Ekonomiczne Problemy Usług nr 1/2017 (126), t. 2

ISSN: 1896-382X | www.wnus.edu.pl/epu
DOI: 10.18276/epu.2017.126/2-08 | strony: 73–82

Grażyna Hołodnik-Janczura
Politechnika Wrocławska
Wydział Informatyki i Zarządzania
Katedra Badań Operacyjnych, Finansów i Zastosowań Informatyki
e-mail: ghj@pwr.edu.pl

Poprawa jakości specyfikacji wymagań za pomocą przykładów

Kod JEL: M15
Słowa kluczowe: cel biznesowy, kryteria jakości, specyfikacja wykonywalna, wspólny język
Streszczenie. Celem artykułu jest porównanie dwóch, różnie nazywanych metodyk, mających na celu poprawę jakości specyfikacji wymagań w zwinnych projektach IT. Omówiono podstawowe zasady podejścia Adzica i BDD Northa. Przedstawiono listę powszechnie stosowanych kryteriów jakości, którą uzupełniono o dwa kryteria, istotne dla rozważanego problemu. Wyniki analizy porównawczej, przeprowadzonej ze względu na opisane kryteria, wskazują na istniejące w zaprezentowanej koncepcji, poszukiwane możliwości poprawy jakości specyfikacji wymagań.

Wprowadzenie

Nowoczesne zarządzanie projektami informatycznymi charakteryzuje się potrzebą włączania przedstawicieli biznesu do prac zespołu deweloperskiego w odpowiedzi na presję jak najszybszego dostarczenia klientowi działającego programu komputerowego, wspomagającego realizację jego celów. Pozornie nie jest to nic nowego, tak było od początku zastosowań informatyki, jednak żądania klientów są coraz trudniejsze do spełnienia. Mimo że dziesiątki lat rozwoju informatyki doprowadziły do powstania wielu metodyk i różnych podejść, usprawniających procesy wytwarzania i wdrażania oprogramowania, to wysoka specjalizacja i oddalenie twórców końcowego produktu od zamawiającego nie poprawiały sytuacji. W efekcie, w połowie lat 90. XX wieku zaczęły

powstawać metody silnie ukierunkowane na klienta, tzw. adaptacyjne lub zwinne, podobne, ale różnie nazywane, z niewiele różniącymi się koncepcjami[1].

Dla jakości oprogramowania szczególnie istotne są takie części procesu jego wytwarzania jak specyfikacja wymagań i testowanie, które charakteryzują się nagromadzeniem terminologii technicznej, a niekiedy nawet chaosem terminologicznym, utrudniając porozumienie interesariuszy obu stron projektów IT. W efekcie wystąpił wzrost trudności komunikacyjnych, a między klientem i deweloperem nadal istnieje trudna do pokonania bariera językowa.

Koncepcja „wspólnej specyfikacji" Gojko Adzica (2014) oraz podejście BDD (*Behavior-Driven Development*), zainicjowane przez Dana Northa (2006), są szansą na uporządkowanie definicji stosowanych terminów, uproszczenie języka technicznego, trudnego dla przedstawicieli biznesu, a także na oczekiwane powiązanie właściwości dostarczanego oprogramowania z celami biznesu. Pojawienie się takiej koncepcji, stało się dlatego przesłanką do analizy ich podobieństwa, a także ich wpływu na poprawę jakości specyfikacji wymagań.

1. Rola jakości specyfikacji wymagań

Jakość oprogramowania według wielu definicji to zgodność z wymaganiami (Crosby, 1979), to przydatność do użytkowania (Juran, Gryna, 1970), a według innych to jedno i drugie (ISO 9126-1: 2000). Okazuje się, że szczegółowe definicje tych pojęć są ze sobą zbieżne i prowadzą do wniosku, że słaba jakość specyfikacji wymagań to również słaba jakość produktu końcowego. Na istniejące zależności między jakością potrzeb i jakością końcowego produktu wskazuje precyzyjnie kompleksowo zorganizowana grupa standardów jakości produktu informatycznego pod nazwą SQuaRE (UIO, 2016).

Jakość opisu wymagań w rankingu przyczyn niepowodzenia projektów IT

Systematyczne badania organizacji Standish Group (Projectsmart 2014) wyraźnie wskazują, że do pierwszych dwóch czynników, przeszkadzających w zakończeniu sukcesem projektów informatycznych, należą: niekompletne wymagania (13,1%) oraz brak zaangażowania użytkownika (12,4%), natomiast, według badań Capersa Jonesa (2009–2012), około 20% usterek oprogramowania wynika z błędów specyfikacji wymagań, które są zaliczane do jednej z najważniejszych kategorii problemów pojawiających się w procesach wytwarzania oprogramowania.

[1] Niejasności dotyczą zarówno zasad i praktyk podejścia Agile, jak i braku jednoznacznych definicji używanych pojęć, jak np. z używanie terminu „sterowanie": podejście sterowane testowaniem TDD, podejście sterowanie testowaniem akceptacyjnym ATDD, podejście sterowanie zachowaniem BDD (Adzic, 2014, s. 18).

Kryteria jakości specyfikacji wymagań

Capers Jones (2009–2012) wymienia aż 36 kategorii problemów dotyczących specyfikacji wymagań. Do najczęściej stosowanej listy kryteriów jakości specyfikacji wymagań należy zaś następująca klasyfikacja standardowa (IEEE 830-1998):

1. Poprawność – nie ma ustalonych reguł poprawności, zależą one od konkretnej organizacji, może to być zgodność z obowiązującymi standardami albo innymi ważnymi dokumentami dla działalności danej organizacji.
2. Jednoznaczność – ze względu na pisanie specyfikacji wymagań w języku naturalnym możliwe jest różne rozumienie podanych terminów, stąd też w celu unikania wieloznaczności zachodzi potrzeba doprecyzowania stosowanej terminologii za pomocą słownika terminów i opisu reguł formalnych, określającej jednoznaczne rozumienie zapisanego wymagania.
3. Kompletność – ta cecha wymaga sprawdzenia trzech warunków: obecność wymagań funkcjonalnych i niefunkcjonalnych, konieczność odpowiedzi programu na wejście poprawne i niepoprawne, a także konieczność powiązania z artefaktami programu.
4. Spójność – dotyczy spójności wewnętrznej. Możliwe niespójności to: niezgodne operacje na danych, np. mnożenie zamiast dodawania, żądany format wyników raz w postaci tabelarycznej, a w drugim miejscu w postaci graficznej.
5. Uporządkowanie według ważności/stabilności – ważność to najczęściej wartość dla klienta, a stabilność to możliwa zmienność danego wymagania.
6. Weryfikowalność – istnienie sposobu jednoznacznego stwierdzenia implementacji, a także poprawności wykonania tego wymagania.
7. Modyfikowalność – struktura i styl dokumentu powinny pozwalać na łatwą i szybką modyfikację danego wymagania.
8. Możliwość śledzenia powiązań – znane pochodzenie wymagania i znane referencje między innymi wymaganiami oraz innymi artefaktami.

W projektach zwinnych, ze względu na pierwszorzędny warunek satysfakcji klienta, uznaje się za niezbędne kryterium powiązania wymagań z celami biznesowymi[2]. Nie występuje ono jednak w jawnej postaci w powyższym wykazie, choć można sugerować jego znaczenia dla spełnienia kryteriów numer 1, 5 i 8. Podobnie wygląda kwestia poprawnego rozumienia każdego wymagania, zarówno przez klienta, jak i dewelopera, jednak nie dotyczy to tylko jednoznacznego rozumienia używanych pojęć (kryterium nr 2), ale przede wszystkim, możliwych konsekwencji jego braku dla obu stron (np. wpływ na straty czy też przychody). Proponuje się zatem uzupełnienie tego wykazu o dwa, klarownie sformułowane kryteria:

- spójność z celami biznesowymi,
- obustronna zrozumiałość.

[2] Odpowiednią techniką do badania i prezentacji powiązań między wymaganiami i celami biznesowymi może być dwuwymiarowa macierz krzyżowa, w której, na przecięciu wiersza i kolumny, powinno być wskazane istnienie związku lub też jego brak, a także charakterystyka jego znaczenia, np. wg odpowiednio przyjętej skali ocen.

2. Procesy wspólnej specyfikacji wymagań z przykładami Adzica

Pojęcie „wspólnej specyfikacji" wymagań, wprowadzone przez Gojko Adzica (2014), oznacza zaangażowanie przedstawiciela biznesu w procesach dewelopera, a także odwrotnie, włączenie dewelopera w problematykę realizacji celów biznesu, wpisującej się w rozwiązywanie problemów identyfikacji wymagań, połączonej z zainteresowaniem obu stron w używaniu również „wspólnego języka"[3].

Wzorce kluczowych procesów specyfikacji wymagań przez przykłady

Koncepcja nowej organizacji i struktury procesów została opracowana na podstawie badań prowadzonych przez Gojko Adzica (2014) na rzeczywistych projektach. Wyniki badań zaprezentowano w postaci wzorców, możliwych do zastosowania przez inne zespoły, bez konieczności powtarzania wielu prób i błędów. Nie stanowią one pełnego kompendium wiedzy, ale mogą być podstawą do opracowania własnej koncepcji wdrożenia tych praktycznych zasad.

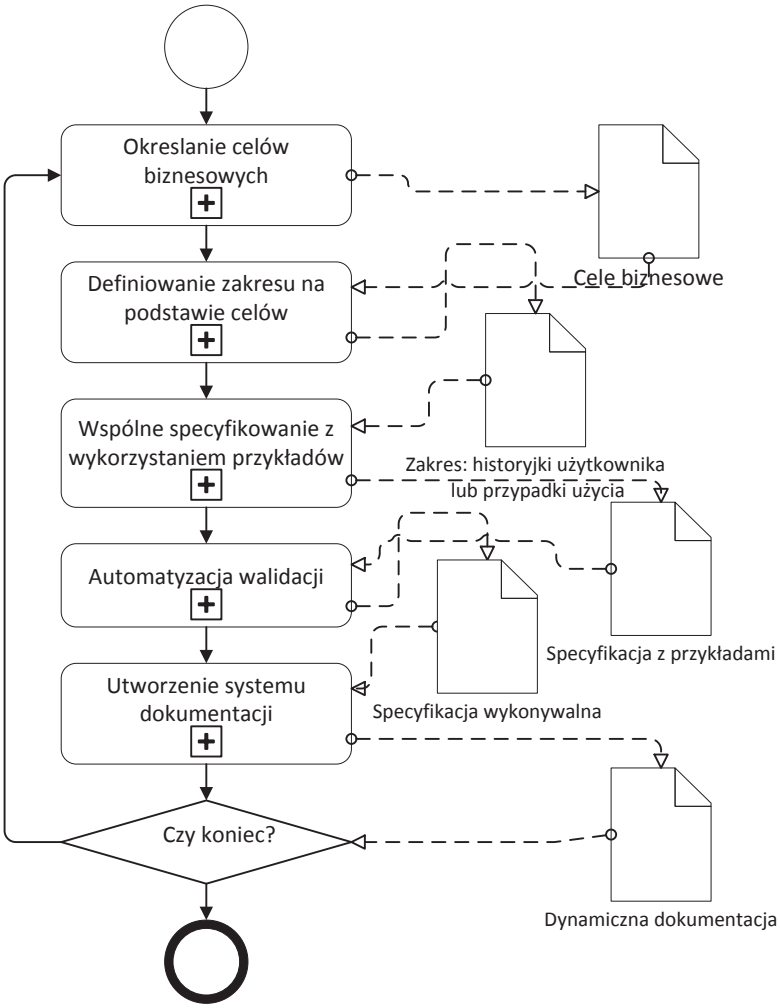
Definiowanie zakresu implementacji na podstawie celów

Właściwości dostarczonego klientowi produktu informatycznego powinny odpowiadać jego potrzebom, przede wszystkim w zakresie realizacji celów biznesowych. Koncepcja identyfikacji wymagań przez zrozumienie celów biznesowych jest podstawą definiowania zakresu historyjek użytkownika lub przypadków użycia. Klient powinien definiować jakie cele chce osiągnąć, a deweloper – jakie są możliwe informatyczne rozwiązania jego problemów.

Wspólne specyfikowanie wymagań

Ludzie biznesu posługują się językiem naturalnym i terminami charakterystycznymi dla własnej dziedziny. Zespół dewelopera to specjaliści korzystający z silnie powiązanym z ich charakterem pracy językiem technicznym. Istniejąca bariera językowa jest trudna do pokonania bez tej świadomości po obu stronach oraz chęci znalezienia wspólnego języka. Wspólny język okazuje się być podstawowym narzędziem niezbędnego, obustronnego zrozumienia. Analiza wielu projektów doprowadziła Gojko Adzica (2014) do przekonania, że specyfikacja wymagań za pomocą rzeczywistych przykładów oczekiwanego zachowania programu komputerowego jest tym elementem języka, który prowadzi do takiego efektu. Klient powinien określać, co powinno robić oprogramowanie, a deweloper, w jaki sposób program komputerowy powinien to wykonywać.

[3] Używanie języka zrozumiałego dla wszystkich uczestników projektu IT, takiego by wszyscy myśleli o danej aplikacji w ten sam sposób jest nie tylko elementem metodyki BDD, ale i Domain-Driven Design, w której ten język jest nazywany „językiem wszechobecnym" (*ubiquitous language*). W niniejszej pracy używa się termin „wspólny język".



Rysunek 2. Wzorzec kluczowych procesów i artefaktów specyfikacji wymagań z przykładami

Źródło: opracowanie własne na podst. Adzic (2014), s. 44.

Przykładowo, klient potrzebuje „sprawdzać status zleconej przesyłki". W odpowiedzi na pytanie dewelopera o szczegóły statusu tego zlecenia, klient może je określić za pomocą następującego przykładu: zlecenie to przesyłka o numerze 100, przyjęta 12.01.2016 roku od firmy o nazwie „ODRA_T", która aktualnie znajduje się w miejscu przeładunkowym „Sztokholm", do którego dotarła 13.01.2016 roku o godzinie 12:15, a czas dotarcia do miejsca docelowego to 13 godzin. Taka narracja opisowa tego przykładu, a także kolejnych, może zostać łatwo uporządkowana za pomocą układu tabelarycznego (tab.1).

Tabela 1. Przykłady zapisu specyfikacji wymagania „sprawdzenie statusu zleconej przesyłki”

Nr zlecenia	Data przyjęcia	Nazwa firmy	Miejscowość przeładunkowa	Data przeładunku	Godzina przeładunku	Czas dotarcia
100	12.01.16	ODRA_T	Sztokholm	13.01.16	12:15	13
001	31.12.16	ART._no5	Gdańsk	01.01.17	10:00	5

Źródło: opracowanie własne.

Powyższy układ zapisu rzeczywistych faktów pozwala na ograniczenie zbędnych szczegółów, eliminację nadprogramowych informacji oraz wybór kluczowych przykładów do wykorzystania nie tylko podczas testowania jednostkowego, ale i w testowaniu akceptacyjnym. Następnie, napisane testy przekształcą statyczny zapis specyfikacji wymagań do postaci specyfikacji wykonywalnej, a powstająca w tym czasie dokumentacja stanie się dokumentacją dynamiczną, nazywaną też żyjącą.

3. Podejście BDD

Podejście sterowane zachowaniem[4] BDD zostało wypracowane i jest dalej rozwijane w badaniach wielu firm i autorów nad tworzeniem oprogramowania w projektach zwinnych w dążeniu do budowy takiego oprogramowania, którego użytkownik rzeczywiście potrzebuje (Ambler, 2016; Evans, 2003; Highsmith, 2009; North, 2006; Smart, 2016, s. 26; Hołodnik-Janczura, 2016). Podstawą tych badań jest przekonanie o potrzebie zrozumienia celów biznesowych organizacji i dostarczenie klientowi takich właściwości programu, które przyczynią się do realizacji jego celów. Metody i narzędzia BDD, niezależnie od rodzaju stosowanych metod, pozwalają na usprawnianie procesów wytwarzania, a przede wszystkim na zmniejszenie bariery komunikacyjnej między ludźmi biznesu i zespołem dewelopera.

Szablon scenariusza jako element wspólnego języka

Tradycyjny proces tworzenia oprogramowania składa się z następujących kroków – klient podczas wywiadu z analitykiem opowiada o swoich działaniach w jego obszarze działalności, następnie analityk biznesowy przekształca je w zrozumiały dla siebie sposób w model wymaganej funkcjonalności za pomocą diagramów i szczegółowych specyfikacji opisowych, programista, już w odosobnieniu, tłumaczy otrzymaną specyfikację wymagań na postać kodu oraz przeprowadza testy jednostkowe. Tester, również osobno, na podstawie notatek analityka, tworzy przypadki testowe i przeprowadza weryfikację wraz z walidacją otrzymanego kodu. W tym czasie dokumentalista opracowuje dokumentację techniczną i użytkową. Podczas każdego z tych przekształceń mogą

[4] Termin „programowanie sterowane zachowaniem” po raz pierwszy zostało użyte przez Erica Evansa (2003). Techniki takiego programowania czerpią z doświadczeń praktyk programowania sterowanego testowaniem TDD oraz DDD, stosowanych w projektach Agile.

wystąpić zniekształcenia lub utrata części informacji, a podczas testowania akceptacyjnego lub dopiero w trakcie jego użytkowania, może się okazać, że właściwości otrzymanego oprogramowania nie spełniają oczekiwań użytkownika.

Techniki BDD polegają natomiast na współpracy analityka biznesowego, programisty i testera, którzy starają się używać tzw. wspólnego języka, zaczynając od potrzeb klienta, poprzez specyfikację wymagań i wczesne testy, przechodzą do pisania działającego kodu, skupiając się na ważnych dla klienta właściwościach. Procesy te przebiegają iteracyjnie i przyrostowo, dostarczając najszybciej, jak to tylko możliwe, najpotrzebniejszych przyrostów oczekiwanego produktu. Kluczowym elementem porozumienia wszystkich uczestniczących stron jest podejście skoncentrowane na wynikach działania kodu programu w kategoriach biznesowych.

W celu prawidłowego zrozumienia rzeczywistych oczekiwań klienta, wymaga się prowadzenia konwersacji z klientem, bazujących na przykładach zachowania programu, wyrażonych w prostym języku naturalnym, ale o specjalnej pseudo strukturze[5]. Zgodnie z szablonem BDD, zapis scenariusza dla przykładu z pkt. 2 „sprawdzanie statusu zlecenia przesyłki" przyjmie postać:

Zakładając, *że przyjęto zlecenie o numerze <nr> w dniu <data przyjęcia> od klienta <nazwa firmy>*

Gdy *sprawdzam status zlecenia,*

Wtedy *powinienem zobaczyć, zgodnie z wyznaczoną trasą, nazwę ostatniego miejsca przeładunku <nazwa miejscowości przeładunkowej> z datą <data przeładunku> i godziną <godzina przeładunku>*

I *obliczony według zadanej formuły, przewidywany czas to <liczba godzin> potrzebna na dotarcie przesyłki do miejsca docelowego.*

Autorzy podejścia BDD proponują uzupełnianie takiego scenariusza przykładami zachowania programu, w takim samym układzie tabelarycznym, jak w podejściu specyfikacji wymagań z przykładami Adzica (tab. 1).

4. Pokrycie kryteriów jakości przez wspólne zasady BDD i Adzica

Metodyka BDD i zasady specyfikacji wymagań przez przykłady charakteryzują podobne założenia, choć różnią się stosowanymi pojęciami. W podejściu BDD pisze się specyfikację wymagań za pomocą scenariuszy uzupełnianych przykładami, natomiast w koncepcji Adzica (2014, s. 16–18), zamiast terminu „scenariusz" wprowadza się termin „przykład", uznany za lepiej przyswajalny. W obu podejściach zauważa się dą-

[5] Narracja takiego szablonu scenariusza stanowi często stosowaną postać pod nazwą Gherkin i przypomina strukturę szablonu historyjki użytkownika Cohna (2010, s. 239). Pisanie takich scenariuszy jest wspomagane za pomocą narzędzi, np. JBehave, Cucumber, SpecFlow, które znacznie ułatwiają automatyzację testowania i pozwalają na prowadzenie wczesnych testów akceptacyjnych (Smart, 2016, s. 37).

żenie do wspólnego celu, czyli poprawy jakości za pomocą podobnej koncepcji. W analizie porównawczej podstawowych elementów obu podejść, zauważono występowanie istotnych możliwości poprawy jakości specyfikacji wymagań w zakresie standardowych kryteriów IEEE 830 (tab. 2, znak „+"), natomiast na pytanie o skalę tej poprawy, autor zamierza odpowiedzieć, prowadząc dalsze badania.

Tabela 2. Wspomaganie jakości specyfikacji wymagań przez BDD i zasady Adzica

Kryterium IEEE 830	Związek z celami	Wspólny język	Specyfikacja z przykładami	Obowiązująca dokumentacja
Poprawność	+			
Jednoznaczność		+	+	
Kompletność		+	+	+
Spójność		+	+	
Uporządkowanie	+	+		+
Weryfikowalność		+	+	
Modyfikowalność			+	+
Śledzenie powiązań	+			+

Źródło: opracowanie własne.

Podsumowanie

Często spotykany rozdźwięk między oczekiwaniami klienta i dostarczonymi możliwościami produktu informatycznego jest najczęściej wynikiem niespełniania kryteriów jakości specyfikacji wymagań. Nie jest to nowy problem, ale związany z brakiem zainteresowania obustronnym zrozumieniem. Początkowy okres tworzenia małych programów dziedzinowych umożliwiał bliski kontakt klienta/użytkownika z informatykiem, co pozwalało na wspólne dochodzenie do oczekiwanych właściwości oprogramowania. Postępująca specjalizacja, rozwój technologii i terminologii doprowadziły natomiast do istniejącej bariery komunikacyjnej między klientem i informatykiem.

Nowe zasady podejścia do procesów wytwarzania oprogramowania, rozwijane od końca lat 90. ubiegłego wieku, ponownie wprowadzają potrzebę bliskiego kontaktu zespołu dewelopera z przedstawicielem biznesu. W skład zespołu dewelopera powinien wchodzić przedstawiciel biznesu o istotnych kompetencjach w zakresie decydowania zarówno o zawartości zbioru wymagań, jak i ich znaczenia dla realizacji zamierzonych celów biznesowych. Zespół dewelopera powinien zaś starać się nie tylko zrozumieć wymagania klienta, ale również jego problemy, które powinny być rozwiązywane za pomocą programu komputerowego.

Taka współpraca powinna zmniejszyć dystans między stronami, a przede wszystkim poprawić komunikację między nimi. Efekt synergii obu stron tego samego procesu, nazywanego podejściem BDD czy też specyfikacją z przykładami, powinien prowadzić

do wielopoziomowej, stopniowo doprecyzowywanej specyfikacji wymagań, a następnie dostarczenia produktu informatycznego, będącego poprawną implementacją tak definiowanego zbioru wymagań.

Bibliografia

Adzic, G. (2014). *Specyfikacja na przykładach*. Gliwice: HELION.

Ambler, S. (2000). *Surveys Exploring the Current State of Information Technology Practices*. Pobrano z: http://www.ambysoft.com/surveys/ (30.12.2016).

Cohn, M. (2010). *Succeeding with Agile*. Addison-Wesley.

Crosby, P.B. (1979). *Quality is Free: The Art of Making Quality Certain*. Nowy Jork: McGraw-Hill.

Evans, D. (2003). *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley Professional.

Highsmith, J. (2009). *Agile project Management: Creating Innovative Products*. Addison-Wesley Professional.

Hołodnik-Janczura, G. (2017). The Extension of User Story Template Structure with an Assessment Question Based on the Kano Model. W: J. Świątek, Z. Wilimowska, L. Borzemski, A. Grzech (red.), *Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology – ISAT 2016 – Part III. Series: Advances in Intelligent Systems and Computing*, Vol. 523, (s. 137–151), Springer.

IEEE Standard 830-1998 (2009). *Institute of Electrical and Electronics Engineers*.

ISO/IEC 9126-1:2000.

Jones C. (2009–2012). *Software Requirements and the Ethics of Software Engineering*. Pobrano z: www.concpts.gilb.com (2.01.2017).

Juran, J.M., Gryna, M. (1970). *Quality Planning and Analysis: From Product Development through Use*. Nowy Jork: McGraw-Hill.

North, D. (2006). *Introducing-BDD.* Pobrano z: https://dannorth.net/introducing-bdd/(30.12.2016).

North, D. (2012). *BDD-is-like-TDD-if.* Pobrano z: https://dannorth.net/2012/05/31/ bdd-is-like-tdd-if/ (30.12.2016).

Smart, J.F. (2016). *BDD w działaniu*. Gliwice: HELION.

www.uio.no/studier/emner/matnat/ifi/INF5181/h11/undervisningsmateriale/reading-materials/Lecture-06/04ZubrowISO25000SWQualityMeasurement.pdf (20.07.2016).

www.projectsmart.co.uk/white-papers/chaos-report.pdf (2014).

IMPROVE THE QUALITY OF REQUIREMENTS SPECIFICATION THROUGH SPECIFICATION BY EXAMPLE

Keywords: business objective, quality criteria, executable specification, ubiquitous language, behavior

Summary. The aim of this study was to verify whether the two, variously called methods, can aid agile projects IT in an improving the quality of requirements specification. The basic principles of approaches of Adzic and BDD of North was discussed. The following is a list of the commonly used criteria of quality, which has been enriched with two criteria relevant to the problem under consideration. The results of the comparative analysis carried out due to the described criteria, indicate existing presented concepts, sought to improve the quality of requirements specification.

Translated by Grażyna Hołodnik-Janczura

Cytowanie

Hołodnik-Janczura, G. (2017). Poprawa jakości specyfikacji wymagań za pomocą przykładów. *Ekonomiczne Problemy Usług*, *1* (126/2), 73–82. DOI: 10.18276/epu.2017.126/2-08.