodu pisaniem testów, to również podjęto próby pisania testów akceptacyjnych. Tym bardziej niezbędnych, kiedy w nowoczesnym podejściu Agile nie występuje odrębna faza specyfikacji wymagań ani odrębna faza testowania. Jednakże fazy te muszą być i są realizowane, ale w zakresie i kolejności wynikającej z zasad podejścia Agile: specyfikacja wymagań na poziomie wymaganym dla *user story*, czyli w wystarczającym zakresie dla podjęcia decyzji o rozpoczęciu ich realizacji w cyklu wytwarzania nazywanym sprintem

[4] Rejestr produktu jest odpowiednikiem specyfikacji wymagań w podejściu tradycyjnym, czyli wymaganych funkcjonalności, ale dokumentowanych w postaci znacznie uproszczonej. Składa się z pozycji nazywanych esejami bądź historyjkami użytkownika (ang. *user stories*), w zależności od poziomu ich złożoności. Stanowi zbiór narracji klienta/użytkownika uporządkowany według priorytetów (Cohn, s. 235).

(iteracja), a testowanie odbywa się w odpowiednim miejscu tego sprintu, najczęściej na początku zgodnie z podejściem TDD.



Rys. 2. Model przepływu sterowania testami przy dwuetapowym podejściu TDD

Źródło: opracowanie własne.

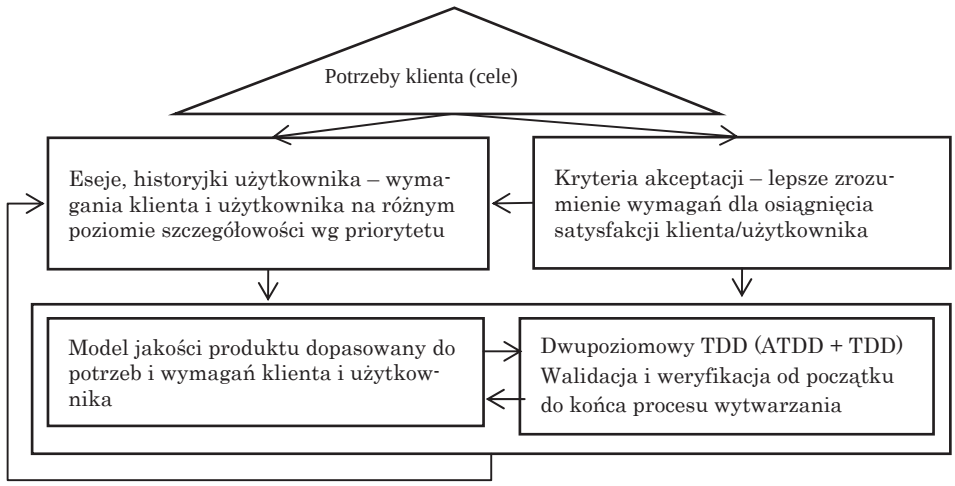
Podsumowując, kryteria akceptacji powinny być dostarczone wraz z treścią *user story* przed rozpoczęciem sprintu, a przynajmniej nie później niż spotkanie przeglądu sprintu, na którym podejmuje się decyzję o akceptacji tego wydania. Najlepszym rozwiązaniem wydaje się być koncepcja, by rejestr produktu zawierał historyjki użytkownika wraz z kryteriami ich akceptacji. Model cyklu wytwarzania

z podejściem TDD staje się dwufazowy, najpierw faza ATDD, a później faza TDD z powrotem do fazy ATDD w celu ostatecznej walidacji programu (rys. 2).

Należy podkreślić, że odpowiedzialność za wyprzedzające testy deweloperskie jest całkowicie po stronie programisty, a za testy akceptacyjne po stronie użytkownika, w praktyce jednak, ze względu na nowość podejścia, wykonywane przez programistę/testera, a najczęściej we współpracy obu stron (Beck, s. 211). Jednakże oznacza to włączanie klienta/użytkownika w proces walidacji znacznie wcześniej, niż jest to tradycyjnie przyjęte, i oznacza dodatkowe obowiązki. Dzieje się to jednak w zamian za brak konieczności tworzenia szybko tracącej na aktualności dokumentacji i z korzyścią dla wcześniejszego sprawdzenia jakości specyfikacji wymagań.

3. Rozwijanie modelu jakości w iteracyjnym, sterowanym testowaniem podejściu Agile

Testy akceptacyjne wykonywane na początku tego procesu, traktowane jako wczesna walidacja produktu, wykonywana na kodzie testowym, pozwalają na wczesne przedstawienie relacji między abstrakcją a jego rzeczywistą implementacją.



Rys. 3. Sterowanie jakością za pomocą iteracyjnego modelu jakości, dopasowanego do potrzeb klienta i użytkownika za pomocą testowania akceptacyjnego i deweloperskiego, wyprzedzającego pisanie kodu produkcyjnego

Źródło: opracowanie własne.

Takie postępowanie uznano za podstawę koncepcji budowy modelu jakości w sposób iteracyjny, czyli taki, jakim charakteryzuje się podejście Agile (rys. 3). Ta

koncepcja wiąże się z włączeniem myślenia o jakości od początku i rozwijaniem modelu na kolejnych poziomach uszczegóławiania wiedzy o projekcie i produkcie. Zaczynając od definiowania celów dla projektu, opracowanie jego wizji, a następnie definiowanie wymagań za pomocą nowych reguł i tworzenie listy pozycji rejestru produktowego, uporządkowanej według priorytetów.

Model ten, dla zapewnienia wymaganej jakości produktu końcowego, będzie stanowił istotne przedstawienie relacji między potrzebami klienta/użytkownika i ich rozwiązaniem. Podstawą tej koncepcji jest powiązanie opowiadań i ich kryteriów z charakterystykami modelu jakości produktu (tabela 1). Uzyskane za pomocą testów wykonywalnych wyniki pozwolą nie tylko przewidywać, ale realnie mierzyć atrybuty jakości wewnętrznej i zewnętrznej oraz efekty użycia produktu, co jednak wymaga przygotowania odpowiedniego środowiska. Model ten będzie rozwijany, zgodnie z projektem rozwoju produktu, wydawanego użytkownikowi do eksploatacji w częściach jako zaakceptowanego i działającego oprogramowania.

Tabela 1

Struktura formularza definiowania modelu jakości z przykładami
(charakterystyki jakości zostały uszczegółowione do poziomu podcharakterystyk)

Ważność	**User story**	**Kryterium akceptacji**	**Jakość W i Z**	**Jakość w użyciu**
Musi być	Jako Administrator, chcę tworzyć konta użytkowników, żeby nadawać użytkownikom uprawnienia dostępu do systemu (seguetech 2013).	Nie chcę przydzielać nowego Użytkownika do Użytkownika „nieaktywnego”.	Funkcjonalność – odpowiedniość	Skuteczność – osiągalność celu zadania – poprawność zakończonego zadania
Musi być	Jako Użytkownik katalogu biblioteki chcę na stronie tytułowej możliwości zaawansowanego wyszukiwania, żeby szybko i łatwo zmieniać moje poszukiwanie (boostagile 2012).	Chcę ograniczenia wyszukiwania wg formatów/typów. Chcę ustalać wyszukiwanie według zakresu dat. Chcę ograniczyć wyszukiwanie wydawcy do takich informacji, jak: tytuł, autor, dziedzina, miejsce, wydawca, telefon.	Funkcjonalność – odpowiedniość	Skuteczność – osiągalność celu zadania – poprawność zakończonego zadania
Musi być	Jako Użytkownik chcę strony dostępnej, żeby nie przeżywać frustracji i nie szukać innej (mountaingoatsoftware 2015).	Chcę pracować na stronie przez 99,999 % czasu, w którym próbuję dostępu do niej.	Niezawodność – odporność na błędy	Satysfakcja – przydatność – zaufanie – komfort

Źródło: opracowanie własne; wykorzystano przykłady z podanych w tabeli stron internetowych oraz charakterystyki modelu jakości ISO/IEC TR 9126-2:2003, ISO/IEC CD 25022.3(E) 2014.

Współpraca z klientem, jako jeden z filarów podejścia Agile, daje naturalną możliwość wczesnej walidacji tworzonych części produktu (nazywanych wydaniami), wspartej bardziej przez rozmowy z właścicielem produktu niż spisaną dokumentację. Właściciel produktu, mając przez dłuższy czas styczność z działającym oprogramowaniem, najpierw w postaci kodu testowego, a następnie produkcyjnego, potrafi odpowiedzieć na pytania dotyczące jakości i wskazywać błędy, których podstawą jest niezrozumienie rzeczywistych potrzeb. Zaangażowanie użytkownika w proces tworzenia produktu i traktowanie takiego procesu nie jako proces ciągłego wyszukiwania błędów, ale jako efektywnej pracy nad produktem, z pewnością korzystnie wpłynie na końcowy efekt, czyli żądaną jakość produktu.

Podsumowanie

Zapewnianie jakości to złożone zadanie, wymagające odpowiedniego planu działań. Model jakości tworzonego produktu może stanowić podstawowe narzędzie zarządzania oczekiwaną jakością. Zbiór żądanych charakterystyk jakości zawarty w modelu będzie wspomagał sterowanie tym procesem.

Dynamicznie rozwijany zbiór wymagań i kryteriów ich akceptacji, jakim jest rejestr produktowy podejścia Agile, pozwala na rozwijanie opisu wymagań jakościowych, prezentowanych za pomocą modelu jakości, który podobnie jak cały proces wytwarzania staje się modelem iteracyjnym.

Podejście sterowane testowaniem, a szczególnie dwuetapowe TDD, pozwala na zapewnienie działań projakościowych od początku procesu wytwarzania oprogramowania. To podejście nie zmienia innych poziomów testowania niż jednostkowe, a testy integracyjne i akceptacyjne na końcu zintegrowanego produktu nie powinny być pomijane, ponieważ każda ingerencja w istniejący kod może spowodować zmianę jego działania. Jednakże czas poświęcony na końcowe testy walidacji produktu będzie znacznie skrócony przez wykonane wcześniejsze testy, a także poprzez możliwość ich automatyzacji, ponieważ będą powtórzeniem testów już wykonanych.

Podstawowe zasady Agile oraz wytwarzanie, sterowane dwuetapowym testowaniem, dostarczają realnych podstaw do zapewniania jakości produktu poprzez aktywny udział przedstawiciela biznesu w przygotowaniu wymagań, definiowaniu kryteriów akceptacji oraz uczestniczeniu w testowaniu akceptacyjnym, wyprzedzającym pisanie kodu produkcyjnego.

Literatura

1. Beck K. (2014), *TDD. Sztuka tworzenia dobrego kodu*, Helion, Gliwice.
2. Boehm B. (1981), *Software Engineering Economics*, Prentice-Hall.
3. Bentley C. (2003), *PRINCE2. Practical Handbook*, Butterworth-Heinemann, Oxford.
4. Cohn M. (2010), *Succeeding with Agile. Software Development Using Scrum*, Addison-Wesley, Michigan.
5. ISO/IEC TR 9126-2:2003.
6. ISO/IEC CD 25022.3(E) 2014.
7. Poppendieck M., Poppendieck T. (2007), *Implementing Lean Software Development.* Addison-Wesley.
8. Sommerville I. (1995), *Software Engineering*, Addison-Wesley, USA.
9. http://agilemanifesto.org (2014).
10. http://www.seguetech.com/blog/2013/03/25/characteristics-good-agile-acceptance-criteria/ (2013).
11. http://boostagile.com/user-stories-part-2-acceptance-criteria/ (2012).
12. http://www.mountaingoatsoftware.com (2015).

ITERATIVE PRODUCT QUALITY MODEL IN AGILE

Summary

As a result of changes in the IT project management, focused in Agile methods, is a concept of iterative approach to define the product quality model. The elaborated model of the two-phase test driven development, by analyzing the process stages of testing is presented. There was noted, the ability to combine the voice of the client/user to the developer, using proposed iterative quality model. Also is described the way of definition elaborated model for selected user stories.
Keywords: quality characteristic, acceptance criteria, testing.

Translated by Grażyna Hołodnik-Janczura