PYTANIA PRÓBNE DO EGZAMINU NA CERTYFIKAT ZAAWANSOWANY REQB

Część 1(3): rozdziały 1 – 4.3

Na podstawie:

Syllabus REQB® Certified Professional for Requirements Engineering, Advanced Level, Requirements Manager
Wersja 1.0. 2011

Bogdan Bereza, 2017-01-20

Rodzaje pytań

K1 - pamiętać, rozpoznać, powtórzyć

K2 - rozumieć, wyjaśnić, uzasadnić, porównać, zaklasyfikować, podsumować

K3 – zastosować w kontekście, w konkretnej sytuacji

K4 – zanalizować

1. Podstawy

1.1 Powtórzenie podstaw (z REQB FL)

1.1.1 Proszę wybrać jedną, **w pełni** poprawną [*choć nie jedyną możliwą*] definicję wymagania:

A. Wymaganie, to warunek albo zdolność systemu, potrzebna, aby uzyskać zwrot z inwestycji
B. Wymaganie, to funkcja lub właściwość systemu informatycznego, potrzebna, aby zaspokoić potrzeby wybranych interesariuszy
C. Wymaganie, to udokumentowany opis warunku, właściwości lub zdolności systemu, potrzebnej, aby spełnić wymagania zawartego kontraktu lub zaspokoić potrzeby kluczowych interesariuszy
D. Wymaganie jest opisem właściwości systemu potrzebnej, aby zrealizować cel projektu

1.1.2 Proszę podać **niepoprawne** zdanie odnoszące się do wymagań poza-funkcjonalnych [niefunkcjonalnych]:

A. Wymagania poza-funkcjonalne dotyczą między innymi możliwości odtworzenia stanu systemu po awarii
B. Wymagania niefunkcjonalne z reguły określane są jakościowo i nie nadają się do określenia ilościowego
C. Wymagania niefunkcjonalne mogą spowodować powstanie, sprzyjających ich realizacji, dodatkowych wymagań funkcjonalnych
D. Nie da się określić jednej, takiej samej dla różnych rodzajów systemów, zasady, które wymagania są istotniejsze – funkcjonalne czy niefunkcjonalne

1.1.3 Proszę określić, które **dwa** z poniższych twierdzeń są prawdziwe:

A. Wymagania procesowe dotyczą procesu wytwórczego, a projektowe – modelu procesu, według którego realizuje się dany projekt
B. Wymaganie brzmiące „ze względu na to, że firma dysponuje już bazą danych Oracle, mowy system również musi korzystać z tej bazy danych”, to wymaganie procesowe
C. Wymagania produktowe określają cele biznesowe projektu, realizującego produkt
D. Wymagania produktowe to inaczej ograniczenia, na przykład wynikające ze względów technicznych
E. Wymagania procesowe mogą odnosić się zarówno do sposobu realizacji projektu, jak i do ograniczeń technicznych dla produktu

1.1.4 Jeśli znak ← oznacza relację zawierania („A ← B” oznacza „B jest częścią A”, czyli „B jest zawarte w A”), to które z poniższych jest prawdziwe?

A. Inżynieria systemów ← modelowanie procesów biznesowych
B. Pozyskiwanie wymagań ← zarządzanie wymaganiami ← śladowanie [śledzenie powiązań] wymagań
C. Inżynieria wymagań ← analiza wymagań ← pozyskiwanie wymagań
D. Inżynieria wymagań ← inżynieria oprogramowania
E. Negocjowanie wymagań ← walidacja wymagań

1.1.5 Jaka jest różnica w zadaniach między kierownikiem wymagań [*Requirements Manager*] a konstruktorem wymagań [*Requirements Developer*]?

A. Kierownik wymagań tworzy wymagania biznesowe, a konstruktor wymagań – systemowe [produktowe]
B. Kierownik wymagań odpowiada za negocjowanie i zarządzanie zmianami wymagań, a konstruktor – za ich modelowanie
C. Konstruktor wymagań przedstawia zaproponowane wymagania do akceptacji kierownikowi wymagań
D. Konstruktor wymagań specjalizuje się w pozyskiwani i analizie wymagań, a kierownik wymagań w zarządzaniu ich zmianami i kontaktach z interesariuszami

2. Modele procesu oraz zarządzanie

2.1 Modele procesu

2.1.1 Czym jest proces tworzenia oprogramowania?

A. To sekwencyjny, tradycyjny sposób realizacji projektów informatycznych
B. To standardowy, powtarzalny zestaw reguł i procedur, stosowanych do realizacji projektów informatycznych
C. To definicja dokumentacji projektowej i produktowej
D. To standardowy, powtarzalny zestaw reguł i procedur inżynierii wymagań

2.1.2 Kiedy warto stosować model kaskadowy?

A. Kiedy wymagania są możliwe do określenia z góry i należy się spodziewać ich częstych zmian
B. Kiedy wymagania produktowe są stabilne, a względy organizacyjne lub kontraktowe obligują do pełnego zakończenia każdej fazy projektu przed rozpoczęciem następnej
C. W projektach, obejmujących wytwarzanie zarówno platformy sprzętowej, jak i oprogramowania systemu IT
D. Tylko wówczas, kiedy zespół projektowy jest bardzo doświadczony i nie wymaga bezpośredniego nadzoru

2.1.3 Model V:

A. Jest iteracyjny i zakłada wczesne testowanie
B. Jest sekwencyjną odmianą modelu kaskadowego
C. Ważną rolę w modelu V pełni założenie, że częściowe testy akceptacyjne należy wykonywać już podczas projektowania architektury systemu
D. Jest sekwencyjny, podobnie jak model kaskadowy, ale kładzie dodatkowy nacisk zarówno na wczesne testowanie, jak i na wczesne planowanie i projektowanie testów dynamicznych

2.1.4 Czym jest RUP?

A. To model procesu wytwarzania oprogramowania, będący etapem pośrednim między modelami sekwencyjnymi a iteracyjnymi
B. To standardowy, komercyjny model procesu iteracyjnego, stworzony przez firmę Rational w latach 80-ych XX wieku
C. Słowo „unified” (ujednolicony) w nazwie RUP oznacza, że obowiązują ściśle określone zasady raportowania przestrzegania zasad tego procesu
D. To bezpośredni prekursor metodyk zwinnych, a jego autorzy stali się sygnatariuszami Manifestu Agile

2.1.5 Inżynieria wymagań w modelu RUP:

A. Zakłada stosowanie przypadków użycia do opisu wymagań
B. To dyscyplina występująca wg RUP w dwóch fazach cyklu życia: w fazie „rozpoczęcia" [*inception*] i w fazie „opracowania" [*elaboration*]
C. Dyscyplina inżynierii wymagań według RUP obejmuje między innymi takie czynności, jak identyfikacja potrzeb interesariuszy, przydzielanie wymaganiom priorytetów oraz projektowanie architektury biznesowej systemu
D. Testy akceptacyjne, będące częścią wymagań [zasada „specyfikacja na przykładach"], wykonuje się w fazie przekazania systemu [*transition*]

2.1.6 Czemu metody zwinne (agile) nazywane są „lekkimi", w przeciwieństwie do metod sekwencyjnych „ciężkich"?

A. Ponieważ metody zwinne przydatne są głównie do realizowania małych, „lekkich" projektów
B. Ponieważ metody zwinne zakładają minimalizację dokumentacji poprzez samodokumentujący się kod źródłowy
C. Ponieważ metody zwinne upraszczają i ograniczają praktyki zapewnienia i kontroli jakości
D. Ponieważ jest to modne określenia, kojarzące się z tańcem i zdrowiem
E. Ponieważ metody zwinne mają na celu elastyczne dostosowanie się do zmiennych wymagań i potrzeb interesariuszy zamiast sztywnego planowania

2.1.7 Manifest Agile:

A. Jest to model procesu wytwarzania oprogramowania, szczegółowo określający praktyki i wzorce iteracyjnego oraz przyrostowego wytwarzania oprogramowania
B. To zbiór zasad, mających na celu większą skuteczność i elastyczność projektów IT, pozostawiający swobodę doboru konkretnych metod i technik
C. To zbiór zasad, zakładających likwidację dokumentacji i planowania podczas realizowania projektów IT
D. Manifest Agile, to zwięzły opis metodyki zalecanej, w projektach wysokiego ryzyka, czyli takich, w których nie wiadomo do końca, co się tak naprawdę robi i jak to prawidłowo zrobić

2.1.8 Jedna z czterech głównych zasad Manifestu Agile [http://agilemanifesto.org/iso/pl/manifesto.html] określa, że „wyżej cenimy […] współpracę z klientem od negocjacji umów". Oznacza to w praktyce, że:

A. Preferowane jest wytwarzanie oprogramowania wolno-dostępnego, z otwartym kodem
B. Aby projekt realizowany zgodnie z zasadami agile zakończył się sukcesem, koniecznym warunkiem jest wcześniejsza znajomość klienta i dostawcy, oraz dobre, przyjazne relacje między uczestnikami projektu
C. Nie zakłada się terminu ukończenia projektu, tylko negocjuje w jego trakcie priorytety i kolejność przyrostowych dostaw funkcjonalności, założonych w planie wydań i sprintów („przebiegów")
D. Umowy cywilno-prawne „wdrożeniowe" powinny w mniejszym stopniu precyzować końcowy produkt, a w większym – zasady i formy częstej współpracy zleceniodawcy i dostawcy oprogramowania w trakcie projektu

2.1.9 Wskaż zasadę, która **NIE** jest jedną z 12 zasad zwinnego tworzenia oprogramowania [http://agilemanifesto.org/iso/pl/principles.html]:

A. Zalecane jest jak najczęstsze dostarczanie oprogramowania działającego, choć jeszcze niegotowego w całości
B. Bezpośrednią rozmowę uważa się za najlepszy sposób komunikacji
C. Zespoły wytwarzające oprogramowanie muszą określić własne, unikalne metody pracy
D. Zespoły tworzące oprogramowanie powinny od czasu do czasu przeanalizować swoje dotychczasowe sposoby pracy i zastanowić się, co i jak można by w nich ulepszyć

2.1.10 Historyjki użytkownika [*user stories*, opowieści użytkownika]:

A. Tworzy się w procesie współpracy między przedstawicielami zespołu deweloperskiego i przedstawicielami biznesu (użytkowników)
B. Muszą być bardzo krótkie i zwięzłe, zapisane w formie jednego zdania niezłożonego (bez spójników)
C. Tworzone są przez przedstawicieli zespołu deweloperskiego w taki sposób, aby maksymalnie uprościć proces ich późniejszej implementacji (realizacji)
D. Każda historyjka użytkownika powinna zaczynać się od słów „Jako użytkownik, chcę…", po których następuje opis wymaganej funkcji

2.1.11 Agile Scrum a Agile Crystal

A. Agile Scrum zakłada, że wymagania gromadzi się w rejestrze produktu [*product backlog*], natomiast Agile Crystal postuluje zastąpienie wymagań tak zwanymi celami produktowymi [*product goals*]
B. Agile Crystal wymaga częstej komunikacji między zespołem deweloperskim, a użytkownikami, podczas gdy Agile Scrum scedowuje tę rolę na Scrum Mastera
C. Agile Scrum i Agile Crystal to dwie metodyki wytwarzania oprogramowania, różniące się stopniem zgodności z zasadami Manifestu Agile
D. Agile Scrum i Agile Crystal to dwie metodyki wytwarzania oprogramowania, różniące się w detalach, ale zgodne z zasadami Manifestu Agile

2.1.12 Proszę wskazać **nieprawdzie** twierdzenie, odnoszące się do programowania ekstremalnego:

A. Programowanie ekstremalne stworzono w odpowiedzi na bardzo zmienne i dynamiczne potrzeby biznesu, w początkowym okresie intensywnej ekspansji nowatorskich rozwiązań internetowych
B. Programowanie ekstremalne zakłada częste kontakty oraz udzielanie informacji zwrotnej między deweloperami a użytkownikami
C. Szkoła programowania ekstremalnego rekomenduje tworzenie kodu przez dwuosobowe zespoły programistów
D. Programowanie ekstremalne realizuje zasadę, że oprogramowanie tworzone jest przez 5-9 osobowe, interdyscyplinarne zespoły deweloperskie

2.1.13 Dwie spośród zasad programowania ekstremalnego, to „odwaga" oraz „szacunek". Co one oznaczają w praktyce?

A. Wymaganie wysokiego poziomu etycznego od uczestników projektów realizowanych tą metodą
B. Bezwzględne dotrzymywanie wzajemnych zobowiązań, niezależnie od trudności technicznych i zmieniających się potrzeb biznesowych
C. Przejrzystość i otwartość udzielania informacji zwrotnej oraz unikanie prób wzajemnej manipulacji miedzy deweloperami a użytkownikami
D. Gotowość podejmowania wyzwań oraz nie lekceważenie przeciwnika

2.1.14 Przebieg (*sprint*) to:

A. Przebieg (inna nazwa: pojemność - *capacity*) określa szybkość pracy zespołu agile Scrum
B. To metoda programistyczna, zalecająca wcześniejsze tworzenie testów, a potem kodu
C. To codzienne, 15-minutowe spotkanie Zaspołu Scrum
D. To stosowane w Scrum określenie na 2-3 tygodniowy okres pracy Zespołu Scrum

2.1.15 Rejestr wymagań przebiegu (*sprint backlog*) to:

A. Wynik pracy (zrealizowane funkcje) zespołu Scrum po zakończeniu przebiegu
B. Lista wymagań, które zespół Scrum zobowiązał się zrealizować podczas przebiegu
C. Lista zadań do wykonania oraz plan pracy, znajdujące się na tablicy Scrum (*Scrum Board*)
D. Diagram, przedstawiający podczas przebiegu zadania wykonane i pozostające jeszcze do wykonania

2.1.16 Historyjka użytkownika | opowieść użytkownika (*user story*)

A. To wzorzec opisywania wymagań, zalecany i często stosowany w metodach agile
B. To metoda organizacji zadań do wykonania w formie tak zwanej „tablicy Kanban"
C. To scenariusz biznesowy, opowiedziany z punktu widzenia użytkownika końcowego
D. To wstępna wersja raportu z wywiadu (jednej z technik identyfikacji wymagań)

2.2 Zarządzanie i nadzorowanie procesu inżynierii wymagań

2.2.1 Proces inżynierii wymagań:

A. Jest sekwencyjny – raz pozyskane wymagania poddaje się analizie i zapewnieniu jakości, aby nie wprowadzać chaosu w projektach
B. Śladowanie [śledzenie powiązań] i nadzorowanie [statusu, zmian] już zebranych wymagań pozwala utrzymać spójność wymagań między sobą oraz wymagań z potrzebami biznesowymi
C. Identyfikacja wymagań jest warunkiem wyboru prawidłowej architektury oprogramowania, które je zrealizuje
D. Zapewnienie jakości wymagań gwarantuje jakość produktu, który te wymagania realizuje

2.2.2 Wymagania wewnętrzne i zewnętrzne… na tle innych podziałów wymagań

A. Jakość usług z perspektywy klienta i użytkowników, to funkcjonalne i niefunkcjonalne wymagania wewnętrzne
B. Przykładowe wymagania zewnętrzne z perspektywy zespołu deweloperskiego, to np. zastosowanie obiektowego języka programowania i dostosowanie formatu danych do formatu już wcześniej stosowanego w firmie klienta
C. Niefunkcjonalne wymaganie produktowe użytkownika, to na przykład żądanie, aby uczestnicy zespołu deweloperskiego znali język holenderski
D. Zewnętrzne wymaganie projektowe, to na przykład żądanie, aby uczestnicy zespołu deweloperskiego znali język portugalski

2.2.3 Jakie czynniki ***wewnętrzne*** **NIE** wpływają niekorzystnie na proces inżynierii wymagań?

A. Dobra znajomość dziedziny oprogramowania przez zespół deweloperski
B. Nieprecyzyjny i zmieniający się cel biznesowy
C. Zastosowanie odpowiedniej metody modelowania wymagań w procesie inżynierii wymagań
D. Zaangażowanie interesariuszy ze strony klienta

2.2.4 Możliwe rozwiązania problemów inżynierii wymagań, wynikający z wewnętrznych i zewnętrznych czynników ☺?

A. Nie ma takich rozwiązań – dlatego stosuje się coraz powszechniej metodyki agile
B. Stosowanie odpowiednich do sytuacji rozwiązań organizacyjnych i technicznych po stronie klienta lub po stronie dostawcy
C. Warsztaty wymagań z udziałem przedstawicieli klienta i dostawcy
D. Wybór jednego z 14 języków modelowania UML, aby wymagania stały się przejrzyste i jednoznaczne

2.2.5 UML (Unified Modelling Language)

A. Został zaprojektowany, aby wspierać modelowanie struktur danych przy tworzeniu baz danych
B. To inna nazwa metody modelowania przy pomocy przypadków użycia i scenariuszy biznesowych
C. Składa się z następujących elementów: aktorzy, przypadki użycia, asocjacje (powiązania między aktorami a przypadkami użycia) oraz scenariusze przypadków użycia
D. Jest zestawem języków modelowania do opisu działania systemów, struktur danych i struktur systemów

2.2.6 SysML (System Modelling Language)

A. To podzbiór i jednocześnie rozwinięcie języka UML, bardziej generyczny niż UML, który ma szereg rozwiązań dostosowanych w pierwszym rzędzie do oprogramowania
B. Składa się z diagramów klas, diagramów BPMN oraz diagramów korespondencji
C. Jest łatwy do nauczenia się i przeznaczony przede wszystkim dla prostych systemów oraz dla niedoświadczonych analityków
D. Jest przeznaczony do modelowania wymagań poza-funkcjonalnych

2.2.7 Przypadki użycia (*Use Cases*)

A. Służą do opisu i modelowania wysokopoziomowych scenariuszy biznesowych
B. To diagramy plus scenariusze przypadków użycia
C. Najlepiej sprawdzają się przy modelowaniu diagramów przejść stanów dla systemów wbudowanych czasu rzeczywistego
D. Pokazują kolejność oraz czas przepływu sygnałów między elementami lub użytkownikami systemu

2.2.8 Interesariusz (*stakeholder*)

A. Interesariusze, to użytkownicy końcowi oraz administratorzy systemu po stronie klienta
B. Interesariusze różnią się między sobą, między innymi znaczeniem lub zaangażowaniem w proces wymagań
C. Interesariusz nie powinien mieć wpływu na implementację systemu, jego rola powinna ograniczać się do potrzeb i wymagań
D. Pojęcie interesariuszy odnosi się do systemów komercyjnych, podczas gdy dla systemów IT w sektorze publicznym, zastępuje je prawo zamówień publicznych (*public procurement law*)

3. Zarządzanie procesem oraz ryzykiem

3.1 Zarządzanie projektem

3.1.1 Co powoduje, że błędy wymagań są główną przyczyną niepowodzeń projektów IT?

A. Zwykle klient ani użytkownicy nie są dostatecznie zainteresowani powodzeniem projektów IT
B. Wymagania określają cel projektu i wynikają z potrzeb biznesowych; niepoprawne wymagania, to praca bez celu
C. Klienci często nie są w stanie formułować poprawnych wymagań, z powodu braku kompetencji informatycznych
D. Wymagania bywają sprzeczne lub nieprecyzyjne

3.1.2 w jakich **fazach projektu** potrzebne są wymagania? (w kolejności chronologicznej) [tak, to idiotyczne pytanie ☹]

A. Inicjacja projektu, realizacja projektu, wdrożenie produktu, informacja zwrotna
B. Rozpoczęcie projektu, weryfikacja, walidacja, wdrożenie
C. Identyfikacja projektu, analiza projektu, zarządzanie projektem, walidacja projektu
D. Rozpoczęcie projektu, negocjowanie kontraktu, definiowanie projektu, realizacja projektu

3.1.3 Podczas definicji projektu:

A. Inżynieria wymagań dostarcza danych wejściowych do projektowania między innymi planu testów i planu projektu
B. Następuje szczególnie intensywne poszukiwanie interesariuszy oraz próby zdefiniowania celu biznesowego projektu
C. W tej fazie proces inżynierii wymagań jest już zakończony i nie ma wpływu na dalsze działania
D. Faza definicji projektu i związane z nią działania inżynierii wymagań pojawiają się w projektach sekwencyjnych (np. kaskadowym), ale nie w projektach iteracyjnych

3.1.4 Jakie są cechy procesu inżynierii wymagań w następujących rodzajach projektów?

A. „Słonie" – wymagają obszernej dokumentacji wymagań; „dzikie koty" – stosują minimalną dokumentację
B. „Konie" – stosują średni poziom formalizacji procesu inżynierii wymagań; „króliki" w ogóle nie stosują inżynierii wymagań [jest ona tam uważana za archaiczną]
C. „Króliki" wykorzystują iteracyjne formy pracy z wymaganiami, a „konie" średni poziom formalizacji procesu wymagań
D. „Dzikie koty" mają mały udział w szybko rosnącym rynku, a „słonie" mają duży udział w stabilnym rynku

3.1.5 Dokument, określający między innymi: *zakres inżynierii wymagań, podział ról i odpowiedzialności, procedury oraz agendę działań inżynierii wymagań*, nosi nazwę:

A. Dokument zakresu oraz wizji (*Vision and Scope Document*)
B. Wysokopoziomowa specyfikacji wymagań (*High-level Requirements Specification*)
C. Plan zarządzania wymaganiami (*Requirements Management Plan*)
D. Strategia inżynierii wymagań (*Requirements Engineering Strategy*)

3.2 Zarządzanie ryzykiem

3.2.1 Podstawy zarządzania ryzykiem

3.2.1.1 Czym jest ryzyko?

A. Są dwie spotykane definicje ryzyka: (1) jako możliwość nastąpienia niepewnych wydarzeń, korzystnych lub nie; lub (2) jako możliwość następowania niekorzystnych lub niebezpiecznych wydarzeń
B. Ryzyko, to niepewność, co do terminu ukończenia projektu oraz zakresu funkcjonalności produktu
C. Są dwie spotykane definicje ryzyka: (1) jako prawdopodobieństwo nastąpienia niepewnych wydarzeń, korzystnych lub nie; lub (2) jako konsekwencje następowania niekorzystnych lub niebezpiecznych wydarzeń
D. Ryzyko składa się z zagrożeń wobec realizacji wymagań funkcjonalnych i poza-funkcjonalnych dla systemów IT

3.2.1.2 Klasyfikacje ryzyka

A. **Techniczne ryzyko produktowe** to niepewność, co do jakości realizacji wymagań funkcjonalnych dla produktu
B. **Ryzyko projektowe** to możliwość opóźnień w realizacji projektu, lub niepełnej funkcjonalności produktu
C. Przykładem **technicznego ryzyka projektowego** jest np. niedostępność nowej technologii w przewidywanym czasie
D. **Ryzyko produktowe** oznacza, możliwość wystąpienia trudności technicznych w projektach

3.2.1.3 Która z poniższych metod **NIE** jest metodą identyfikacji ryzyka?

A. Ocena zagrożeń, wykonana przez niezależnych doradców [konsultantów]
B. Warsztaty ryzyka
C. Burza mózgów
D. Projektowanie obiektowe

3.2.1.4 Jak obliczać wielkość ryzyka?

A. Poprzez wywiady z ekspertami
B. Jako funkcję prawdopodobieństwa i skutków materializacji ryzyka
C. Według krzywej Boehm'a – ryzyko jest tym większe, im w późniejszej fazie projektu się ujawnia
D. Jako zmienną zależną od konsekwencji ryzyka oraz złożoności oprogramowania

3.2.1.5 Które z poniższych zestawień metod redukcji | zmniejszenia ryzyka (*mitigation*) jest poprawne (w znaczeniu: zgodne z sylabusem)?

A. Unikanie (*avoidance*), powstrzymywanie (*retention*), dzielenie się ryzykiem (*sharing*), zmniejszanie skutków (*reduction*)
B. Unikanie (*avoidance*), delegowanie (delegating), powstrzymywanie (*retention*), usuwanie (*deletion*)
C. Zmniejszanie skutków (*reduction*), analizowanie (*analysis*), monitorowanie (*monitoring*), podział (*sharing*)
D. Lokalizowanie (*localization*), przewidywanie (*prediction*), zmniejszenie skutków (*reduction*), usuwanie (*deletion*)

3.2.1.6 Według ISO 31000 [rodzina standardów zarządzania ryzykiem], zarządzanie ryzykiem powinno – między innymi (według sylabusa! ☹):

A. Być systematyczne, uwzględniać czynnik ludzki oraz zlokalizowanie
B. Posługiwać się modelowaniem ryzyka z użyciem metod symulacyjnych
C. Określać zakres zastosowań systemu oraz jego cel biznesowy
D. Być systematyczne, uwzględniać czynnik ludzki oraz wykorzystywać najlepsze, dostępne dane

3.2.1.7 Monitorowanie (*monitoring*) ryzyka i radzenie sobie z ryzykiem (*resolving*):

A. Są konieczne z powodu zmienności ryzyka w czasie
B. Monitorowanie pozwala brać pod uwagę zmiany ryzyka, a rozwiązywanie – zmniejszać jego skutki
C. Rozwiązywanie pozwala brać pod uwagę zmiany ryzyka, a monitorowanie – zmniejszać jego skutki
D. Zarówno monitorowanie, jak i rozwiązywanie, stosowane są w projektach iteracyjnych oraz dynamicznych

3.2.1.8 Proszę znaleźć **niepoprawne** zdanie:

A. Unikanie (*avoidance*) ryzyka polega na niepodejmowaniu ryzykownych działań, a dzielenie się ryzykiem (*sharing*) na przypisaniu odpowiedzialności za nie innym osobom
B. Powstrzymywanie (*retention*) ryzyka, to podejmowanie działań zmniejszających jego prawdopodobieństwo, a zmniejszanie skutków (*reduction*), to planowanie i ewentualne podejmowanie działań awaryjnych
C. Unikanie (*avoidance*) ryzyka polega na niepodejmowaniu ryzykownych działań, a dzielenie się ryzykiem (*sharing*) na ignorowaniu zagrożenia
D. Testowanie oprogramowanie służy do powstrzymywania (*retention*) ryzyka, a planowanie sposobów obejścia awarii, to zmniejszanie skutków (*reduction*) ryzyka

3.2.1.9 Analiza rodzajów i skutków błędów (FMEA – *Failure Mode and Effect Analysis*)

Uwaga! ISTQB – system certyfikacji dla testerów – przyjmuje następującą terminologię:

- **Pomyłka** lub **błąd** (*mistake* albo *error*): ludzka lub procesowa pomyłka (nieprawidłowe działanie), powodująca niepoprawny wynik, czyli **defekt**;
- **Defekt** (*defect*) – wada konstrukcyjna oprogramowania („bug"), albo wadliwa informacja np. w dokumencie („niezgodność");
- **Awaria** (*failure*) – niepoprawne działanie oprogramowanie, spowodowane **defektem.**

Jeśli stosować tę terminologię, FMEA nazywałoby się „DMFA" (*defect mode and failure analysis*), czyli analizą rodzajów defektów i (ich skutków) awarii.

A. FMEA służy do planowania działań, pozwalających identyfikować możliwe skutki awarii systemów krytycznych dla bezpieczeństwa
B. FMEA, to metoda statystyczna, służąca do projektowania działań mających na celu powstrzymywanie ryzyka poprzez optymalizację procedur
C. FMEA, tak samo, jak inne sposoby zarządzania ryzykiem, powinna być stosowana w fazie realizacji projektu
D. FMEA klasyfikuje rodzaje błędów [czyli defektów] biorąc pod uwagę ich prawdopodobieństwo, skutki oraz łatwość ich wykrycia

3.2.1.10 Zalety i wady stosowania FMEA [Please RTFS! – *read the -ing syllabus* ☺]

A. Wadą FMEA jest jej pracochłonność, a jedną z zalet – zwiększenie zadowolenia klientów
B. Wady FMEA, to między innymi jej formalizm i nieopłacalność, a zalety – to między innymi minimalizacja późnych zmian i metodyczność
C. Wadą FMEA jest konieczność tworzenia obszernej dokumentacji, a zaletą – stosowanie formalnych technik opisywania wymagań
D. Wadą FMEA jest jej niedostosowanie do metodyk iteracyjnych, a zaletami – ulepszanie produktu oraz motywacji uczestników zespołu

3.2.2 Ryzyka związane z wymaganiami

3.2.2.1 **Najczęstsze** ryzyka, związane z wymaganiami, to:

A. Nadmierna szczegółowość oraz zbyt fachowe słownictwo
B. Brak niektórych wymagań oraz trudności w porozumiewaniu się (*communication*)
C. Brak odniesień wymagań do rozwiązań technicznych oraz niedostateczna asertywność inżynierów wymagań
D. Niska jakość opisu (specyfikacji, dokumentacji) wymagań oraz nadmierne śladowanie (śledzenie powiązań, *traceability*), zwiększające koszty zmian

3.2.2.2 Zmniejszanie zagrożeń, związanych z wymaganiami:

A. Jest opłacalne tylko w warunkach dużej zmienności wymagań
B. To między innymi stosowanie dobrych metod pozyskiwania (identyfikacji) wymagań
C. To między innymi analiza wpływu oraz posługiwanie się precyzyjną informatyczną terminologią
D. Wymaga stosowania języka naturalnego do opisu wymagań

3.2.3 Zmniejszanie ryzyka przy pomocy prototypowania

3.2.3.1 Korzyści prototypowania

A. Możliwość wdrożenia prototypu bezpośrednio jako wersji produkcyjnej
B. Możliwość otrzymania informacji zwrotnej oraz wczesna identyfikacja błędów wydajności
C. Ułatwienie określenia celu biznesowego dla tworzonego systemu IT
D. Pomoc przy doprecyzowywaniu wymagań dla produktu

3.3 Role w zarządzaniu wymaganiami

3.3.1 Jaka jest różnica między analitykiem systemowym, a inżynierem wymagań? [to krańcowa bzdura, ale tako rzecze Zaratustra…. to znaczy sylabus ☹]

A. Analityk systemowy określa wymagania biznesowe, natomiast inżynier wymagań – wymagania techniczne
B. Rolą inżyniera wymagań jest tworzenie **wymagań systemowych** z wymagań biznesowych, natomiast rolą analityka systemowego jest projektowanie **rozwiązania** systemu
C. Analityk systemowy prowadzi negocjacje z interesariuszami, natomiast inżynier wymagań przede wszystkim zajmuje się tworzeniem formalnych modeli rozwiązania
D. Analityk systemowy współpracuje bezpośrednio z kierownikiem projektu i z zespołem deweloperskim, natomiast inżynier wymagań współpracuje z analitykiem biznesowym i użytkownikami

3.3.2 Jaki jest – według sylabusa ☹ - podział funkcji między konstruktorem (*requirements developer*) a menedżerem wymagań (*requirements manager*)?

A. Menedżer wymagań zarządza całościowo procesem inżynierii wymagań, podczas gdy konstruktor wymagań zajmuje się głównie identyfikacją, analizą i walidacją wymagań
B. Menedżer wymagań wydaje polecenia konstruktorowi wymagań
C. Menedżer wymagań zajmuje się zarządzaniem zmianami i śladowaniem (*tracing*) wymagań, natomiast konstruktor wymagań – ich modelowaniem
D. Menedżer wymagań zbiera wymagania od interesariuszy i przekazuje je konstruktorowi wymagań do specyfikowania

3.3.3 Proszę znaleźć **niepoprawne** [w sensie sylabusa ☹ oczywiście] zdanie:

A. Menedżer konfiguracji (*Configuration Manager*) utrzymuje macierz zaakceptowanych wymagań, a Menedżer Jakości (*Quality Manager*) dba o to, aby istniał i był przestrzegany właściwy proces inżynierii wymagań
B. Rada Kontroli Zmian (*Change Control Board - CCB*) zajmuje się analizą wpływu (*impact analysis*) oraz ocenia wykonalność (*feasibility*) proponowanych wymagań
C. Kierownik testów (*Test Manager*) odpowiada za wykonywanie testów, a kierownik jakości (*Quality Manager*) za tworzenie strategii testowej dla projektu
D. Inżynier wymagań współpracuje z klientem, aby zidentyfikować szczegółowe wymagania, doprecyzowujące wymagania biznesowe, określone przez analityka biznesowego

4. Identyfikacja wymagań

4.1 Klient

4.1.1 Kiedy – między innymi - **niezbędny** jest udział przedstawicieli klienta w procesie inżynierii wymagań?

A. Przy tworzeniu dużych, złożonych systemów IT
B. Kiedy tworzy się oprogramowanie na zamówienie
C. W procesie tworzenia powiązań między wymaganiami biznesowymi a technicznymi
D. W projektach realizowanych według metodyki agile scrum

4.1.2 Jaki jest związek kontraktu / umowy z inżynierią wymagań?

A. Kontrakt powinien zawierać wszystkie szczegółowe wymagania wobec systemu
B. Częścią kontraktu są wymagania wysokopoziomowe oraz lista uczestników projektu po stronie dostawcy
C. Kontrakt zawiera wymagania wysokopoziomowe, kryteria ich akceptacji oraz listę dostaw
D. Negocjowanie kontraktu jest częścią procesu analizy i walidacji wymagań

4.2 Identyfikacja interesariuszy

4.2.1 Kim są interesariusze?

A. Interesariuszami projektu są osoby związane z projektem
B. Interesariusze, to użytkownicy i sponsorzy systemu, oraz uczestnicy projektu
C. Interesariusze, to osoby fizyczne, lub prawne, na które projekt ma wpływ, lub które mają wpływ na projekt
D. Interesariusze produktu nie są interesariuszami projektu, i odwrotnie

4.2.2 Przeczytajcie sobie proszę ten kawałek o typowych kategoriach interesariuszy w sylabusie, bo już mi od tego mózg staje w poprzek ☹

Kto jest, a kto nie jest interesariuszem projektu tworzenia nowej gry on-line? [Uwaga! To pytanie, to SUCHAR]

A. Teściowa menedżera jakości? (prawda | fałsz)
B. Adwokat, reprezentujący prezesa konkurencyjnej firmy w jego sprawie rozwodowej? (prawda | fałsz)
C. Autor artykułu w „Computerworld" pod tytułem „Nowe trendy w grach on-line"? (prawda | fałsz)
D. Asystentka menedżera działu, w którym projekt wytwórczy wynajmuje stanowiska do testów kompatybilności tej gry? (prawda | fałsz)

4.2.3 Analityk testów, zatrudniony przez organizację Klienta, będący członkiem zespołu projektowego dostawcy, z zadaniem projektowania, wykonywania i raportowania wyników testów akceptacyjnych, to:

A. Interesariusz zewnętrzy
B. Interesariusz, będący przyszły użytkownikiem systemu
C. Interesariusz zewnętrzny względem organizacji dostawcy, wewnętrzny względem zespołu projektowego
D. Interesariusz pośredni

4.2.3 Na czym polega „analiza funkcji" według sylabusa? [sorry, folks, na ten temat sylabus bredzi ☹, ale egzamin, to egzamin!]

A. Na analizowaniu funkcjonalności rozwiązania z punktu widzenia różnych interesariuszy tych funkcji
B. Na identyfikowaniu potrzebnych funkcjonalności rozwiązania przy pomocy analizy potrzeb biznesowych różnych interesariuszy
C. Na identyfikowaniu przedstawicieli zewnętrznych interesariuszy ze strony klienta, dla interesariuszy grupowych
D. Na analizie punktów funkcyjnych

4.2.4 Kwestionariusz | ankieta z pytaniami, wspierającymi identyfikację interesariuszy oraz ich potrzeb:

Proszę w tym celu przeczytać „Techniki przesłuchań" (http://www.student.lex.pl/czytaj/-/artykul/techniki-przesluchan) ☺ oraz akapit na stronach 41-42 sylabusa (wersja 2011, nie mylić z http://reqb.org/docs/REQB_ALRM_Syllabus.pdf!)

4.2.5 Interesariusze główni (*baseline*) i poboczni (*satellte*):

A. Interesariusze główni to użytkownicy systemu, a interesariusze poboczni, to ich współpracownicy
B. Interesariusze główni, to użytkownicy oraz konstruktorzy (deweloperzy) systemu, osoby odpowiedzialne finansowo za produkt, a także osoby prawne (organizacje), w sferze których system się znajduje
C. Interesariusze poboczni, to osoby, które nie używają systemu bezpośrednio, ale na których pracę system wywiera wpływ
D. Identyfikacja interesariuszy pobocznych nie wymaga wcześniejszej identyfikacji interesariuszy głównych

4.3 Techniki identyfikacji wymagań

4.3.1 Połącz ze sobą poniższe pojęcia oraz ich opisy

(a) Pytania zamknięte	(i) Np. skala 1, 2, 3, 4 […]
(b) Pytania otwarte	(ii) Np. pytania zaczynające się od „Czy… (tak | nie)"
(c) Skala dychotomiczna	(iii) Np. skala, której wartości, to liczby rzeczywiste
(d) Skala polinomiczna, nominalna	(iv) Np. skala „zielony", „czerwony", „żółty", […]
(e) Skala polinomiczna, porządkowa	(v) Np. pytania zaczynające się od „Opisz, jak…"
(f) Skala ciągła	(vi) Np. skala kobieta | mężczyzna

A. (a) – (i), (b) – (v), (c) – (vi), (d) – (iv), (e) – (ii), (f) – (iii)
B. (a) – (iii), (b) – (v), (c) – (vi), (d) – (iv), (e) – (i), (f) – (ii)
C. (a) – (vi), (b) – (v), (c) – (ii), (d) – (iv), (e) – (i), (f) – (iii)
D. (a) – (ii), (b) – (v), (c) – (vi), (d) – (iv), (e) – (i), (f) – (iii)

4.3.2 Która z poniższych list technik pozyskiwania wymagań jest **niepoprawna** (to znaczy, że zawiera co najmniej jedno określenie, które nie oznacza techniki identyfikacji wymagań)?

A. Ankieta (kwestionariusz), wywiad, ponowne użycie specyfikacji
B. Burza mózgów, obserwacja w polu, czeladnictwo (terminowanie), gra symulacyjna
C. Warsztat, ponowne użycie, poszukiwanie w dokumentach („archeologia")
D. Reprezentant klienta w projekcie, samo-rejestracja, wywiad, ponowne użycie specyfikacji

---

Ciąg dalszy w częściach 2 i 3

5. Specyfikacja i dokumentacja wymagań

6. Analiza wymagań