

# Syllabus

**REQB®**

## **Certyfikowany Profesjonalista Inżynierii Wymagań**

Poziom Podstawowy



**Requirements  
Engineering**  
Qualifications Board

Wersja 1.3

2011

## Historia zmian

Wersja	Data	Komentarz
0.1	17.04.2006	Pierwsza wersja sylabusu; stworzenie podstawowej struktury sylabusu
0.2	20.07.2006	Rozszerzenie wersji 0.1
0.3	04.09.2006	Dalsze rozszerzenie i rewizja wersji 0.1
0.4	10.10.2006	Rewizja wersji 0.3
0.5	15.12.2006	Rewizja wersji 0.4
0.6	07.02.2007	Kompletnie zrewidowana wersja 0.5
0.7	10.04.2007	Wersja zrewidowana dla przeglądu
0.8	15.06.2007	Wersja Alpha
0.9	01.09.2007	Wersja Beta
1.0	15.01.2008	Wydanie 1.0
1.1	29.05.2008	Zaktualizowana wersja 1.1
1.2	01.07.2008	Zaktualizowana wersja 1.2
1.3	10.07.2011	Zaktualizowana wersja 1.3
1.3PL	25.07.2011	Tłumaczenie wersji 1.3

## Myśl przewodnia

Głównym tematem tego syllabusu jest ciągle rosnąca złożoność oprogramowania i nasza zależność od niego. W rezultacie pojawia się wysoki poziom zależności od niezawodności oprogramowania. Requirements Engineering Qualifications Board (REQB) postanowił zatem stworzyć jednolite międzynarodowe standardy w dziedzinie Inżynierii Wymagań. Standardy są jednak jak języki - tylko wtedy, gdy je zrozumiesz, możesz pracować efektywnie. Aby stworzyć taki jednolity język w tak ważnej dziedzinie, jak Inżynieria Wymagań, międzynarodowi eksperci zebrali się w REQB i opracowali niniejszy syllabus.

## Podziękowania

Grupa robocza syllabusu Poziomu Podstawowego REQB (edycja 2011): Karolina Zmitrowicz (przewodnicząca), Alain Betro, Dorothée Blocks, Jérôme Khoualed, Eric Riou Du Cosquer, Chris Hofstetter, Michał Figarski, Francine Lafontaine, Beata Karpińska, Folke Nilsson, Ingvar Nordström, Alain Ribault, Radosław Smilgin.

Oryginalna wersja syllabusu została przetłumaczona przez zespół polskiego oddziału gasq w składzie: Karolina Zmitrowicz, Michał Figarski, Krzysztof Chytła.

## Spis treści

Spis treści.....	4
Wprowadzenie .....	9
1 Podstawy (K2) .....	11
1.1 Wymaganie (K2).....	12
1.1.1 Definicja i klasyfikacja (K2).....	12
1.1.2 Problemy związane z wymaganiami (K1).....	14
1.1.3 Kryteria jakościowe dla wymagań (K2).....	14
1.1.4 Rozwiązanie (K1).....	15
1.1.5 Zobowiązanie (K1) .....	15
1.1.6 Odpowiedzialność prawna i defekty (K1) .....	16
1.1.7 Priorytet i krytyczność wymagań (K1) .....	16
1.1.8 Weryfikacja i walidacja (K1).....	17
1.1.9 Inżynieria Wymagań, Zarządzanie Wymaganiami i Rozwijanie Wymagań (K2) .....	17
1.2 Standardy i normy (K1) .....	19
1.2.1 Standardy (K1) .....	19
1.2.2 Normy procesowe (K1) .....	20
1.2.3 Powody zaniedbywania Inżynierii Wymagań (K2).....	20
2 Modele procesu i proces Inżynierii Wymagań (K2) .....	22
2.1 Modele procesu (K2).....	23
2.1.1 Modele procesu (K2).....	23
2.1.2 Ogólny model V (K2) .....	24
2.1.3 Rational Unified Process (RUP®) (K2) .....	24
2.1.4 Podejścia zwinne.....	25
2.1.5 Programowanie Ekstremalne (K2) .....	25
2.1.6 Scrum (K2).....	26
2.1.7 Model Dojrzałości (K2).....	27
2.2 Proces Inżynierii Wymagań (K2).....	29
2.2.1 Definicja Procesu Inżynierii Wymagań (K2) .....	29
2.2.2 Czynniki wpływające na Inżynierię Wymagań .....	29
3 Zarządzanie Projektem i Ryzykiem (K2) .....	31

3.1	Zarządzanie Projektem (K2) .....	32
3.1.1	Potrzeba Inżynierii Wymagań w projektach (K2).....	32
3.1.2	Jakie błędy mogą wystąpić w Inżynierii Wymagań? (K2).....	33
3.2	Zarządzanie Ryzykiem (K2).....	34
3.2.1	Potrzeba Zarządzania Ryzykiem (K2) .....	34
3.2.2	Ryzyko (K2).....	34
3.2.3	Zarządzanie Ryzykiem (K2) .....	35
3.2.4	Analiza Trybów i Efektów Awarii (K2).....	36
4	Odpowiedzialności i role (K2) .....	38
4.1	Podstawowe Role (K1) .....	39
4.1.1	Podstawowe role (K2).....	39
4.1.2	Interesariusz (K2) .....	40
4.2	Zadania Inżynierii Wymagań (K2).....	42
4.2.1	Zadania Inżynierii Wymagań (K2) .....	42
4.2.2	Wiedza Profesjonalisty Inżynierii Wymagań (K1) .....	42
5	Identyfikacja Wymagań (K2) .....	43
5.1	Klient (K1).....	44
5.1.1	Klient (K1) .....	44
5.1.2	Kontrakt (K2).....	44
5.2	Wizja i Cele Projektu (K2).....	46
5.2.1	Wizja (K2).....	46
5.3	Identyfikacja Interesariuszy (K2).....	48
5.3.1	Identyfikacja Interesariuszy (K2).....	48
5.3.2	Procedura identyfikacji i oceny interesariuszy (K2).....	48
5.4	Techniki Identyfikacji Wymagań (K2).....	49
5.4.1	Cel identyfikacji wymagań (K2).....	49
5.4.2	Techniki (K1) .....	49
5.5	Wymagania Funkcjonalne i Niefunkcjonalne (K2) .....	55
5.5.1	Wymagania funkcjonalne (K2).....	55
5.5.2	Wymagania niefunkcjonalne (K2).....	55
5.6	Opis Wymagań (K2).....	57
5.6.1	Opis Wymagań (K2) .....	57

5.6.2	Procedura Konstrukcji Wymagania (K3) .....	57
5.6.3	Dokument wymagań (K2) .....	59
6	Specyfikacja Wymagań (K2).....	60
6.1	Specyfikacja (K2) .....	61
6.1.1	Specyfikacja (K1) .....	61
6.1.2	Specyfikacja Wymagań (K2).....	61
6.1.3	Historyjki Użytkownika (K2).....	62
6.1.4	Specyfikacja Rozwiązania (K2) .....	62
6.1.5	Ważne standardy (K1).....	63
6.2	Procedura (K3) .....	64
6.2.1	Procedura Specyfikacji Rozwiązania (K3).....	64
6.3	Formalizacja (K2).....	65
6.3.1	Stopnie formalizacji (K2).....	65
6.4	Jakość Wymagań (K2).....	66
6.4.1	Wprowadzenie.....	66
6.4.2	Miary dla doskonalenia jakości i zapewnienia jakości wymagań (K2) .....	66
7	Analiza Wymagań (K2).....	68
7.1	Wymagania i Rozwiązania (K1) .....	69
7.1.1	Cel Analizy Wymagań (K2) .....	69
7.1.2	Procedura Analizy Wymagań (K2) .....	69
7.1.3	Różnica strukturalna pomiędzy wymaganiami a rozwiązaniami (K2).....	69
7.2	Metody i Techniki (K2) .....	70
7.2.1	Metody i modele analizy .....	70
7.2.2	Typy modeli (K2) .....	71
7.2.3	Różne perspektywy systemu (K2).....	71
7.2.4	Różne modele (K1).....	72
7.3	Analiza Obiektowa (K2) .....	74
7.3.1	UML (K1) .....	74
7.3.2	SysML (K2).....	76
7.4	Szacowanie Kosztów (K2).....	77
7.4.1	Typy estymat (K2) .....	77
7.4.2	Czynniki wpływające na koszty wytwarzania (K2) .....	77

7.4.3	Podjęcia do szacowania kosztu .....	78
7.5	Priorytetyzacja (K2) .....	82
7.5.1	Priorytetyzacja (K2).....	82
7.5.2	Procedura ustalania priorytetów (K2) .....	82
7.5.3	Skale priorytetowe (K2) .....	83
7.6	Akceptacja Wymagań (K2) .....	84
7.6.1	Akceptacja (K2) .....	84
7.6.2	Korzyści z akceptacji wymagań (K1).....	85
8	Śledzenie Wymagań (K2) .....	86
8.1	Śledzenie w Projekcie (K2) .....	87
8.1.1	Rozwój wymagań (K1).....	87
8.1.2	Możliwość śledzenia (K2).....	87
8.1.3	Typy możliwości śledzenia (K2).....	88
8.2	Zarządzanie Zmianą (K2) .....	89
8.2.1	Zmiany wymagań (K1).....	89
8.2.2	Zarządzanie Zmianą (K2).....	89
8.2.3	Żądanie Zmiany (K2) .....	90
8.2.4	Komitet Kontroli Zmian (K1) .....	90
8.2.5	Cykl życia wymagania (K2) .....	91
8.2.6	Rozróżnienie pomiędzy Zarządzaniem Defektami a Zarządzaniem Zmianą (K2) .....	91
8.2.7	Wpływ zmiany na projekt (K2).....	91
9	Zapewnienie Jakości (K2) .....	93
9.1	Czynniki Oddziałujące (K1) .....	94
9.1.1	Czynniki oddziałujące na Inżynierię Wymagań (K1) .....	94
9.2	Weryfikacja wymagań w fazie pozyskiwania wymagań (K2) .....	95
9.3	Zapewnienie Jakości poprzez Testowalność (K2).....	96
9.3.1	Inżynieria Wymagań a testowanie (K2) .....	96
9.3.2	Kryteria Akceptacji (K2) .....	96
9.3.3	Metody dla testowania (K2) .....	96
9.3.4	Wymagania a proces testowy (K2) .....	97
9.4	Metryki (K2).....	99
9.4.1	Metryka (K1) .....	99

9.4.2	Metryki dla wymagań (K1)	99
9.4.3	Pomiar jakości wymagań (K2)	100
10	Narzędzia (K2)	101
10.1	Korzyści Narzędzi (K2)	102
10.1.1	Zastosowanie narzędzi w Inżynierii Wymagań (K2)	102
10.1.2	Korzyści z używania narzędzi (K2)	102
10.2	Kategorie Narzędzi (K2)	103
10.2.1	Kategorie Narzędzi (K2)	103
11	Literatura	105
12	Index	108

## Wprowadzenie

### Cel sylabusu

Niniejszy sylabus definiuje poziom podstawowy (*Foundation Level*) programu szkolenia umożliwiającego zostanie Certyfikowanym Profesjonalistą Inżynierii Wymagań REQB (w skrócie CPRE - Certified Professional for Requirements Engineering). Sylabus ten został opracowany przez REQB we współpracy z Global Association for Software Quality. Przedmiotem REQB są wszelkiego typu produkty związane z IT, gdzie produkt może obejmować sprzęt oraz usługi, jak i oprogramowanie oraz jego zatwierdzonymi wymaganiami oraz wymaganiami biznesowymi wraz z dokumentacją.

Syllabus ma służyć jako podstawa dla dostawców szkoleń, którzy chcieliby pozyskać akredytację trenerską. Wszystkie obszary niniejszego sylabusu muszą być włączone do materiałów szkoleniowych. Sam sylabus powinien jednak również służyć jako przygotowanie do certyfikacji. Wszystkie obszary wymienione tutaj są zatem istotne dla egzaminu, do którego można podchodzić albo po ukończeniu akredytowanych kursów, albo podczas egzaminów otwartych.

Syllabus podaje również rekomendowany czas trwania kursu dla każdego rozdziału.

### Egzamin

Egzamin na poziom podstawowy Certyfikowanego Profesjonalistę Inżynierii Wymagań odbywa się na podstawie niniejszego sylabusu, dlatego wszystkie sekcje sylabusu mogą podlegać weryfikacji. Pytania egzaminacyjne są rozdzielone na poszczególne sekcje. Jedno pytanie może odnosić się do kilku sekcji sylabusu.

Format egzaminu to test wielokrotnego wyboru.

Do egzaminu można podejść po uczestnictwie w akredytowanym kursie lub w bez wcześniejszego kursu podczas otwartych egzaminów. Szczegółowe informacje dotyczące terminów egzaminów można znaleźć na stronie internetowej gasq ([www.gasq.org](http://www.gasq.org)) lub na stronie internetowej REQB ([www.reqb.eu](http://www.reqb.eu)).

### Akredytacja

Dostawcy kursu na poziom podstawowy Certyfikowanego Profesjonalisty Inżynierii Wymagań REQB muszą być akredytowani przez Global Association for Software Quality. Eksperti GASQ dokonują przeglądu dokumentacji szkoleniowej pod kątem zgodności z sylabusem. Kurs akredytowany jest uznawany jako zgodny z sylabusem. Na zakończenie takiego kursu, może zostać przeprowadzony oficjalny egzamin na Certyfikowanego Profesjonalistę Inżynierii Wymagań (egzamin CPRE). Egzamin jest przeprowadzany przez niezależny instytut certyfikacji (zgodnie z zasadami normy ISO 17024).



Akredytowane ośrodki szkoleniowe są identyfikowane poprzez oficjalne logo akredytowanego dostawcy szkoleń REQB.

### **Międzynarodowość**

Niniejszy syllabus został opracowany we współpracy grupy międzynarodowych ekspertów, dlatego też jego zawartość może być postrzegana jako międzynarodowy standard. Tym samym syllabus umożliwia szkolenie i egzaminowanie na arenie międzynarodowej na jednakowym poziomie.

### **Poziomy nauczania**

Cele nauczania w niniejszym syllabusie zostały podzielone na różne poziomy poznawcze (K). Pozwala to kandydatowi rozpoznać „poziom wymaganej wiedzy” każdego punktu.

W niniejszym syllabusie istnieją 3 K-poziomy:

- K1 – zapamiętaj, rozpoznaj, przypomnij
- K2 – zrozum, wyjaśnij, podaj powód, porównaj, sklasyfikuj, podsumuj
- K3 – zastosuj w konkretnej sytuacji

<b>1 Podstawy (K2)</b>	<b>40 minut</b>
------------------------	-----------------

*Cele nauczania dla Poziomu Podstawowego Inżynierii Wymagań*  
Cele określają wiedzę i umiejętności, uzyskiwane po ukończeniu każdego modułu.

### **1.1 Wymaganie (K2)**

- LO-1.1.1 Podać definicję wymagania (K1)
- LO-1.1.2 Wyjaśnić, jakie jest znaczenie i cel wymagań (K2)
- LO-1.1.3 Wyjaśnić, jak można sklasyfikować wymagania (K2)
- LO-1.1.4 Opisać różne typy wymagań (K2)
- LO-1.1.5 Wyjaśnić, jakie problemy pojawiają się w związku z wymaganiami (K1)
- LO-1.1.6 Opisać, jakie koncepcje są ważne w powiązaniu z wymaganiami (K2)
- LO-1.1.7 Wyjaśnić, jaka jest różnica między Zarządzaniem Wymaganiami a Inżynierią Wymagań (K2)

### **1.2 Standardy i normy (K1)**

- LO -1.2.1 Podać normy i standardy, które wiążą się z wymaganiami (K1)
- LO-1.2.2 Wyjaśnić, dlaczego Inżynieria Wymagań jest ważna (K2)

<b>1.1 Wymaganie (K2)</b>	<b>40 minut</b>
---------------------------	-----------------

## Pojęcia

Krytyczność, Inżynieria Wymagań, Priorytet, Rozwiązanie, Walidacja, Weryfikacja, Wymagania Procesowe, Wymagania Produktowe, Wymaganie, Wymaganie Funkcjonalne, Wymaganie Niefunkcjonalne, Zarządzanie Wymaganiami, Zobowiązanie

### 1.1.1 Definicja i klasyfikacja (K2)

Definicja tego, co jest rozumiane pod pojęciem „Wymaganie” (K1):

Wymaganie [IEEE 610.12]:

- Stan lub zdolność potrzebna użytkownikowi, aby rozwiązać problem lub osiągnąć cel.
- Stan lub zdolność, która musi zostać spełniona lub posiadana przez system lub moduł, aby spełnić kontrakt, standard, specyfikację lub inne formalne dokumenty.
- Reprezentacja w formie dokumentu stanu lub zdolności zgodnie z (1) lub (2)

Znaczenie i cel wymagań (K2):

- Podstawa dla oceny, planowania, przeprowadzania i monitorowania czynności projektowych
- Oczekiwania klienta
- Składnik umów, zamówień, planów projektowych itp.
- Ustanawianie granic systemu, zakresu dostawy, zakontraktowanych usług serwisowych

Klasyfikacja wymagań [Ebert05] (K2)

Wymagania dzielimy na:

- Wymagania procesowe
- Wymagania produktowe

Wymagania procesowe są związane z procesem wytwarzania oprogramowania. Obejmują na przykład: koszty, marketing, czas przetwarzania, sprzedaż i dystrybucję, organizację, dokumentację. Wymagania procesowe opisują potrzeby i ograniczenia procesów biznesowych.

Wymagania produktowe składają się z funkcjonalnych i niefunkcjonalnych wymagań produktowych. Oba rodzaje mogą być postrzegane z punktu widzenia użytkownika (zewnątrzne) lub klienta oraz z punktu widzenia zespołu wytwarzającego produkt (wewnętrzne).

*Uwaga: Użytkownik i klient mogą być różnymi osobami!*

Wymagania funkcjonalne opisują funkcję (zachowanie) systemu.

Wymagania niefunkcjonalne opisują atrybuty jakościowe systemu. Są często nazywane „atrybutami jakościowymi”.

Różnica między wymaganiami funkcjonalnymi i niefunkcjonalnymi może zostać opisana następującymi stwierdzeniami:

- Wymagania funkcjonalne opisują, co system robi
- Wymagania niefunkcjonalne opisują *jak* system to robi.

Przykłady:

Funkcjonalne wymagania produktowe z punktu widzenia użytkownika i klienta to: interfejs użytkownika, aplikacje, usługi.

Funkcjonalne wymagania produktowe z punktu widzenia zespołu deweloperów to: architektura, zasilanie, rozkład obciążenia.

Niefunkcjonalne wymagania produktowe z punktu widzenia użytkownika i klienta to: niezawodność, wydajność, użyteczność.

Niefunkcjonalne wymagania produktowe z punktu widzenia zespołu deweloperów to: testowalność, utrzymanialność, narzędzia.

Podstawowe typy wymagań (K1):

- Wymagania klienta
  - Życzenia, potrzeby i oczekiwania klienta.
  - Ograniczenia biznesowe
- Wymagania systemowe/specyficzne dla danego rozwiązania
  - Specyfikacja potrzeb klienta
- Wymagania produktowe/modułowe
  - Funkcje i charakterystyka rozwiązania
  - Podstawa do szczegółowej analizy i projektu (przykład: przypadki użycia systemu)

### 1.1.2 Problemy związane z wymaganiami (K1)

Najczęstsze problemy związane z wymaganiami to:

- Niejasne cele
- Problemy komunikacyjne
- Bariery językowe
- Bariery wiedzy
- Ogólnikowe sformułowania
- Zbyt formalne określenia
- Niestabilność wymagań
- Zła jakość wymagań (np., dwuznaczność, nadmierne specyfikowanie, niejasne, niemożliwe, sprzeczne wymagania)
- Zbytnie „ozłocenie”
- Niewystarczające zaangażowanie użytkownika
- Pominięte klasy użytkowników (skutkujące brakującymi interesariuszami)
- Nieprecyzyjne planowanie
- Minimalna specyfikacja

### 1.1.3 Kryteria jakościowe dla wymagań (K2)

Kryteria jakościowe dla wymagań [Wiegers05] są następujące(K2):

1. Każde wymaganie musi być:
  - Poprawne – wymaganie musi precyzyjnie opisywać dostarczaną funkcjonalność. Punktem odniesienia do sprawdzenia poprawności wymagania jest jego źródło (na przykład wymaganie systemowe klienta lub wyższego poziomu).
  - Wykonalne – wymaganie musi być możliwe do zaimplementowania zgodnie ze znanymi możliwościami i/lub ograniczeniami systemu i środowiska.
  - Niezbędne – wymaganie powinno dokumentować to, czego klient (lub inni interesariusze) naprawdę potrzebują, aby osiągnąć zewnętrzne wymaganie lub interfejs lub konkretny standard.
  - Spriorytetyzowane – wymaganie powinno mieć przydzielony priorytet wskazujący jak bardzo jest niezbędne dla danego wydania produktu.

- Jednoznaczne – wymaganie powinno być interpretowalne tylko w jeden sposób. Różne osoby czytające wymaganie powinny mieć tą samą interpretację i rozumienie wymagania.
- Weryfikowalne – wymaganie powinno być możliwe do zweryfikowania czy zostało poprawnie zaimplementowane.
- Jednostkowe – nie opisuje wielu wymagań. Zapewnia odpowiednią ziarnistość, aby opisać pojedyncze wymaganie.
- Niezależne od projektu – opisuje „Co” a nie „Jak”.

2. Specyfikacja wymagań musi być:

- Kompletna – nie powinno brakować żadnych wymagań ani niezbędnych informacji w specyfikacji wymagań. Kompletność jest także wyrażana jako oczekiwana charakterystyka pojedynczego wymagania i poziom jego szczegółowości.
- Spójna – wymaganie nie może być w sprzeczności z innymi wymaganiami dotyczącymi oprogramowania lub z wymaganiami wyższego poziomu (systemowymi lub biznesowymi).
- Modyfikowalna – specyfikacja musi dopuszczać wprowadzanie zmian do wymagań. Historia zmian każdego z wymagań powinna być przechowywana.
- Umożliwiająca śledzenie zmian – powinna być możliwość powiązania każdego wymagania z jego źródłem (np. wymaganiem systemowym wyższego poziomu, przypadkiem użycia, lub oczekiwaniem klienta) oraz z artefaktami powstałymi w fazie implementacji (np. elementów projektu, kodu źródłowego, przypadków testowych).

#### 1.1.4 Rozwiązanie (K1)

##### Rozwiązanie (K1)

Rozwiązanie jest implementacją wymagań.

Rozwiązanie może być oprogramowaniem, ulepszeniem procesu, itd.

#### 1.1.5 Zobowiązanie (K1)

##### Zobowiązanie (K1)

Zobowiązanie jest poziomem zobligowania się do spełnienia wymagań.

Zobowiązanie definiuje się za pomocą słów kluczowych:

- System powinien...

Słowa kluczowe mogą zawierać: „musi”, „będzie”, „powinien”, „mógłby”.

Słowa kluczowe “musi”, “powinien”, “mógłby” odnoszą się do biznesowych wymagań I wymagań użytkownika przed porozumieniem. Po porozumieniu i uzgodnieniu wymagań słowa kluczowe powinny bardziej wprost definiować poziom zobligowania do wykonania:

- System będzie...

W niektórych przypadkach wymaganie jest opisane jako poziom zobligowania oraz priorytet:

- System będzie... [priorytet = średni]
- System będzie ... [priorytet = niski]

Poziom zobligowania do wykonania wymagania może zostać wyrażony przy użyciu MoSCoW.

### 1.1.6 Odpowiedzialność prawna i defekty (K1)

#### Odpowiedzialność prawna (K1)

Istnieją odpowiedzialności prawne związane z jakością oprogramowania. Odpowiedzialności zazwyczaj wiążą się z defektami produktu.

Odpowiedzialność prawna powinna zostać zdefiniowana w kontrakcie pomiędzy dostawcą a klientem. Niektóre gałęzie przemysłu mogą wymagać spełnienia wymagań prawnych lub kontraktowych lub standardów specyficznych dla tej gałęzi przemysłu.

#### Usterka (defekt) (K1)

Błąd w module lub systemie, który może spowodować, że moduł lub system nie będzie w stanie wykonać oczekiwanej funkcji, np. błędna definicja danych lub deklaracja. Defekt napotkany podczas działania może spowodować awarię modułu lub systemu (ISTQB).

### 1.1.7 Priorytet i krytyczność wymagań (K1)

#### Priorytet wymagań (K1)

Ewaluacja ważności/pilności wymagania:

Przykłady priorytetów wymagań:

- Wysoki
- Średni
- Niski

### Krytyczność wymagań (K1)

Ewaluacja ryzyka wymagania poprzez ewaluację strat powstałych w przypadku niewypełnienia wymagania. Krytyczność jest wyrażana za pomocą poziomów, czym wyższy poziom tym poważniejsze konsekwencje w przypadku awarii funkcjonalności.

### **1.1.8 Weryfikacja i walidacja (K1)**

Walidacja jest procesem potwierdzania, że specyfikacja danej fazy lub całego systemu spełnia wymagania klienta.

Walidacja jest zazwyczaj przeprowadzana przy współpracy z klientem i stawia za cel potwierdzenie, że wymagania lub specyfikacja wymagań opisuje to, czego klient potrzebuje.

Weryfikacja jest to porównanie pośredniego produktu z jego specyfikacjami. Jest więc ustaleniem czy oprogramowanie zostało wyprodukowane poprawnie i zgodnie ze specyfikacjami stworzonymi w poprzednich fazach.

Najczęściej używaną techniką do walidacji i weryfikacji jest testowanie.

Różnicę między walidacją a weryfikacją można wyrazić za pomocą stwierdzenia:

- Walidacja – czy tworzymy właściwy produkt?
- Weryfikacja – czy tworzymy produkt właściwie?

### **1.1.9 Inżynieria Wymagań, Zarządzanie Wymaganiami i Rozwijanie Wymagań (K2)**

#### Rozgraniczenie między Inżynierią Wymagań z Zarządzaniem Wymaganiami i Rozwijaniem Wymagań (K2)

Inżynieria Wymagań jest dyscypliną wchodzącą w skład Inżynierii Oprogramowania skoncentrowaną na identyfikacji i zarządzaniu wymaganiami sprzętowymi lub systemów oprogramowania. Inżynieria oprogramowania obejmuje Zarządzanie Wymaganiami i Wytwarzanie Wymagań.

Dyscyplina Inżynierii Wymagań składa się z następujących podprocesów: pozyskiwanie wymagań, analiza i negocjacje, specyfikowanie, modelowanie systemu oraz walidacja wymagań. Inżynieria Wymagań zawiera podstawowe umiejętności inżynierskie.

Te podprocesy nakładają się, np. modelowanie systemu może być częścią zarówno analizy, jak i specyfikacji, a nawet w niektórych przypadkach pozyskiwania wymagań.

Zarządzanie Wymaganiami ustanawia środowisko robocze dla Inżynierii Wymagań i pośredniczy w wymianie informacji między innymi procesami takimi jak zarządzanie projektami, zarządzanie konfiguracją czy zarządzanie jakością.

Celem Zarządzania Wymaganiami jest zarządzanie wymaganiami produktów oraz komponentów projektu w celu zapewnienia spójności pomiędzy tymi wymaganiami a planami projektu oraz produktami pracy w trakcie cyklu życia produktów w projekcie (cykl życia wytwarzania oraz utrzymywania).

Zarządzanie Wymaganiami obejmuje procesy służące całościowemu zarządzaniu wymaganiami. Jest to proces ciągłego dokumentowania, analizowania, śledzenia, priorytetyzowania, komunikowania oraz uzgadniania wymagań i zarządzania ich zmianami.

Rozwijanie Wymagań jest zbiorem czynności, zadań, technik oraz narzędzi służących do identyfikacji, analizowania oraz walidowania wymagań. Rozwijanie Wymagań obejmuje proces przekształcania potrzeb w wymagania.

Celem Rozwijania Wymagań jest pozyskiwanie, analizowanie, ustanawianie oraz walidowanie wymagań klienta, wymagań produktowych oraz komponentowych.

<b>1.2 Standardy i normy (K1)</b>	<b>20 minut</b>
-----------------------------------	-----------------

Aby zdać egzamin, nie trzeba znać zawartości wszystkich norm. Ważne jest jednak (K1) by wiedzieć, które normy są ważne dla Inżynierii Wymagań.

### 1.2.1 Standardy (K1)

ISO 9000:

Wymagania w stosunku do systemu zarządzania jakością:

- Zdefiniowana podstawowa koncepcja Systemów Zarządzania Jakością
- Niezależny od domeny czy gałęzi przemysłu

ISO 9126 (zastąpiony przez ISO/IEC 25000):

Definiuje model jakości w sześciu kategoriach:

Funkcjonalność, niezawodność, użyteczność, efektywność, utrzymywalność, przenaszalność

IEEE 610:

Standardowy Słownik dla Terminologii Inżynierii Oprogramowania

IEEE 830:

Rekomendowane Praktyki dla Specyfikacji Wymagań Oprogramowania

IEEE 1233:

Wytyczne do Opracowywania Specyfikacji Wymagań Oprogramowania

IEEE 1362:

Wytyczne dla Technologii Informatycznych – Definicja Systemu

SWEBOK – The Guide to the Software Engineering Body of Knowledge (znany jako ISO Technical Report 19759):

SWEBOK opisuje ogólnie akceptowaną wiedzę o inżynierii oprogramowania. Jego 10 obszarów wiedzy podsumowuje podstawowe koncepcje oraz zawiera listę odniesień do szczegółowej informacji.

### 1.2.2 Normy procesowe (K1)

ISO 12207:

Standard dla Procesu Cyklu Życia Oprogramowania

ISO 15288:

Proces Cyklu Życia Oprogramowania

ISO 15504:

Ulepszanie i określanie możliwości procesu oprogramowania (ang. Software Process Improvement and Capability Determination - SPICE)

### 1.2.3 Powody zaniedbywania Inżynierii Wymagań (K2)

Inżynieria Wymagań ma kluczowe znaczenie jednak ciągle jest zaniedbywana.

Powody zaniedbywania Inżynierii Wymagań mogą być następujące (K2):

- Duża presja czasu
- Zorientowanie na szybkie wyniki
- Zorientowanie tylko i wyłącznie na koszt
- Branie pod uwagę tylko wymagań funkcjonalnych
- Niewłaściwa interpretacja (wiele rzeczy jest postrzeganych tak jak je podano) I brak zrozumienia ważności Inżynierii Wymagań dla sukcesu projektu

Możliwe konsekwencje zaniedbywania Inżynierii Wymagań (K2):

- Wymagania stają się nieprecyzyjne
- Wymagania są niejednoznaczne
- Wymagania są sprzeczne

- Wymagania często się zmieniają podczas wytwarzania oprogramowania
- Wymagania nie spełniają kryteriów
- Wymagania mogą być interpretowane na różne sposoby
- Brakujące wymagania

<b>2 Modele procesu i proces Inżynierii Wymagań (K2)</b>	<b>60 minut</b>
--	-----------------

*Cele nauczania dla Poziomu Podstawowego Inżynierii Wymagań*  
Cele określają wiedzę i umiejętności, uzyskiwane po ukończeniu każdego modułu.

### **2.1 Modele procesu (K2)**

- LO-2.1.1      Opisać różne modele procesu (K2)
- LO-2.1.2      Wyjaśnić różnice pomiędzy różnymi modelami procesu (K2)

### **2.2 Proces Inżynierii Wymagań (K2)**

- LO-2.2.1      Opisać charakterystyki procesu Inżynierii Wymagań (K2)
- LO-2.2.2      Objaśnić fazy procesu Inżynierii Wymagań (K2)

<b>2.1 Modele procesu (K2)</b>	<b>30 minut</b>
--------------------------------	-----------------

## Pojęcia

Cykl życia produktu, Model V, Modele procesu, Programowanie Ekstremalne, Rational Unified Process, Scrum

### 2.1.1 Modele procesu (K2)

Modele procesu są niezależnym od metody opisem procesu wytwarzania.

Brane pod uwagę są role, czynności, fazy i dokumenty.

Model procesu oprogramowania dostarcza standardowego formatu dla planowania, organizowania i wykonywania projektu softwarowego.

#### Cykl życia produktu (PLC) (K2)

Definiuje różne fazy wytwarzania produktu.

Podstawowe fazy to:

1. Planowanie
2. Wytwarzanie
3. Utrzymanie
4. Koniec życia

Faza planowania obejmuje: wizję, strategię, plan biznesowy oraz analizę kosztów i korzyści.

Faza wytwarzania obejmuje: specyfikację, szkic projektu oraz implementację. Faza wytwarzania jest często podzielona na następujące cztery fazy:

- Analiza
- Projekt
- Implementacja
- Testowanie

### 2.1.2 Ogólny model V (K2)

Kroki wytwarzania:

- Definicja wymagań, określenie wymagań (specyfikacja wymagań wysokopoziomowych)
- Szkic funkcjonalny system, analiza systemowa (specyfikacja funkcjonalna)
- Szkic techniczny systemu, zarys architektury (projekt oprogramowania)
- Specyfikacja modułów
- Implementacja

Model ma kształt litery V i każdy poziom w procesie wytwarzania ma odpowiadający mu poziom testów:

Definicja i analiza wymagań	→	Testowanie Akceptacyjne
Projekt funkcjonalny system	→	Testowanie Systemowe
Projekt techniczny	→	Testowanie Integracji
Projekt komponentów (modułów)	→	Testowanie Jednostkowe
Implementacja		

*Dla firm szkoleniowych: graficzna reprezentacja ogólnego modelu V; głębszy opis modelu.*

### 2.1.3 Rational Unified Process (RUP©) (K2)

Rational Unified Process jest modelem procesu opracowanym przez IBM Rational ©

Jest to model iteracyjny (tzn. Proces jest powtarzany dopóki wszystkie scenariusze, ryzyka i zmiany nie zostały zaadresowane). Model ten ma 9 dyscyplin włącznie z dyscypliną wymagań (6 dyscyplin inżynierskich oraz 3 dyscypliny pomocnicze). Każda dyscyplina jest pokryta przez 4 fazy cyklu życia projektu: fazę inceptji, elaboracji, konstrukcji i przejścia.

*Dla firm szkoleniowych: uszczegółowienie modelu RUP© wraz z reprezentacją graficzną; szczegółowe studium dyscypliny wymagań.*

## 2.1.4 Podejścia zwinne

### Zarządzanie Wymaganiami

W środowiskach zwinnych wymagania są komunikowane i śledzone poprzez mechanizmy takie jak rejestry produktowe, historyjki i prototypy ekranów. Zobowiązania co do wymagań są wykonywane albo przez zespół, albo przez uprawnionego kierownika zespołu. Przydział zadań jest regularnie (np. codziennie, co tydzień) korygowany w oparciu o postęp prac i w miarę wzrastającego zrozumienia wymagań oraz rozwiązania. Możliwość śledzenia oraz spójność pomiędzy wymaganiami a produktami prac jest zarządzana poprzez powyżej wspomniane mechanizmy oraz poprzez czynności początku i końca iteracji, takie jak „retrospektywy” oraz „dni demo”.

### Rozwijanie Wymagań

W środowiskach zwinnych, potrzeby i pomysły klienta są pozyskiwane, rozwijane, analizowane i walidowane iteracyjnie. Wymagania są dokumentowane w formach takich, jak historyjki użytkownika, scenariusze, przypadki użycia, rejestry produktowe oraz wyniki iteracji (działający kod w przypadku oprogramowania). To, które wymagania będą adresowane w danej iteracji zależy od oceny ryzyka oraz priorytetów przypisanych do tego, co pozostało w rejestrze produktowym. To, jaki poziom szczegółowości wymagań (i innych artefaktów) zastosować, zależy od potrzeby koordynacji (pomiędzy członkami zespołu, zespołami i następnymi iteracjami) oraz ryzyka utraty wiedzy. Kiedy klient jest w zespole, nadal może być konieczne utrzymanie osobnej dokumentacji klienta oraz produktu, aby umożliwić eksplorację wielu rozwiązań. W miarę rozwijania rozwiązania, odpowiedzialność za wydzielone wymagania jest przypisana do odpowiednich zespołów.

## 2.1.5 Programowanie Ekstremalne (K2)

Programowanie Ekstremalne jest metodologią wytwarzania oprogramowania opracowaną przez Kenta Blacka w celu odpowiedzi na zmieniające się wymagania klienta. Zarządzanie projektem (np. Scrum, sekcja 2.1.6), definiuje jak i kiedy zmiany wymagań klienta są zaimplementowane w rozwiązaniu (np. w której iteracji). Metodologia zaleca częste dostarczanie oprogramowania do klienta w krótkich cyklach operacji (ramy czasowe), co ma na celu ulepszenie produktywności i dostarczenie punktów kontrolnych, w których można przyswoić nowe wymagania.

Niektóre z charakterystyk programowania ekstremalnego obejmują:

- Programowanie parami
- Intensywny przegląd kodu
- Testowanie jednostkowe kodu
- Unikanie programowania funkcjonalności do momentu, w którym są one rzeczywiście potrzebne

- Płaska struktura zarządzania (brak złożonej hierarchii wewnątrz zespołu). Najlepiej można to zaobserwować w zespołach Scrum – zespół Scrum jest „samoorganizujący”, nie ma ról takich, jak: kierownik liniowy, kierownik projektu etc.
- Prostota i jasność kodu
- Oczekiwanie zmian w wymaganiach klienta wraz z upływem czasu i lepszym zrozumieniem problemu
- Częsta komunikacja z klientem oraz pomiędzy programistami
- Kompletny brak określania wymagań (nie ma osobnej „fazy” wymagań przed startem wytwarzania; rozwijanie, doskonalenie i odkrywanie wymagań jest częścią właściwego wytwarzania oprogramowania (programowania))

### 2.1.6 Scrum (K2)

Scrum jest zwinnym środowiskiem pracy. Scrum pierwotnie został sformalizowany dla projektów wytwarzania oprogramowania. Scrum obejmuje zbiór praktyk i predefiniowanych ról. Główne role w Scrum to:

- Mistrz Młyna (Scrum Master) – odpowiedzialny za utrzymywanie procesu
- Właściciel Produktu (Product Owner) – reprezentuje interesariuszy i biznes
- Zespół (Team) – wielofunkcyjna grupa realizująca właściwą analizę, projekt, implementację, testowanie etc.

Jedną z głównych charakterystyk Scrum jest podział wytwarzania na „przebiegi” (zwykle o czasie trwania od dwóch do czterech tygodni). Podczas każdego przebiegu, zespół tworzy tak zwany produkt potencjalnie możliwy do wydania.

Scrum umożliwia zarządzanie wymaganiami poprzez „rejstry”. Istnieją dwa typy rejestrów:

- Rejestr Produktu (Product Backlog) – wysokopoziomowa lista utrzymywana przez cały projekt. Lista gromadzi wymagania w formie opisów wszystkich potencjalnych cech, uszeregowanych według wartości biznesowej. Rejestr Produktu jest własnością Właściciela Produktu.
- Rejestr Przebiegu (Sprint Backlog) – lista zadań do wykonania przez zespół w ciągu następnego przebiegu. Funkcjonalności są dzielone na zadania, które – zgodnie z dobrymi praktykami – powinny trwać od czterech do szesnastu godzin pracy. Rejestr Przebiegu jest własnością Zespołu.

Główne charakterystyki podejścia Scrum to:

- Na koniec każdego przebiegu powinno być dostarczone funkcjonalne oprogramowanie – w praktyce, zespoły rozpoczynają pracę nad analizą wymagań i kontynuują ją w trakcie trwania przebiegu. Podczas dekompozycji zadań, wymagania są uszczegóławiane.

- Od członków zespołu oczekuje się, że będą wchodzić w interakcje, a regularne zaangażowanie klienta jest kluczowym pojęciem w metodykach zwinnych. Opinia klienta może być udzielona na sesji prezentacji nowych wersji oprogramowania pod koniec przebiegu, lub poprzez testy akceptacyjne użytkownika.
- Wymagania zmieniają się pod wpływem informacji zwrotnej otrzymywanej od klienta – działające oprogramowanie pomaga klientom wyjaśnić ich wymagania.
- Wymagania nie są w pełni określone przed rozpoczęciem wytwarzania, ale funkcje wybrane dla danego przebiegu, są określone na początku tego przebiegu.
- Priorytety funkcji mogą być zmieniane w miarę postępów projektu.

Głównym wpływem Scrum na Inżynierię Wymagań jest to, że specyfikacje wymagań nie są ukończone i walidowane przed rozpoczęciem wytwarzania.

Właściciel Produktu i zespół uzgadniają, które cechy z Rejestru Produktowego będą uwzględnione w kolejnym przebiegu na podstawie priorytetu biznesowego i wymaganego nakładu pracy. Wymagania użytkownika są formułowane przez Właściciela Produktu jako „historyjki użytkownika”, które zawierają informacje o tym, „kto, co, dlaczego” w danym wymaganiu, nie zawierają natomiast informacji „jak”. Na początku przebiegu wybrane funkcjonalności są dzielone na zadania w Rejestrze Przebiegu, a następnie implementowane.

Zaangażowanie Właścicieli Produktu, na przykład poprzez prezentowanie im zaimplementowanych funkcji oprogramowania, pomaga zespołowi i samym Właścicielom Produktu wyjaśnić wymagania.

*Dla firm szkoleniowych: podać przykłady historyjek użytkownika oraz odpowiadające im elementy Rejestrów Produktowego i Przebiegu.*

*Dla firm szkoleniowych: opisać co najmniej dwa inne modele zwinne, w tym Crystal.*

### 2.1.7 Model Dojrzałości (K2)

Poziomy dojrzałości służą do identyfikacji i doskonalenia dojrzałości procesu (ocena procesu i doskonalenie procesu).

#### ISO/IEC 15504 (SPICE – Software Process Improvement and Capability Determination) (K1)

ISO/IEC 15504 (SPICE – Software Process Improvement and Capability Determination) jest zbiorem standardów technicznych dla procesu wytwarzania oprogramowania oraz powiązanych z nim biznesowych funkcji zarządczych.

ISO/IEC 15504 może być użyte jako środek do doskonalenia procesu i/lub określenia możliwości (na przykład, oceny możliwości procesu dostawcy).

SPICE definiuje procesy podzielone na pięć kategorii procesów:

- Klient – dostawca
- Inżynieria
- Wsparcie
- Zarządzanie
- Organizacja

Dla każdego z powyższych procesów, ISO/IEC 15504 określa poziom możliwości:

5 Proces optymalizowany

4 Proces przewidywany

3 Proces ustalony

2 Proces zarządzany

1 Proces wykonywany

0 Proces niekompletny

*Dla firm szkoleniowych: uszczegółowienie ISO 15504/SPICE; wraz z graficznym opisem typowych wymagań dla Inżynierii Wymagań*

### Capability Maturity Model Integrated (CMMI)

Definiuje pięć poziomów dojrzałości dla wytwarzania, usług oraz zakupów:

1. Wstępny (chaotyczny, ad hoc, indywidualny heroizm) – punkt startowy dla użycia nowego procesu.
2. Zarządzany – proces jest zarządzany zgodnie z ustalonymi metrykami.
3. Zdefiniowany – proces jest zdefiniowany/uzgodniony jako standardowy proces biznesowy i zdekomponowany do poziomów 0, 1 i 2.
4. Zarządzany ilościowo
5. Optymalizowany – zarządzanie procesem obejmuje celową optymalizację/ulepszenie procesu.

## 2.2 Proces Inżynierii Wymagań (K2)

30 minut

### Pojęcia

Inżynieria Wymagań, Perspektywa, Proces zorientowany na klienta

### 2.2.1 Definicja Procesu Inżynierii Wymagań (K2)

Inżynieria Wymagań jest dyscypliną obejmującą procesy potrzebne do identyfikacji, strukturyzacji i zarządzania wymaganiami. Inżynieria Wymagań obejmuje następujące pod-procesy:

- Identyfikacja wymagań
- Analiza wymagań
- Specyfikacja wymagań
- Uzgadnianie wymagań
- Zmiany wymagań
- Walidacja i zapewnienie jakości

### 2.2.2 Czynniki wpływające na Inżynierię Wymagań

Niektóre czynniki mogą mieć negatywny wpływ na Inżynierię Wymagań:

- Po wewnętrznej stronie (wewnątrz organizacji dostawcy oprogramowania):
  - Brak wiedzy o dziedzinie użytkownika
  - Nieefektywne podejście/metodologia do Inżynierii Wymagań
  - Niewystarczające doświadczenie i umiejętności personelu
- Po zewnętrznej stronie (na zewnątrz organizacji dostawcy oprogramowania):
  - Brak komunikacji
  - Niejasne i/lub zmieniające się cele biznesowe powodujące niestabilne wymagania
  - Brak wiedzy o procesie wytwarzania oprogramowania
  - Brak zaangażowania użytkowników i/lub interesariuszy biznesowych

Istnieją różni interesariusze z różnymi punktami widzenia na proces Inżynierii Wymagań. Ogólnie, proces można postrzegać z punktu widzenia klienta i z punktu widzenia dostawcy (sprzedawcy).

Przykład:

Z punktu widzenia klienta, najbardziej istotnymi aspektami w Inżynierii Wymagań mogą być: interfejs użytkownika, aplikacje i usługi. Z punktu widzenia dostawcy, rozważane będą inne aspekty: architektura, dystrybucja obciążenia.

Metody procesu Inżynierii Wymagań zorientowane na klienta:

- Analiza i projektowanie zorientowane na klienta
- Podejście prototypowe
- Użycie demonstracji jako sposobu walidowania przyrostów systemu

<b>3 Zarządzanie Projektem i Ryzykiem (K2)</b>	<b>60 minut</b>
--	-----------------

*Cele nauczania dla Poziomu Podstawowego Inżynierii Wymagań*  
Cele określają wiedzę i umiejętności, uzyskiwane po ukończeniu każdego modułu.

### **3.1 Zarządzanie Projektem (K2)**

- LO-3.1.1 Wyjaśnić, dlaczego Inżynieria Wymagań jest ważna w projektach (K2)
- LO-3.1.2 Podać błędy, które mogą wystąpić w Inżynierii Wymagań (K1)

### **3.2 Zarządzanie Ryzykiem (K3)**

- LO-3.2.1 Rozpoznać ryzyka związane z Inżynierią Wymagań (K1)

<b>3.1 Zarządzanie Projektem (K2)</b>	<b>30 minut</b>
---------------------------------------	-----------------

## Pojęcia

Definicja projektu, Koncepcja projektu, Negocjacje kontraktu, Realizacja projektu

### 3.1.1 Potrzeba Inżynierii Wymagań w projektach (K2)

Niektóre z głównych powodów, dla których projekty odnoszą porażkę są związane z wymaganiami. Zaniedbanie Inżynierii Wymagań może spowodować, że wymagania są nieprecyzyjne lub sprzeczne, albo nie spełniają kryteriów i potrzeb interesariuszy. W związku z tym staranny i uporządkowany proces Inżynierii Wymagań jest niezbędnym elementem każdego projektu.

Inżynieria Wymagań powinna dostarczać wkładu do następujących obszarów (K1):

- Koncepcja projektu
  - Identyfikacja potrzeb i oczekiwań klientów względem rozwiązania danego problem
  - Ustalenie wymagań wysokopoziomowych
- Negocjacje kontraktu
  - Ocena wymagań klientów
  - Określenie wstępnego zakresu oraz zasobów wymaganych dla projektu
  - Określenie kosztu wytwarzania (implementacji wymagań)
  - Uzgodnienie priorytetów wymagań
- Definicja projektu
  - Definicja ról, zadań, czynności oraz dodatkowych procesów (na przykład: Zarządzanie Zmianą)
  - Szczegółowy projekt biznesowy rozwiązania
  - Wkład do projektu architektury
  - Wkład do produktów procesu testowego
- Realizacja projektu
  - Dostarczenie podstaw do implementacji wymagań oraz weryfikacji i walidacji wymagań (testowanie)
  - Wymuszenie przeglądu planów i dostosowanie ich do obecnego zakresu rozwiązania w przypadku zmian wymagań

### 3.1.2 Jakie błędy mogą wystąpić w Inżynierii Wymagań? (K2)

Najbardziej popularne błędy to:

- Niejasne wymagania
- Zmieniające się wymagania (jeśli zmiany wynikają z niejasnych celów projektu i braku znajomości dziedziny biznesu klienta; zmiany w wymaganiach nie są postrzegane jako błąd w podejściach zwinnych oraz iteracyjnych)
- Niestabilna podstawa produktu i projektu dla podwykonawców
- Niejasne odpowiedzialności (zarówno po stronie klienta, jak i dostawcy)
- Rozbieżności pomiędzy oczekiwaniami klienta i zawartością projektu
- Niewystarczające zaangażowanie klienta
- Definicja projektu z kamieniami milowymi, z nierealnymi założeniami
- Nieprecyzyjne oszacowanie kosztów
- Nieprecyzyjne oszacowanie wpływu
- Brak możliwości śledzenia

## 3.2 Zarządzanie Ryzykiem (K2)

30 minut

### Pojęcia

Analiza Trybów i Efektów Awarii, Ryzyko Produktowe, Ryzyko Projektowe, Ryzyko, Zarządzanie Ryzykiem, Plan Zarządzania Ryzykiem

### 3.2.1 Potrzeba Zarządzania Ryzykiem (K2)

Efektywne Zarządzanie Ryzykiem jest kluczowe do zmniejszenia ryzyk projektowych i produktowych. Identyfikacja, odpowiednia analiza oraz planowanie adekwatnych reakcji na ryzyka, minimalizuje szanse wystąpienia ryzyka oraz konsekwencje w przypadku, gdy ryzyko wystąpi.

### 3.2.2 Ryzyko (K2)

#### Ryzyko (K1)

Ryzyko jest zdefiniowane jako efekt niepewności celów, zarówno pozytywny, jak i negatywny [ISO 31000].

Inna definicja opisuje ryzyko jako szansę zdarzenia, zagrożenia lub sytuacji występujących i powodujących niepożądane skutki lub potencjalny problem. Poziom ryzyka jest określany jako prawdopodobieństwo wystąpienia niepomyślnego zdarzenia oraz jego wpływu (szkody wynikłej z tego wydarzenia) [ISTQB].

#### Typy ryzyka (K2)

Istnieją dwa typy ryzyka:

- Ryzyko produktowe
- Ryzyko projektowe

#### Ryzyka projektowe (K2)

Ryzyka projektowe to te ryzyka, które odnoszą się do zdolności projekty do spełnienia jego celów, takie jak:

- Czynniki organizacyjne:
  - Umiejętności, szkolenia oraz braki kadrowe
  - Problemy z personelem

- Kwestie polityczne, takie jak:
  - Problemy z interesariuszami komunikującymi swoje potrzeby i oczekiwania
  - Niepowodzenia w przestrzeganiu informacji uzyskanych podczas przeglądów (na przykład: brak doskonalenia praktyk dokumentowania wymagań)
- Niewłaściwe oczekiwania i podejście do Inżynierii Wymagań
- Kwestie techniczne:
  - Problemy w zdefiniowaniu właściwych wymagań
  - Zakres, w jakim wymagania nie mogą być spełnione przy obecnych ograniczeniach
  - Implementacja lub środowisko testowe nie gotowe na czas
  - Niska jakość projektu, kodu, danych konfiguracyjnych, danych testowych i testów
- Problemy dostawcy:
  - Porażka trzeciej strony (na przykład: komponenty nie dostarczone na czas)
  - Kwestie kontraktowe

### Ryzyka produktowe (K2)

Potencjalne obszary awarii (niepożądanych przyszłych zdarzeń lub zagrożeń) w oprogramowaniu lub systemie, gdyż są to ryzyka w odniesieniu do jakości produktu. Obejmuje to:

- Wyższe ryzyko porażki dostarczonego oprogramowania (oprogramowanie lub system nie realizuje pożądaných funkcji w określonych limitach)
- Niska jakość dokumentacji oprogramowania (niekompletna, niespójna, trudna do utrzymania)
- Ryzyko, że oprogramowanie/sprzęt wyrządzi szkodę jednostce lub organizacji
- Słabe charakterystyki oprogramowania (na przykład: funkcjonalność, wiarygodność, użyteczność lub wydajność)
- Niska integralność i jakość danych (na przykład: kwestie migracji danych, problemy z konwersją danych, problemy z transportem danych, naruszenie standardów danych)
- Oprogramowanie nie wykonuje pożądaných funkcji i nie spełnia potrzeb interesariuszy
- Ryzyko biznesowe wynikające ze złej jakości

### **3.2.3 Zarządzanie Ryzykiem (K2)**

Zarządzanie Ryzykiem jest procesem identyfikacji, oceny i priorytetyzacji, planowania reakcji na ryzyko oraz rozwiązywania i monitorowania ryzyk. Umożliwia identyfikację potencjalnych czynników, które mogą mieć negatywny wpływ na realizację projektu i przygotowanie odpowiednich akcji radzenia sobie z ryzykiem, jeśli wystąpi.

Od różnych interesariuszy mogą pochodzić różne ryzyka: na przykład, zespół programistyczny dostrzeże inne ryzyka, niż interesariusze biznesowi lub użytkownicy końcowi.

Zarządzanie Ryzykiem składa się z następujących czynności [ISTQB]:

- Identyfikacja ryzyka
- Analiza ryzyka
- Łagodzenie ryzyka

#### Potencjalne sposoby obchodzenia się z ryzykiem

Techniki zarządzania ryzykiem dzielą się na cztery główne kategorie:

- Unikanie
- Redukcja
- Dzielenie
- Retencja

#### Plan Zarządzania Ryzykiem

Plan Zarządzania Ryzykiem powinien być stworzony przed i po (okresowo aktualizowany) utworzeniu Planu Projektu. Plan Zarządzania Ryzykiem powinien dostarczać efektywnej kontroli bezpieczeństwa dla zarządzania ryzykami oraz zawierać harmonogram implementacji punktów kontrolnych i osoby odpowiedzialne za te czynności.

Plan Zarządzania Ryzykiem obejmuje:

- Listę ryzyk
- Prawdopodobieństwa wystąpienia i/lub priorytet
- Dotkliwość skutku dla każdego ryzyka (wraz z kosztem, jeśli to możliwe)
- Strategie łagodzenia dla każdego ryzyka (wraz z osobą/grupą odpowiedzialną za podejmowanie czynności, jeśli to możliwe)
- Macierz oceny ryzyka

### **3.2.4 Analiza Trybów i Efektów Awarii (K2)**

Analiza Trybów i Efektów Awarii (Failure Mode and Effects Analysis) jest popularną techniką Zarządzania Ryzykiem (identyfikacji, analizy i planowania reakcji).

FMEA umożliwia uszeregowanie potencjalnych awarii zgodnie z dotkliwością ich konsekwencji, częstością występowania i tym, jak łatwo mogą one być wykryte. FMEA dokumentuje również obecną wiedzę i czynności związane z ryzykiem awarii w celu użycia w ciągłym doskonaleniu. W większości przypadków FMEA jest stosowana podczas fazy projektowania w projekcie a jej głównym celem jest unikanie przyszłych awarii. W późniejszych fazach może być użyta do kontroli procesu. FMEA powinna rozpocząć się w najwcześniejszych fazach conceptualnych w projekcie i trwać przez cały cykl życia. Idealnie, FMEA powinna być zaplanowana tak szybko, jak tylko dostępna jest wstępna informacja.

Wynikiem FMEA są czynności zapobiegające lub zmniejszające dotkliwość lub prawdopodobieństwo awarii.

#### Kroki implementacji FMEA

FMEA jest tworzona w trzech głównych krokach:

- Krok 1: Dotkliwość (identyfikacja dotkliwości potencjalnej awarii)
- Krok 2: Wystąpienie (określenie, jak często może wystąpić potencjalna awaria)
- Krok 3: Wykrycie (identyfikacja technik wykrywania awarii)

<b>4 Odpowiedzialności i role (K2)</b>	<b>50 minut</b>
--	-----------------

*Cele nauczania dla Poziomu Podstawowego Inżynierii Wymagań*  
Cele określają wiedzę i umiejętności, uzyskiwane po ukończeniu każdego modułu.

#### **4.1 Podstawowe Role (K1)**

- LO-4.1.1 Podać podstawowe role występujące w Inżynierii Wymagań (K1)
- LO-4.1.2 Opisać cel oraz rolę interesariusza (K2)

#### **4.2 Zadania Inżynierii Wymagań (K2)**

- LO-4.2.1 Opisać zadania Inżynierii Wymagań (K2)
- LO-4.2.2 Podać charakterystyki Profesjonalisty Inżynierii Wymagań (K1)

<b>4.1 Podstawowe Role (K1)</b>	<b>20 minut</b>
---------------------------------	-----------------

## Pojęcia

Interesariusz, Klient, Kontraktor

### 4.1.1 Podstawowe role (K2)

#### Role, które wpływają lub na które ma wpływ Inżynieria Wymagań

##### Klient

Osoba, grupa lub organizacja zamawiająca rozwiązanie.

##### Kontraktor (dostawca)

Osoba, grupa lub organizacja dostarczająca rozwiązanie.

Klient formułuje swoje potrzeby i dostarcza wstępnych potrzeb oraz oczekiwań biznesowych. Zwykle je to dostarczane wraz z zapytaniem ofertowym. Odpowiedzialnością dostawcy jest przeanalizowanie tych potrzeb i na ich podstawie pozyskanie wymagań.

Kontraktor dostarcza rozwiązania w oparciu o potrzeby klienta.

#### Role w Inżynierii Wymagań

##### Kierownik Wymagań

Kierownik Wymagań jest osobą odpowiedzialną za dokumentację, analizę, śledzenie, priorytetyzację oraz uzgadnianie wymagań a następnie za kontrolowanie zmian i komunikowanie ich odpowiednim interesariuszom.

##### Wytwórca Wymagań

Wytwórca Wymagań jest osobą zorientowaną technicznie, zaangażowaną głównie w pozyskiwanie, analizę i priorytetyzację wymagań.

### 4.1.2 Interesariusz (K2)

Interesariusz jest grupą lub jednostką, na którą ma wpływ, lub która jest w pewien sposób odpowiedzialna za wynik przedsięwzięcia. Interesariusze projektu to osoby i organizacje aktywnie zaangażowane w projekt, lub których interesy mogą być dotknięte przez wynik realizowania lub ukończenia projektu.

Interesariusze mogą być osoby fizyczne, osoby prawne lub osoby abstrakcyjne.

Interesariusze często mają konflikty interesów. To nierzadko powoduje sprzeczne wymagania. Problem sprzecznych wymagań musi być zaadresowany podczas fazy Analizy Wymagań.

Istnieje wiele kategorii interesariuszy:

- Klienci
- Użytkownicy końcowi
- Kierownicy
- Ludzie zaangażowani w procesy organizacyjne
- Inżynierowie odpowiedzialni za wytwarzanie i utrzymanie
- Klienci organizacji, którzy będą używać system
- Ciała zewnętrzne (regulatorzy)
- Eksperti dziedzinowi

Typowymi interesariuszami są:

- Klient
- Użytkownik końcowy
- Kierownik projektu
- Kierownik produktu
- Analityk systemowy
- Analityk biznesowy
- Reprezentant biznesu
- Personel ds. marketing i sprzedaży
- Programista
- Personel zapewnienia jakości
- Ekspert techniczny (architekt, inżynier baz danych)
- Kierownik zmian

- Zespół projektowy
- Zespół zarządczy

Identyfikacja wszystkich udziałowców jest niezbędna do odpowiedniego rozpatrzenia wszystkich punktów widzenia na planowane rozwiązanie.

*Dla firm szkoleniowych: opis typowych interesariuszy (np. Dyrektor Zarządzający, Kierownik Projektu, klient etc.)*

## 4.2 Zadania Inżynierii Wymagań (K2)

30 minut

### 4.2.1 Zadania Inżynierii Wymagań (K2)

Głównym zadaniem Inżynierii Wymagań jest:

- Analiza procesów biznesowych wykonywanych w organizacji
- Identyfikacja i analiza wymagań
- Strukturyzacja i modelowanie wymagań
- Zapewnienie jakości wymagań i specyfikacji
- Stworzenie specyfikacji wymagań
- Analiza ryzyka (w kontekście wymagań)
- Zarządzanie zmianami wymagań
- Uzgadnianie wymagań z interesariuszami

### 4.2.2 Wiedza Profesjonalisty Inżynierii Wymagań (K1)

Profesjonalista Inżynierii Wymagań powinien posiadać następujące umiejętności miękkie:

- Umiejętność moderacji
- Pewność siebie
- Umiejętność przekonywania
- Umiejętności językowe
- Umiejętności komunikacji
- Precyzja
- Analityczne, jasne myślenie
- Umiejętność działania w sposób uporządkowany
- Kompetencje metodologiczne
- Odporność na stres

<b>5 Identyfikacja Wymagań (K2)</b>	<b>150 minut</b>
-------------------------------------	------------------

### *Cele nauczania dla Poziomu Podstawowego Inżynierii Wymagań*

Cele określają wiedzę i umiejętności, uzyskiwane po ukończeniu każdego modułu.

#### **5.1 Klient (K1)**

LO-5.1.1 Podać zawartość kontraktu (K1)

LO-5.1.2 Określić, co powinno być rozważane podczas oceny wymagań (K2)

#### **5.2 Wizja i Cele Projektu (K2)**

LO-5.2.1 Wyjaśnić charakterystyki typowej wizji projektu (K2)

#### **5.3 Identyfikacja Interesariuszy (K2)**

LO-5.3.1 Wyjaśnić, jak można identyfikować interesariuszy (K2)

LO-5.3.2 Zidentyfikować interesariuszy w określonym projekcie (K3)

#### **5.4 Techniki Identyfikacji Wymagań (K2)**

LO-5.4.1 Określić cele identyfikacji wymagań (K2)

LO-5.4.2 Zastosować różne techniki identyfikacji wymagań (K3)

#### **5.5 Wymagania Funkcjonalne i Niefunkcjonalne (K2)**

LO-5.5.1 Opisać charakterystyki wymagań funkcjonalnych oraz niefunkcjonalnych (K2)

LO-5.5.2 Porównać różnice pomiędzy wymaganiami funkcjonalnymi oraz niefunkcjonalnymi (K2)

#### **5.6 Opis Wymagań (K2)**

LO-5.6.1 Opisać zawartość standardowego dokumentu wymagań (K2)

LO-5.6.2 Opisać charakterystyki dobrego wymagania (K2)

LO-5.6.3 Skonstruować wymaganie (K3)

<b>5.1 Klient (K1)</b>	<b>20 minut</b>
------------------------	-----------------

## Pojęcia

Klient, Kontrakt

### 5.1.1 Klient (K1)

Klient jest organizacją lub osobą zamawiającą oprogramowanie i jednym z kluczowych interesariuszy projektu. Potrzeby klienta muszą być zaspokojone.

Klient musi być zawsze zaangażowany. Celem tego jest zrozumienie klienta i wypracowanie wspólnego porozumienia pomiędzy klientem a dostawcą. Dostawca powinien zatem zawsze stawać się w pozycji klienta.

Podczas oceny wymagań dla celów planowania projektu (co jest jednym z tematów, które muszą być zrealizowane), należy wziąć pod uwagę różne punkty widzenia, gdyż poszczególne wymagania mogą mieć inny priorytet i krytyczność dla różnych interesariuszy.

### 5.1.2 Kontrakt (K2)

Umowa (kontrakt) powinna formalnie określać oraz opisywać to, czego żąda klient. Należy zapewnić, iż interes klienta jest najważniejszy (na przykład: dostawca nie wymusza preferowanego przez siebie rozwiązania, ale analizuje potrzeby klienta i rekomenduje rozwiązanie, które umożliwi spełnienie tych potrzeb w możliwie najlepszy sposób).

Umowa:

- Musi być zgodna z zasobami dostępnymi do implementacji rozwiązania
- Jest oparta na: oszacowaniach, terminach, stawkach i planach projektu

Inżynieria Wymagań dostarcza informacji wejściowej dla tego typu oszacowań.

Kontrakt powinien zawierać:

- Krótki opis planowanego rozwiązania
- Listę spriorytetyzowanych wymagań wysokopoziomowych
- Kryteria akceptacji dla każdego wymagania
- Listę produktów (dokumentacja, kod, działające oprogramowanie)

- Terminy dla wytwarzania oraz dostarczenia produktu
- Inne potrzeby i oczekiwania, takie jak preferowana technologia, wymagania odnośnie zasobów etc.

<b>5.2 Wizja i Cele Projektu (K2)</b>	<b>20 minut</b>
---------------------------------------	-----------------

## Pojęcia

Cel, Wizja

### 5.2.1 Wizja (K2)

Opracowanie wizji projektu jest pierwszym krokiem Inżynierii Wymagań.

Wizja powinna:

- Definiować klientów, rynki oraz konkurencję
- Definiować cele do osiągnięcia
- Umożliwić osiągnięcie wspólnego zrozumienia pomiędzy interesariuszami

Bardzo ważne jest posiadanie jasnej definicji wizji projektu.

*Dla firm szkoleniowych: zaprezentować typowe wizje projektu.*

#### Ważne pytania dotyczące wizji projektu (K2):

- Co projekt zmieni?
- Do czego projekt jest potrzebny?
- Co się zdarzy, kiedy projekt zostanie przerwany?
- Kto zyska na projekcie?
- Jakie koszty jesteśmy w stanie ponieść?
- Jakie ryzyko jesteśmy gotowi podjąć?

Dla każdego projektu, wizja musi zostać zdefiniowana na nowo.

Czynniki wpływające na wizję projektu (K1)

Następujące czynniki mogą mieć wpływ na wizję projektu:

- Klienci
  - Cele klienta
  - Preferencje klienta
- Strategia
  - Strategia organizacji
  - Pozycjonowanie na rynku
  - Kierunki rozwoju
- Konkurencja
  - Dane konkurentów
  - Rozwój rynku
- Produkty
  - Poziom innowacji
  - Grupa docelowa
- Technologie
  - Nowe narzędzia
  - Nowe standard
- Dostępne zasoby
  - Czas, współpracownicy, możliwości

<b>5.3 Identyfikacja Interesariuszy (K2)</b>	<b>20 minut</b>
--	-----------------

## Pojęcia

Interesariusz

### 5.3.1 Identyfikacja Interesariuszy (K2)

Należy zidentyfikować wszystkich interesariuszy po stronie klienta i dostawcy.

Każdy interesariusz, lub każda grupa interesariuszy, może dostarczyć nowych wymagań i wpływać na projekt planowanego rozwiązania. Jeśli nie zidentyfikowano wszystkich interesariuszy, istnieje ryzyko, że pewne ważne wymagania lub ograniczenia pozostaną nieznanymi i nie będą uwzględnione w projekcie. Pominięci interesariusze mogą powodować konieczność złożonych zmian w oprogramowaniu w późnej fazie projektu lub po wdrożeniu systemu na środowisko produkcyjne.

Niektórzy z interesariuszy mogą tworzyć grupy interesów (na przykład: wszyscy interesariusze biznesowi). Grupy interesów powinny być zebrane razem w celu bardziej efektywnego zarządzania ich wymaganiami.

### 5.3.2 Procedura identyfikacji i oceny interesariuszy (K2)

Procedura identyfikacji i oceny interesariuszy obejmuje następujące czynności:

- Identyfikacja interesariuszy (analiza procesów biznesowych, określenie właścicieli procesów oraz produktów, analiza struktury organizacyjnej i rynku)
- Grupowanie interesariuszy (jeśli możliwe)
- Określenie relacji
- Identyfikacja potencjalnych konfliktów
- Analiza konfliktów, ich źródeł i określenie możliwości strategii wygrana-wygrana
- Identyfikacja interesariuszy minimalizujących ryzyko w celu większego zaangażowania ich w czynności projektowe
- Identyfikacja perspektyw interesariuszy

*Dla firm szkoleniowych: wyjaśnienie procesu identyfikacji i oceny interesariuszy.*

## 5.4 Techniki Identyfikacji Wymagań (K2)

40 minut

### Pojęcia

Burza mózgów, Kwestionariusz, Obserwacja polowa, Ponowne użycie, Reprezentant klienta, Samo-rejestracja, Terminowanie, Warsztaty, Wywiad

### 5.4.1 Cel identyfikacji wymagań (K2)

Głównymi celami Identyfikacji Wymagań są:

- Identyfikacja wszystkich pożądaných funkcji, charakterystyk, ograniczeń oraz oczekiwań
- Zorientowanie wymagań względem wizji projektu
- Uszczegółowienie wymagań wysokopoziomowych i jasne opisanie funkcji oraz usług
- Wyłączenie funkcji i cech, których klient nie potrzebuje

### 5.4.2 Techniki (K1)

Najpopularniejszymi technikami Identyfikacji Wymagań są:

- Kwestionariusze
- Wywiady
- Samo-rejestracja (nagrywanie)
- Reprezentant klienta w zespole
- Identyfikacja na podstawie istniejących dokumentów
- Ponowne użycie (ponowne użycie specyfikacji z określonego projektu)
- Burza mózgów
- Obserwacja polowa
- Terminowanie
- Warsztaty

#### 5.4.2.1 Kwestionariusze

Kwestionariusz może zawierać pytania otwarte lub zamknięte. Pytanie otwarte wymaga od respondenta sformułowania jego własnej odpowiedzi. W przypadku pytań zamkniętych, respondent jest proszony o wybór odpowiedzi z pewnej liczby możliwych opcji. Opcje te powinny się wykluczać.

Korzyści:

- Niskie koszty
- Możliwość dotarcia do większej grupy odbiorców

Wady:

- Nieodpowiednie do pozyskiwania niejawnnej informacji
- Niski wskaźnik zwrotu przy niskiej motywacji respondentów
- Kwestionariusze mogą być ukierunkowane, co utrudnia identyfikację prawdziwych potrzeb użytkowników

#### 5.4.2.2 Wywiad

Wywiad jest techniką konwersacyjną, w której przeprowadzający wywiad zadaje pytania respondentowi w celu uzyskania informacji na określony temat. Technika ta jest bardzo interaktywna i umożliwia modyfikację kolejności uprzednio przygotowanych pytań w zależności od odpowiedzi respondent oraz sytuacji.

Przeprowadzenie dobrego wywiadu jest trudniejsze, niż mogłoby się to wydawać. Powodem tego jest typowe zachowanie podczas rozmowy (na przykład: kończenie zdań za partnera), co może prowadzić do wprowadzenia nadinterpretacji do danych. Przeprowadzający wywiad powinien zadawać pytania otwarte w celu uzyskania informacji a pytania zamknięte w celu potwierdzenia statusu (na przykład: potwierdzenia już zidentyfikowanych wymagań).

Korzyści:

- Możliwość dostosowania przebiegu wywiadu do określonego respondenta

Wady:

- Czasochłonność
- Niewystarczająca reprodukcja wyników (trudność uzyskania tych samych odpowiedzi podczas powtarzania wywiadu)

#### 5.4.2.3 Samo-rejestracja

W tej technice interesariusz (na przykład: użytkownik końcowy) dokumentuje swoje czynności wykonywane w celu ukończenia określonego zadania.

Oprócz dokumentowania czynności w obecnej postaci („AS-IS”), użytkownik opisuje również zmiany, żądania i potrzeby.

Technikami powiązаныmi z tymi podejściem są: demonstracje oraz przeglądy dokumentacji.

Korzyści:

- Niski nakład czasu i pracy dla Inżyniera Wymagań po stronie dostawcy

Wady:

- Zaniedbywanie czynności “automatycznych” (jak drukowanie i odbiór wydrukowanej kopii)
- Wysoka zależność od motywacji i doświadczenia użytkownika

#### 5.4.2.4 Reprezentant klienta w zespole

Podejście to jest jednym z najbardziej efektywnych metod Identyfikacji (i walidacji) Wymagań, ponieważ umożliwia reprezentantowi systematyczne monitorowanie postępu, weryfikację poprawności projektu oraz dostarczanie informacji zwrotnej tam, gdzie to potrzebne.

Obecność reprezentanta klienta w zespole jest jedną z podstawowych zasad metod zwinnych.

Korzyści:

- Szybka informacja zwrotna
- Dostarczanie zorientowanych na użytkownika wymagań, które są łatwo akceptowane

Wady:

- Wysoki koszt dla klienta
- Koszty adaptacji

#### 5.4.2.5 Identyfikacja wymagań na podstawie istniejących dokumentów

Ta technika może być użyte w przypadku, gdy w organizacji istnieje dokumentacja, która może pomóc w identyfikacji wymagań. Dokumentacją taką mogą być:

- Modele i mapy procesów
- Opisy procesów
- Diagramy organizacyjne
- Specyfikacje produktów (w znaczeniu rezultatu określonego procesu)
- Procedury (na przykład: procedury pracy)
- Standardy oraz instrukcje

Zidentyfikowane wymagania są podstawą do dalszej Analizy Wymagań i muszą być uszczegółowione oraz rozszerzone poprzez inne, powiązane, wymagania.

Korzyści:

- Nie jest pomijana żadna funkcjonalność

Wady:

- Wysokie koszty
- Nie można zastosować, jeśli nie istnieje żadna, lub istnieje tylko podstawowa, dokumentacja w organizacji
- Nie można zastosować, jeśli dokumentacja nie jest prawidłowo utrzymywana (nieaktualna)

#### 5.4.2.6 Ponowne użycie (ponowne użycie specyfikacji z określonego projektu)

Ponowne użycie specyfikacji z określonego projektu może być wykonane, jeśli organizacja ukończyła już jeden lub więcej projektów podobnych do aktualnego projektu. Specyfikacja wymagań przygotowana dla poprzedniego projektu(ów) może być użyta w innym projekcie w celu skrócenia czasu trwania analizy i dokumentacji wymagań – i tym samym umożliwia wcześniejsze rozpoczęcie implementacji.

W większości przypadków jedynie niektóre części istniejącej specyfikacji mogą być użyte w nowym projekcie. Dokumentacja, która będzie ponownie używana powinna być zawsze sprawdzona pod kątem zgodności z aktualnymi potrzebami oraz wymaganiami i odpowiednio dostosowana.

Korzyści:

- Oszczędność kosztów

Wady:

- Wysokie koszty pierwszego projektu
- Reużycie wymagań może wymagać intensywnego i kosztownego zarządzania zmianami jeśli nie było ono poprawnie prowadzone w poprzednich projektach

#### 5.4.2.7 Burza mózgów

Burza mózgów jest powszechnie stosowaną techniką pozyskiwania wymagań związanych z mało znanymi lub nowymi obszarami działalności organizacji lub funkcjonalności planowanego systemu. Umożliwia zebranie wielu pomysłów od różnych interesariuszy w krótkim czasie i z niskim kosztem. Podczas sesji burzy mózgów, uczestnicy zgłaszają pomysły i koncepcje dotyczące danego problemu.

Korzyści:

- Niskie koszty
- Szansa zebrania wielu wartościowych pomysłów w krótkim czasie

Wady:

- Trudne z niezmotywowanymi uczestnikami

- Trudne do zastosowania w zespołach rozproszonych

#### 5.4.2.8 Obserwacja polowa

Obserwacja polowa umożliwia obserwację czynności i procesów wykonywanych przez użytkowników i identyfikację wymagań systemowych na tej podstawie. Wykonywanie obserwacji polega na obserwowaniu użytkowników przy pracy i dokumentowaniu procesu, zadań oraz wyników. W niektórych przypadkach obserwacja jest rozszerzona o wywiad z użytkownikami na temat ich pracy oraz sposobów, w jaki realizują swoje zadania.

Korzyści:

- Możliwość obserwacji użytkowników przy pracy i identyfikacji rzeczywistych wymagań
- Użyteczne w przypadku, gdy interesariusze mają trudności z wyrażeniem swoich potrzeb

Wady:

- Ryzyko pominięcia przypadków wyjątkowych
- Niemożliwe do zastosowania w niektórych sytuacjach (na przykład: z powodów bezpieczeństwa lub prawnych)

#### 5.4.2.9 Terminowanie

Celem terminowania jest pozyskanie wymagań od klienta, szczególnie w przypadku, gdy procesy i czynności wykonywane przez personel klienta nie są łatwe do opisanego przy użyciu innych technik, takich jak wywiady, lub klient ma trudności z wyartykułowaniem wymagań, co do planowanego oprogramowania.

Terminowanie jest procesem uczenia się od klienta jego pracy. Klient, który najlepiej wie, jak wykonać określoną pracę, uczy Inżyniera Wymagań – jak mistrz i uczeń.

Korzyści:

- Pomaga ominąć trudności, jakie mogą mieć pracownicy klienta w myśleniu abstrakcyjnym oraz wyrażaniu swoich zadań w słowach

Wady:

- Kosztowne i czasochłonne
- Niemożliwe do zastosowania w niebezpiecznych środowiskach

#### 5.4.2.10 Warsztaty

Warsztat jest rodzajem spotkania koncentrującym się na określonym (uprzednio zdefiniowanym i zakomunikowanym uczestnikom) temacie i w krótkim, intensywnym okresie czasu angażującym interesariuszy reprezentujących zwykle różne obszary i/lub dziedziny.

Warsztaty mogą mieć różne cele:

- Identyfikacja wymagań (na przykład: w celu ustalenia zakresu rozwiązania)
- Odkrycie ukrytych wymagań (na przykład: wymagań, które nie są bezpośrednio wyrażone przez interesariuszy, lub których obecności interesariusze nie są świadomi, a które są konieczne do spełnienia pewnych ich potrzeb lub wymagań wysokopoziomowych)
- Rozwinięcie (uszczegółowienie) wymagań w nowo odkrytym obszarze
- Ustalenie priorytetów wymagań
- Osiągnięcie porozumienia co do wymagań, w momencie zatwierdzenia wymagań
- Przegląd wyników określonego procesu lub czynności (na przykład: przegląd Specyfikacji Wymagań Funkcjonalnych) i rozwiązanie problemów, które mogły się pojawić

Korzyści:

- Zaangażowanie ludzi, którzy mają różne punkty widzenia na dany problem
- Możliwość określenia i opisanie wymagań pochodzących z różnych perspektyw
- Możliwość szybkiego wykrycia oraz rozwiązania potencjalnych konfliktów pomiędzy wymaganiami interesariuszy

Wady:

- Trudne do zastosowania w przypadku zespołów rozproszonych geograficznie
- Dostępność wszystkich osób wymaganych do udziału w warsztacie
- Podczas warsztatu porozumienie nie zawsze jest łatwo osiągnięte, dyskusja może utknąć na (pomniejszych) problematycznych kwestiach, tym samym przedłużając proces i demotywując uczestników

<b>5.5 Wymagania Funkcjonalne i Niefunkcjonalne (K2)</b>	<b>20 minut</b>
--	-----------------

## Pojęcia

Wymagania funkcjonalne, Wymagania niefunkcjonalne

Aby zdać egzamin, kandydaci powinni potrafić podać przykłady wymagań funkcjonalnych oraz niefunkcjonalnych oraz uszczegółwić poszczególne charakterystyki jakościowe.

### 5.5.1 Wymagania funkcjonalne (K2)

Wymagania funkcjonalne określają, co robi system. Specyfikują one funkcje system postrzegane przez użytkownika końcowego. Wymagania funkcjonalne opisują również wyzwalacze procesu (czynności użytkownika, wejścia/wyjścia danych powodujące start procesu biznesowego).

Wymagania funkcjonalne powinny charakteryzować się następującymi charakterystykami jakościowymi [ISO/IEC 25000] (K1):

- Przydatność
- Odpowiedniość
- Możliwość współpracy
- Funkcjonalność
- Zgodność
- Bezpieczeństwo

### 5.5.2 Wymagania niefunkcjonalne (K2)

Wymagania niefunkcjonalne określają, jak system działa; opisują one atrybuty jakościowe całego systemu lub jego określonego komponentu lub funkcji. Ograniczają rozwiązanie na przykład poprzez wymagania określonych parametrów wydajnościowych.

Wymagania niefunkcjonalne są trudne do opisanie i z tego powodu często wyrażane są niejasno lub nie są w ogóle udokumentowane. To powoduje, iż są one trudne do testowania. Dlatego też należy zwracać szczególną uwagę na wymagania niefunkcjonalne na wszystkich etapach procesu Inżynierii Wymagań.

Z powodu problemów z wyrażaniem wymagań niefunkcjonalnych, mogą być one trudne do testowania. Dlatego też wymagania niefunkcjonalne powinny być wyrażane jasno i być mierzalne.

Wymaganiami niefunkcjonalnymi są [ISO/IEC 25000] (K1):

- Wiarygodność
- Użyteczność
- Wydajność
- Utrzymywalność
- Przenaszalność

Wymagania niefunkcjonalne określają kryteria, które mogą być użyte do oceny działania systemu. Z tego powodu mają one znaczący wpływ na satysfakcję klienta z używania oprogramowania. Wymagania funkcjonalne mają dostarczać funkcji; wymagania niefunkcjonalne określają, jak łatwo i efektywnie te funkcje mogą być używane.

<b>5.6 Opis Wymagań (K2)</b>	<b>30 minut</b>
------------------------------	-----------------

### 5.6.1 Opis Wymagań (K2)

Wymagania muszą być określone jasno i precyzyjnie. Powinny być one mierzalne w celu zapewnienia, że są testowalne a ich implementacja może być właściwie kontrolowana. Ważne jest, aby pamiętać, że język potoczny ma pewne ograniczenia i wady. Może to spowodować, iż opis wymagań jest niejasny i niejednoznaczny. Dlatego wszędzie tam, gdzie to możliwe powinny być zastosowane odpowiednie standardy i szablony. Standardy zapewniają wspólne zrozumienie i najlepsze praktyki specyfikacji, podczas gdy szablony ograniczają język, który może być użyty.

Oprócz standardów i szablonów, ważnym narzędziem dla ułatwienia komunikacji pomiędzy różnymi interesariuszami oraz wprowadzenia pewnej kontroli nad niejednoznacznością naturalnego są słowniki.

Opis wymagań musi spełniać różne kryteria (np. jasny, dokładny, jednoznaczny i wymierny, bez zbędnych słów itp.)

### 5.6.2 Procedura Konstrukcji Wymagania (K3)

Konstrukcja wymagania przebiega w następujących krokach:

1. Określenie procesu
  - Koncentracja na funkcjonalności
  - Określenie wejść oraz wyjść
2. Klasyfikacja czynności systemu
  - Identyfikacja niezależnej czynności system
  - Identyfikacja interakcji z użytkownikiem
  - Identyfikacja wymagań interfejsowych (integracyjnych)
3. Określenie stopnia zobowiązania
  - Wyjaśnienie zobowiązania za pomocą słów kluczowych (powinien, musi itp.)
4. Udoskonalenie opisu procesu
  - Uszczegółowiony opis obiektów i punktów integracji
5. Ograniczenia logiczne i czasowe

- Ustalenie warunków brzegowych

Przykład: wymaganie związane z generowaniem faktury z danymi pobranymi z system zewnętrznego.

Krok procedury	Wynik – opis wymagania
1. Określenie procesu: <ul style="list-style-type: none"> <li>• Proces systemowy generowania faktury</li> </ul>	System generuje fakturę
2. Klasyfikacja czynności systemu <ul style="list-style-type: none"> <li>• Generowanie faktury po komendzie użytkownika</li> <li>• Ustalony interfejs z systemem zewnętrznym</li> </ul>	Po wywołaniu żądania faktury przez użytkownika, system generuje fakturę przy użyciu danych pobranych z systemu zewnętrznego
3. Określenie stopnia zobowiązania <ul style="list-style-type: none"> <li>• System musi generować fakturę z poprawnymi danymi pobranymi z systemu zewnętrznego</li> </ul>	Po wywołaniu żądania faktury przez użytkownika, system ma generować fakturę przy użyciu danych pobranych z systemu zewnętrznego
4. Udoskonalenie opisu procesu <ul style="list-style-type: none"> <li>• Ustalenie nazwy systemu zewnętrznego</li> </ul>	Po wywołaniu żądania faktury przez użytkownika, system ma generować fakturę przy użyciu danych pobranych z systemu SAP
5. Ograniczenia logiczne i czasowe <ul style="list-style-type: none"> <li>• Określenie ograniczenia czasowego – maksymalny czas trwania operacji: 30 sekund</li> </ul>	Po wywołaniu żądania faktury przez użytkownika, system w ciągu 30 sekund ma generować fakturę przy użyciu danych pobranych z systemu SAP

*Dla firm szkoleniowych: opis poszczególnych kroków konstrukcji wymagania*

### 5.6.3 Dokument wymagań (K2)

Standardowa zawartość dokumentu wymagań [IEEE 830]:

Spis treści

#### 1. Wprowadzenie

- 1.1 Cel
- 1.2 Zakres
- 1.3 Definicje, akronimy i skróty
- 1.4 Referencje
- 1.5 Przegląd (skrót treści)

#### 2. Opis ogólny

- 2.1 Perspektywa produktowa
- 2.2 Funkcje produktu
- 2.3 Charakterystyki użytkownika
- 2.4 Ograniczenia
- 2.5 Założenia i zależności

#### 3. Wymagania

- 3.1 Interfejsy zewnętrzne
- 3.2 Funkcje
- 3.3 Wymagania wydajnościowe
- 3.4 Wymagania logiczne bazy danych
- 3.5 Ograniczenia projektowe
  - 3.5.1 Zgodność ze standardami
- 3.6 Atrybuty oprogramowania
  - 3.6.1 Wiarygodność
  - 3.6.2 Dostępność
  - 3.6.3 Bezpieczeństwo
  - 3.6.4 Utrzymywalność
  - 3.6.5 Przenaszalność

Aneksy

Indeks

<b>6 Specyfikacja Wymagań (K2)</b>	<b>100 minut</b>
------------------------------------	------------------

*Cele nauczania dla Poziomu Podstawowego Inżynierii Wymagań*

Cele określają wiedzę i umiejętności, uzyskiwane po ukończeniu każdego modułu.

**6.1 Specyfikacja (K2)**

- LO-6.1.1 Opisać cel oraz zawartość specyfikacji wymagań (K2)
- LO-6.1.2 Opisać charakterystyki specyfikacji wymagań (K2)
- LO-6.1.3 Opisać cel oraz zawartość specyfikacji rozwiązania (K2)
- LO-6.1.4 Opisać charakterystyki specyfikacji rozwiązania (K2)
- LO-6.1.5 Podać ważne standardy dla specyfikacji wymagań oraz specyfikacji rozwiązania (K1)
- LO-6.1.6 Wyjaśnić, czym jest historyjka użytkownika (K2)

**6.2 Procedura (K3)**

- LO-6.2.1 Zastosować typową procedurę specyfikacji wymagań (K3)

**6.3 Formalizacja (K2)**

- LO-6.3.1 Wyjaśnić, jakie stopnie formalizacji istnieją dla specyfikacji wymagań (K2)
- LO-6.3.2 Zastosować określony poziom formalizacji w danym scenariuszu (K3)

**6.4 Jakość Wymagań (K2)**

- LO-6.4.1 Podać konsekwencje błędów w wymaganiach (K1)
- LO-6.4.2 Opisać możliwości unikania błędów w wymaganiach (K2)

## 6.1 Specyfikacja (K2)

30 minut

### Pojęcia

IEEE 830, Specyfikacja rozwiązania , Specyfikacja wymagań

#### 6.1.1 Specyfikacja (K1)

Specyfikacja jest wyraźnie określonym zbiorem wymagań, które muszą być spełnione przez materiał, produkt lub usługę.

Specyfikacja służy do śledzenia i zarządzania wymaganiami. W specyfikacji, wymagania podane są w sposób usystematyzowany i są oddzielnie modelowane (wymagania są modelowane „niezależnie”, to znaczy wymagania wysokopoziomowe dzielone są na poziom, na którym każde poszczególne wymaganie stanowi „niezależny” obiekt, który może być dalej rozwijane i śledzone). Specyfikacja jest formalnym porozumieniem odnośnie wymagań do implementacji w planowanym systemie oprogramowania (lub w innej formie rozwiązania).

Termin "specyfikacja" może być używany w kontekście wymagań i rozwiązania.

#### 6.1.2 Specyfikacja Wymagań (K2)

Specyfikacja wymagań opisuje obszar problemu (obszar zainteresowania na przykład: rozwiązanie zagadnienia biznesowego, nowa funkcjonalność etc.) i zawiera co najmniej następujące informacje:

- Wymagania klienta
- Ograniczenia
- Kryteria akceptacji

Stworzenie specyfikacji wymagań klienta powinno być zadaniem klienta, jednakże w niektórych przypadkach, dostawca może wspierać przygotowanie takiej specyfikacji.

Stworzenie specyfikacji innego typu: wymagań systemowych, wymagań na oprogramowanie, wymagań bezpieczeństwa i zabezpieczeń, wymagań środowiskowych, prawnych etc. jest zadaniem dla innych ról.

### 6.1.3 Historyjki Użytkownika (K2)

Historyjki użytkownika są stosowane w zwinnych metodologiach wytwarzania oprogramowania. Historyjki użytkownika są szybką metodą operowania na wymaganiach klienta/użytkownika. Celem historyjki użytkownika jest umożliwienie szybszej odpowiedzi – i z mniejszym narzutem – na gwałtownie zmieniające się wymagania świata rzeczywistego.

Historyjki użytkownika opisują funkcjonalność, która będzie wartościowa. Historyjki są zbudowane z trzech aspektów:

- Pisany opis historyjki używany dla celów planowania oraz jako przypomnienie
- Konwersacja o historyjce służąca pozyskaniu szczegółów
- Testy, które przenoszą i dokumentują szczegóły i które mogą być użyte do stwierdzenia, kiedy historyjka jest kompletna [Mike Cohn: User Stories applied. 2009]

### 6.1.4 Specyfikacja Rozwiązania (K2)

Specyfikacje rozwiązania są nazywane również specyfikacjami funkcjonalnymi, specyfikacjami wymagań systemowych lub specyfikacjami wymagań na oprogramowanie. Opisują one obszar rozwiązania.

Specyfikacja funkcjonalna jest dokumentem, który opisuje jasno wymagania techniczne rozwiązania. Specyfikacja funkcjonalna jest podstawą dalszego wytwarzania systemu, dlatego też musi dostarczać precyzyjnej informacji o wszystkich aspektach funkcjonalnych oprogramowania, które ma być implementowane. W oparciu o te dane, architekci i programiści są stanie efektywnie zaprojektować techniczne aspekty systemu. Specyfikacja funkcjonalna dostarcza testerom wskazówek co do weryfikacji każdego wymagania funkcjonalnego (specyfikacja jest jedną z podstaw testów).

Specyfikacja funkcjonalna nie opisuje, jak będą zaimplementowane funkcje systemu i jaka technologia będzie użyta. Koncentruje się na funkcjonalności, interakcjach pomiędzy użytkownikiem a oprogramowaniem.

Celem specyfikacji funkcjonalnej może być:

- Dostarczenie podstaw do wzajemnego porozumienia co do zakresu i funkcjonalności rozwiązania, które będzie implementowane
- Zapewnienie porozumienia w zespole co do tego, co system ma osiągnąć przez rozpoczęciem pisania kodu źródłowego, podręczników, przygotowaniem danych oraz testowaniem
- Dostarczenie zespołowi programistycznego szczegółowego opisu wymaganej funkcjonalności w znaczeniu interakcji pomiędzy użytkownikiem a oprogramowaniem
- Dostarczenie podstaw do wyrocni testowych dla zespołu testowego

### **6.1.5 Ważne standardy (K1)**

Przy tworzeniu specyfikacji można posłużyć się następującymi standardami:

- IEEE 1362 (System Performance Specifications) – Specyfikacja Operacji Systemu
- IEEE 830 (Software Requirements Specification) – Specyfikacja Wymagań Oprogramowania
- IEEE 1233 (System Functional Specifications) – Specyfikacja Funkcjonalna Systemu

<b>6.2 Procedura (K3)</b>	<b>30 minut</b>
---------------------------	-----------------

### 6.2.1 Procedura Specyfikacji Rozwiązania (K3)

Specyfikacja jest czynnością służącą formalizacji wyników Analizy Wymagań (K2).

Czynności identyfikacji, analizy i specyfikacji prowadzą do akceptacji wymagań (patrz: Akceptacja Wymagań (K2)).

Po identyfikacji, analizie oraz modelowaniu, wymagania powinny być udokumentowane w jasny, jednoznaczny sposób.

Procedura specyfikacji rozwiązania obejmuje następujące czynności:

1. Identyfikacja interesariuszy
2. Definicja wizji oraz celu
3. Określenie wymagań
4. Ustrukturyzowana specyfikacja wymagań
5. Opis środowiska system
6. Określenie rozwiązania (definicja systemu oraz zakresu, wraz z odpowiednimi aspektami poza zakresem, mającymi wpływ na rozwiązanie/zakres, jak interfejsy do systemów zewnętrznych)
7. Analiza wymagań
8. Modelowanie problemu
9. Modelowanie rozwiązania

Procedura formalizuje wyniki procesu Analizy Wymagań. Model rozwiązania jest podstawą do projektu oraz implementacji.

Procedura angażuje wielu interesariuszy, którzy wspierają pracę nad specyfikacją w różnych dziedzinach. Wynik procedury, Specyfikacja Rozwiązania, służy jako punkt startowy do projektowania oprogramowania, sprzętu i baz danych. Opisuje funkcje (specyfikacje funkcjonalne i niefunkcjonalne) systemu, wydajność systemu oraz ograniczenia operacyjne i związane z interfejsem użytkownika.

## 6.3 Formalizacja (K2)

20 minut

### Pojęcia

Poziom formalny, Poziom nieformalny, Poziom średnio-formalny

### 6.3.1 Stopnie formalizacji (K2)

Specyfikacja wymagań może być stworzona na różnych poziomach formalizacji:

- Nieformalny
- Średnio-formalny
- Formalny

#### 6.3.1.1 Poziom Nieformalny

Podjęcie nieformalne do pisania specyfikacji oznacza, iż dokument jest napisany w języku naturalnym, bez żadnej formalnej notacji. To podejście może być zastosowane, kiedy odbiorcy nie mają doświadczenia z bardziej formalnymi i technicznymi językami specyfikacji i mogliby mieć trudności ze zrozumieniem zawartości dokumentu. Główną słabością tego podejścia jest to, iż jest ono niejednoznaczne i może prowadzić do nieporozumień i nadinterpretacji. Dodatkowo, specyfikacja nieformalna nie jest dobrą podstawą do implementacji oraz testowania, ponieważ nie jest dość jasna i precyzyjna.

#### 6.3.1.2 Poziom Średnio-formalny

Specyfikacja średnio-formalna powinna zawierać notację formalną i być dobrze ustrukturyzowana. Zwykle takie specyfikacje opierają się na określonych szablonach (często wydzielonych z odpowiednich standardów). Specyfikacje średnio-formalne mogą wyrażać wymagania w postaci modeli i używać sformalizowanego naturalnego języka.

Jedną z najpopularniejszych notacji używanych do dokumentacji średnio-formalnej jest UML oraz SysML.

#### 6.3.1.3 Poziom Formalny

Specyfikacja formalna jest matematycznym opisem oprogramowania, który może być użyty do wytworzenia implementacji. Specyfikacja formalna opisuje, co system powinien robić, zwykle nie opisuje tego, jak powinien to robić. Ponieważ taka specyfikacja jest oparta na matematycznych formułach i jest trudniejsza do nauczenia się, specyfikacja formalna jest używana stosunkowo rzadko i wymaga wiedzy matematycznej.

<b>6.4 Jakość Wymagań (K2)</b>	<b>20 minut</b>
--------------------------------	-----------------

## Pojęcia

Charakterystyki jakościowe, Lista kontrolna, Możliwość śledzenia, Przegląd, Walidacja, Weryfikacja

### 6.4.1 Wprowadzenie

#### Błędy wymagań jako przyczyna wysokich kosztów (K2)

Jako, że wymagania są podstawą do rozwijania systemu, każdy błąd lub pominięte wymaganie propaguje na inne procesów wytwórcze w projekcie. Należy pamiętać, że wady wynikające z niskiej jakości wymagań są droższe do naprawienia w późniejszych fazach projektu, niż inne rodzaje defektów. Ponadto, im później zostaną wykryte uszkodzenia, tym wyższe są koszty ich naprawy.

Dlatego stosowanie weryfikacji (czy tworzymy produkt poprawnie) i walidacji (czy tworzymy właściwy produkt) wymagań jest konieczne.

Wymagania powinny być udokumentowane a następnie testowane na podstawie kryteriów jakościowych (patrz: 1.1 Wymaganie (K2)).

### 6.4.2 Miary dla doskonalenia jakości i zapewnienia jakości wymagań (K2)

Do doskonalenia jakości i zapewnienia jakości wymagań można użyć następujących narzędzi oraz technik:

- Standardy i szablony
- Przeglądy i inspekcje
- Możliwość śledzenia
- Prototypowanie
- Obserwacja kryteriów jakościowych (kryteriami jakościowymi mogą być: kompletność, poprawność, zgodność specyfikacji wymagań z odpowiednimi standardami)

Jakość Specyfikacji Wymagań może być udoskonalona poprzez uwzględnienie następujących elementów:

- Opis celu dokumentu, zakresu, definicji i słownika

- Opis celów na różnych poziomach (na przykład: specyfikacja wymagań wysokopoziomowych ma inne cele, niż szczegółowa specyfikacja wymagań funkcjonalnych)
- Zdefiniowanie ograniczeń dotyczących projektu i implementacji
- Określenie stopni/priorytetów wymagań
- Jasne stwierdzenie, co system powinien robić, zamiast jak powinien to robić
- Udokumentowanie kwestii prawnych, założeń, reguł biznesowych i zależności
- Unikanie dodatkowych opisów diagramów, które są jasne i wystarczające (zamiana trudnych, abstrakcyjnych tekstów diagramami, jeśli to możliwe)
- Jasno określony katalog użytkowników oraz schemat uprawnień (prawa i uprawnienia użytkownika)
- Stosowanie ustrukturyzowanej prezentacji
- Stosowanie prostego, jasnego, precyzyjnego i jednoznacznego języka

<b>7 Analiza Wymagań (K2)</b>	<b>140 minut</b>
-------------------------------	------------------

### *Cele nauczania dla Poziomu Podstawowego Inżynierii Wymagań*

Cele określają wiedzę i umiejętności, uzyskiwane po ukończeniu każdego modułu.

#### **7.1 Wymagania i Rozwiązania (K1)**

- LO-7.1.1 Podać cel Analizy Wymagań (K1)
- LO-7.1.2 Opisać procedurę Analizy Wymagań (K2)
- LO-7.1.3 Wyjaśnić koncepcję różnicy strukturalnej pomiędzy wymaganiami a rozwiązaniami (K2)

#### **7.2 Metody i Techniki (K2)**

- LO-7.2.1 Podać różne modele Analizy Wymagań (K1)
- LO-7.2.2 Zastosować różne metody Analizy Wymagań w określonych typach modeli (K3)

#### **7.3 Analiza Obiektowa (K2)**

- LO-7.3.1 Opisać charakterystyki UML (K2)
- LO-7.3.2 Opisać charakterystyki SysML (K2)

#### **7.4 Szacowanie Kosztu (K2)**

- LO-7.4.1 Podać powody szacowania kosztów (K1)
- LO-7.4.2 Rozpoznać ważne czynniki wpływające na szacowanie kosztów (K1)

#### **7.5 Priorytetyzacja (K2)**

- LO-7.5.1 Zastosować procedurę priorytetyzacji w danym scenariuszu (K3)

#### **7.6 Akceptacja Wymagań (K2)**

- LO-7.6.1 Określić, co powinno być uwzględnione podczas akceptacji wymagań (K2)

<b>7.1 Wymagania i Rozwiązania (K1)</b>	<b>20 minut</b>
---	-----------------

### 7.1.1 Cel Analizy Wymagań (K2)

Celem Analizy Wymagań jest stworzenie rozwiązania dla implementacji wymagań. Analizy Wymagań bierze pod uwagę różnych interesariuszy projektu, potencjalnie sprzeczne wymagania, analizuje związki i zależności pomiędzy wymaganiami itp.

### 7.1.2 Procedura Analizy Wymagań (K2)

Procedura Analizy Wymagań obejmuje następujące kroki:

1. Analiza potrzeb
2. Opis rozwiązania
3. Szacowanie kosztu oraz priorytetyzacja

### 7.1.3 Różnica strukturalna pomiędzy wymaganiami a rozwiązaniami (K2)

Różnica strukturalna pomiędzy wymaganiami a rozwiązaniami oznacza, iż wymagania i rozwiązania różnią się. Rozwiązanie jest implementacją wymagania. Model wymagań może być postrzegany jako model biznesowy, przedstawiający problem biznesowy, który należy rozwiązać poprzez model rozwiązania. Model rozwiązania jest bardziej szczegółowy i obejmuje techniczną specyfikację wymagań; jest podstawą do wytwarzania i testowania.

Istnieją trzy podstawowe poziomy modeli:

- Model wymagań
  - Często specyfikacja nieformalna
  - Notacje modelowania biznesowego (BPMN – Business Process Modeling Notation)
- Model rozwiązania
  - Struktury funkcjonalne (algorytmy, procedury, przebieg czynności)
- Model konceptualny
  - Specyfikacje technologiczne oprogramowania/sprzętu

Pomiędzy tymi modelami powinno istnieć śledzenie.

## 7.2 Metody i Techniki (K2)

20 minut

### Pojęcia

Dekompozycja funkcjonalna, Model kontekstowy, Model przejść stanów, Model przepływu danych, Model związków encji, Modele koncepcyjne, Modele rozwiązań, Modele wymagań

### 7.2.1 Metody i modele analizy

Różne aspekty system są reprezentowane poprzez różne widoki.

Modele rozwija się poprzez odpowiednie metody analizy.

Metoda analizy	Model analizy
Analiza kontekstu	Model kontekstowe
Analiza architektury	Dekompozycja funkcjonalna
Analiza przepływu danych	Model przepływu danych
Analiza warunków	Model warunków
Analiza decyzji	Tabela decyzyjna
Analiza danych	Model semantyczny danych Model związków encji Słownik danych

## 7.2.2 Typy modeli (K2)

Istnieją trzy główne typy modeli:

- Model wymagań
  - Opisuje obszar problemu
  - Projektowany we wczesnych etapach projektu
  - Służy celom Analizy Wymagań i szacowaniu kosztów
  - Dostarcza podstaw dla modelu rozwiązania
  - Przykład: model biznesowych przypadków użycia, model klas biznesowych, model procesów biznesowych
- Model rozwiązania
  - Opisuje obszar rozwiązania z różnych punktów widzenia na system
  - Projektowany równocześnie z modelem wymagań
  - Określa kształt implementacji wymagań funkcjonalnych oraz нефункциональных
  - Dostarcza podstaw do projektu systemu
  - Przykład: model przypadków użycia, model sekwencji, model aktywności, model przejść stanów
- Model konceptualny
  - Specyfikacje technologiczne oprogramowania/sprzętu (moduły, komponenty sprzętowe, charakterystyki PC)

## 7.2.3 Różne perspektywy systemu (K2)

Niektóre z punktów widzenia związanych z systemem:

- Widok logiczny
  - Wymagania funkcjonalne
- Widok procesu
  - Komunikacja
  - Interakcje
  - Wymagania нефункциональные
- Widok implementacji
  - Komponenty (moduły)

- Widok instalacji
  - Integracja
  - Architektura systemu

#### 7.2.4 Różne modele (K1)

Można zastosować następujące modele:

- Model kontekstowy
  - Statyczny opis systemu
  - Wyraża podstawową architekturę
- Dekompozycja funkcjonalna
  - Statyczny opis systemu
  - Wyraża sukcesywną dekompozycję system
- Model przepływu danych
  - Dynamiczny opis systemu
  - Graficzna reprezentacja przepływu danych w systemie
  - Nie dostarcza informacji o czasach trwania procesów i o tym, czy procesy wykonują się sekwencyjnie, czy równolegle
- Model przejść stanów
  - Dynamiczny opis systemu
  - Reprezentuje zachowanie system jako ciąg zdarzeń, które mogą wystąpić w jednym lub więcej stanie
- Model związków encji
  - Abstrakcyjna i konceptualna reprezentacja danych
  - Zawiera elementy składowe: encje, związki oraz atrybuty

Kiedy należy zastosować określony model? Różne modele odpowiadają na różnego typu pytania odnośnie danego rozwiązania.

- Model kontekstowy ⇒ CO (reprezentacja głównych przepływów w systemie i na zewnątrz systemu)
- Dekompozycja funkcjonalna ⇒ CO (jakie funkcje i cechy istnieją w zakresie systemu)
- Model przepływu danych ⇒ CO (przepływu pomiędzy procesami biznesowymi/funkcjami/czynnościami)
- Model związków encji ⇒ CO (jakie istnieją związki pomiędzy określonymi encjami (obiektami) systemu)
- Model przejść stanów ⇒ DLACZEGO (przyczyna i skutek)

## 7.3 Analiza Obiektowa (K2)

30 minut

### Pojęcia

Analiza Obiektowa, Diagram aktywności, Diagram behawioralny, Diagram klas, Diagram komponentów, Diagram komunikacji, Diagram maszyny stanów, Diagram obiektów, Diagram pakietów, Diagram przeglądu interakcji, Diagram przypadków użycia, Diagram sekwencji, Diagram strukturalny, Diagram struktury złożonej, Diagram wdrożenia, Diagram wymagań, SysML, UML

### 7.3.1 UML (K1)

Unified Modeling Language jest ujednoczoną notacją dla analizy i projektowania systemów. Zawiera różne rodzaje diagramów dla różnych punktów widzenia na system (diagramy strukturalne i behawioralne).

#### 7.3.1.1 Diagramy behawioralne (K2)

Diagramy behawioralne przedstawiają cechy zachowania systemu lub procesu biznesowego.

Diagramy te obejmują następujące typy diagramów:

- Diagramy aktywności
  - Modelują zachowanie system oraz sposób, w jaki te zachowania są powiązane z ogólnym przepływem system
- Diagramy przypadków użycia
  - Przechwytyją przypadki użycia i relacje pomiędzy aktorami a systemem
  - Opisują wymagania funkcjonalne systemu, sposób, w jaki sposób operatorzy zewnątrz oddziałują na granicy systemu oraz odpowiedzi systemu
- Diagramy maszyny stanów
  - Prezentują, w jaki sposób element może przechodzić pomiędzy stanami, klasyfikują jego zachowanie zgodnie z wyzwalaczami przejść oraz ograniczeniami
- Diagramy czasowe
  - Definiują zachowanie różnych obiektów na skali czasowej
  - Dostarczają wizualnej reprezentacji o zmianach stanów obiektów oraz interakcji w przeciągu czasu
- Diagramy sekwencji

- Ustrukturyzowana reprezentacja zachowania jako ciągu sekwencyjnych kroków w przeciągu czasu
- Używane do prezentacji przepływu zadań, przepływu komunikatów oraz tego, jak w przeciągu czasu ogólnie współdziałają elementy, by osiągnąć dany rezultat
- Diagramy komunikacji
  - Prezentują interakcję pomiędzy elementami w czasie rzeczywistym, wizualizując relacje międzyobiektywne
- Diagramy przeglądu interakcji
  - Wizualizują kooperację pomiędzy innymi diagramami współpracy w celu przedstawienia przepływu kontroli służącemu danemu celowi

### 7.3.1.2 Diagramy strukturalne (K2)

Diagramy strukturalne przedstawiają strukturalne elementy składowe systemu lub funkcji. Diagramy te odzwierciedlają statyczne relacje struktury, takie jak diagramy klas lub pakietów, lub architekturę w czasie działania, jak diagramy obiektów lub struktury złożonej.

Diagramy strukturalne obejmują następujące typy diagramów:

- Diagramy klas
  - Przedstawiają logiczną strukturę systemu, klasy oraz obiekty tworzące model, opisujące, co istnieje i jakie posiada atrybuty oraz zachowanie
- Diagramy struktury złożonej
  - Odzwierciedlają wewnętrzną współpracę klas, interfejsów oraz komponentów (i ich właściwości) celem opisu funkcjonalności
- Diagramy komponentów
  - Prezentują części oprogramowania tworzące system oraz ich organizację i zależności
- Diagramy wdrożenia
  - Opisują architekturę wykonawczą systemu
- Diagramy obiektów
  - Prezentują obiektywne wystąpienia klas oraz ich związki w danym punkcie czasu
- Diagramy pakietów
  - Przedstawiają organizację elementów modelu w pakiety oraz zależności pomiędzy nimi

### 7.3.2 SysML (K2)

System Modeling Language jest specjalnym językiem modelowania dla Inżynierii Systemów. Jest rozszerzeniem UML 2.1.

SysML oferuje pewne ulepszenia nad UML. Te ulepszenia to:

- Bardziej elastyczna i ekspresyjna semantyka
  - Redukuje ograniczenia UML związane z orientacją na oprogramowanie i dodaje dwa nowe typy diagramów, diagram wymagań oraz diagram parametryczny
  - Umożliwia modelowanie szerokiego zakresu systemów, które obejmują sprzęt, oprogramowanie, informację, procesy, personel i urządzenia
- Łatwy do nauki i zastosowania
  - Mniejszy, niż UML (nie wykorzystuje wielu konstrukcji UML zorientowanych na oprogramowanie)
- Wsparcie modeli i widoków
  - Modele i widoki są architektonicznie zgodne z IEEE-Std-1471-2000
- Wykorzystuje siedem z diagramów UML i wprowadza dwa nowe typy diagramów (diagram wymagań oraz diagram parametryczny) dla całkowitej liczby dziewięciu typów diagramów
  - Diagramy wymagań do prezentacji wymagań funkcjonalnych, wydajnościowych oraz interfejsowych
  - Diagram parametryczne do definiowania ograniczeń wydajnościowych oraz ilościowych

## 7.4 Szacowanie Kosztów (K2)

20 minut

### Wprowadzenie

Szacowanie kosztów wiąże Inżynierię Wymagań z Zarządzaniem Projektem.

#### 7.4.1 Typy estymat (K2)

Najbardziej popularnymi aspektami będącymi przedmiotem szacowania są:

- Koszty
- Czas
- Wymagania

Estymaty kosztu pomagają rozpoznać koszt wytwarzania, zmian itp.

#### 7.4.2 Czynniki wpływające na koszty wytwarzania (K2)

Koszt projektu zależy od wielu czynników:

- Typ projektu
- Dojrzałość procesu
- Metody i narzędzia projektowe oraz testowe
- Technologia
- Złożoność planowanego rozwiązania
- Cele jakościowe (na przykład: pożądany poziom jakości oprogramowania)
- Kwalifikacje zespołu
- Podział zespołu
- Doświadczenie

Dokładność szacowania kosztów zależy od stopnia postępu projektu oraz jego dojrzałości.

### 7.4.3 Podejścia do szacowania kosztu

Szacowanie kosztu może być zrealizowane przy użyciu:

- Konkluzji przez analogię
- Procedury algorytmicznej

#### 7.4.3.1 Konkluzja przez analogię (K2)

Szacowanie kosztu jest oparte na porównaniu kosztów podobnego projektu. Opiera się na doświadczeniu, nie na wzorach matematycznych. Z tą techniką, obecny projekt jest porównywany z przeszłymi projektami.

Porównanie może obejmować:

- Liczbę wymagań
- Zakres rozwiązania
- Zastosowaną technologię
- Charakterystyki personelu (umiejętności, doświadczenie)

W oparciu o wyniki porównania, można opracować oszacowanie.

Procedury szacowania zawsze są oparte na danych historycznych oraz warunkach środowiskowych.

#### Metoda Delphi

Jest to ustrukturyzowana metoda komunikacji używana do przeprowadzania interaktywnego przewidywania. Angażuje panel ekspertów [Linstone75].

Eksperci są proszeni o udzielenie odpowiedzi na kwestionariusz w kilku rundach. Po każdej rundzie, wykonywane jest anonimowe podsumowanie prognoz ekspertów wraz z uzasadnieniem osądu. Następnie eksperci rewidują swoje wcześniejsze odpowiedzi uwzględniając odpowiedzi innych członków panelu.

Wierzy się, że podczas tego procesu zakres odpowiedzi będzie się zmniejszać a grupa będzie dążyć do "właściwej" odpowiedzi.

Proces jest zatrzymywany po określonym kryterium końca. Średnie wyniki ostatniej rundy określają rezultat.

#### Szacowanie zwinne

W projektach zwinnych, często zarządzanych przez Scrum, istnieje metoda estymacji zwana Grą Planistyczną lub Pokerem Planistycznym. Metody tej używa się w celu uzyskania porozumienia w zespole. Możliwości zespołu są następnie mierzone za pomocą Wskaźnika Wypalenia i

doskonalone poprzez sesje Retrospektywy wykonywane po każdym przebiegu, gdzie planowane liczby są porównywane z aktualnymi. Umożliwia to udoskonalenie zdolności zespołu w szacowaniu.

#### 7.4.3.2 Procedura algorytmiczna (K2)

W tym podejściu koszty wyliczane są na podstawie parametrów. Parametry mogą opisywać produkt (wielkość, czas trwania), warunki graniczne (wydajność) itp.

Można zastosować następujące metody:

- Równanie Putnama
- Punkty Funkcyjne
- Constructive Cost Model (CoCoMo)

#### Równanie Putnama [Putnam91]

$$\text{Wysiłek} = [\text{Rozmiar}/\text{Produktywność} * \text{Czas}^{4/3}]^3 * B$$

Gdzie:

- Rozmiar – rozmiar produktu (szacowany przy użyciu dowolnej estymaty rozmiaru stosowanej przez organizację). Równanie Putnama używa ESLOC (Efektywne Linie Kodu Źródłowego)
- B – wskaźnik skalujący; jest funkcją rozmiaru projektu
- Produktywność – produktywność procesu, zdolność określonej organizacji do wyprodukowania oprogramowania o danym rozmiarze i przy określonym wskaźniku defektów
- Czas – całkowity czas trwania projektu w latach
- Wysiłek – całkowity wysiłek odnoszący się do projektu w osobolatach

#### Punkty Funkcyjne

Punkt funkcyjny jest jednostką pomiaru wyrażającą ilość funkcjonalności biznesowej dostarczonej użytkownikowi przez system informacyjny. Koszt pojedynczego punktu funkcyjnego jest wyliczany na podstawie przeszłych projektów.

Istnieje pięć standardowych „funkcji” do zliczania jako punkty funkcyjne.

- Funkcje danych:
  1. Wewnętrzne pliki logiczne
    - Tabele w relacyjnej bazie danych
    - Informacja kontrolna aplikacji
  2. Pliki interfejsów zewnętrznych
    - Tabele w relacyjnej bazie danych, informacja kontrolna aplikacji nie utrzymywane przez rozważaną aplikację
- Funkcje transakcyjne:
  1. Wejścia zewnętrzne
    - Dane wprowadzane przez użytkowników
    - Dane lub pliki zasilane przez aplikacje zewnętrzne
  2. Wyjścia zewnętrzne
    - Raporty tworzone przez daną aplikację zawierające dane pochodne
  3. Zapytania zewnętrzne
    - Raporty tworzone przez aplikację, która jest szacowana, w przypadku gdy raporty nie zawierają danych pochodnych

Zidentyfikowane funkcje są ważone według trzech poziomów złożoności; liczby punktów przypisanych do każdego poziomu różni się w zależności od typu punktu funkcyjnego.

Przykład przedstawiony jest poniżej:

Złożoność	Punkty
Niska	7
Średnia	10
Wysoka	15

CoCoMo

CoCoMo umożliwia wyliczenie nakładu pracy oraz kosztu oprogramowania w zależności od wielkości programu. Wielkość oprogramowania jest wyrażona w EKLOC (szacowane tysiące linii kodu).

CoCoMo stosuje się w trzech klasach projektów informatycznych:

- Projekt łatwy
- Projekt średni
- Projekt złożony

Podstawowe równania CoCoMo:

- Nakład pracy (E) =  $a_b(\text{KLOC})^{b_b}$  [osobomiesiący]
- Czas wytwarzania (D) =  $c_b(\text{Nakład pracy})^{d_b}$  [miesiące]
- Liczba ludzi (P) = Nakład pracy / Czas wytwarzania [liczba]

KLOC – szacowana ilość dostarczonych linii kodu dla projektu (wyrażona w tysiącach)

$a_b$ ,  $b_b$ ,  $c_b$  oraz  $d_b$ :

Projekt	$a_b$	$b_b$	$c_b$	$d_b$
Prosty	2.4	1.05	2.5	0.38
Średni	3.0	1.12	2.5	0.35
Złożony	3.6	1.20	2.5	0.32

<b>7.5 Priorytetyzacja (K2)</b>	<b>20 minut</b>
---------------------------------	-----------------

## Pojęcia

Skala, Skala trzy-stopniowa

### 7.5.1 Priorytetyzacja (K2)

Priorytetyzacja umożliwia ustalenie względnej ważności wymagań i implementację najbardziej krytycznych wymagań w pierwszej kolejności.

Priorytetyzacja wspiera wytwarzanie przyrostowe, jako że umożliwia grupowanie wymagań oraz określanie priorytetów implementacji.

### 7.5.2 Procedura ustalania priorytetów (K2)

Procedura ustalania priorytetów wymagań obejmuje następujące czynności:

1. Grupowanie wymagań
  - Identyfikacja wymagań wpływających na siebie i zależnych od siebie (na przykład: wymagania tworzące jedną złożoną funkcjonalność)
2. Analiza wymagań
  - Analiza wykonywana przez wszystkich zaangażowanych interesariuszy w celu uzgodnienia poziomu ważności
  - Analiza wpływu jako środek wspierający analizę wymagań
  - Ustalenie priorytetów wymagań
3. Stworzenie planu projektu wymagań
  - Ustalenie planu, w którym wymagania o wysokich priorytetach będą wytwarzane jako pierwsze
  - Przypisanie odpowiedzialności (kto implementuje dane wymaganie)
4. Planowanie testowanie przyrostów system
  - Zaprojektowanie przypadków testowych dla testów każdego przyrostu systemu (ustalonych na podstawie uszeregowanych wymagań) na bazie priorytetów wymagań

### 7.5.3 Skale priorytetowe (K2)

Popularnym podejściem do priorytetów jest grupowanie wymagań w kategoriach priorytetowych. Zazwyczaj używana jest skala trzy-stopniowa (na przykład: Wysoki, Średni i Niski).

Ze względu na to, że takie skale są subiektywne i nieprecyzyjne, wszyscy zaangażowani interesariusze muszą uzgodnić znaczenie każdego poziomu w skali, których używają. Definicja priorytetu powinna być wyraźnie określona i powinna być kluczowym atrybutem każdego wymagania.

Przykładami skal trzy-stopniowych są:

Nazwa	Opis
Wysoki	Wymaganie krytyczne, wymagane w pierwszym wydaniu systemu
Średni	Wspiera wymagane operacje systemu, ale w razie konieczności może poczekać do późniejszego wydania
Niski	Udoskonalenie funkcjonalne lub jakościowe, tak zwane "byłoby było mieć"

Nazwa	Opis
Kluczowe	Produkt nie jest akceptowalny dopóki nie są spełnione takie wymagania
Warunkowe	Ulepszony produkt, ale w przypadku braku produkt nie jest nieakceptowany
Opcjonalne	Funkcje, które mogą być lub mogą nie być wartościowe

<b>7.6 Akceptacja Wymagań (K2)</b>	<b>20 minut</b>
------------------------------------	-----------------

## Pojęcia

Podpisywanie wymagań

### 7.6.1 Akceptacja (K2)

Uzgodnienie wymagań, często nazywane podpisywaniem wymagań, jest formalnym porozumieniem mówiącym, iż treść i zakres wymogów są dokładne i kompletne.

Formalna akceptacja jest podstawą projektu. Wymagania wysokopoziomowe (wymagania biznesowe) powinny zostać uzgodnione przed rozpoczęciem projektu. Szczegółowe wymagania (funkcjonalne i нефункционалне) powinny być uzgodnione i podpisane przed przejściem do etapu realizacji.

Uzyskanie akceptacji wymagań jest zwykle zadaniem analizy i projektowania wymagań.

Akceptacja wymagań powinna być wykonywana przez interesariuszy projektu, włączając:

- Kierowników Projektu po stronie klienta oraz dostawcy
- Reprezentantów biznesu klienta
- Analityków biznesowych i systemowych
- Inżynierów wymagań
- Reprezentantów zespołów zapewnienia jakości, testowania oraz programistycznych

Lista wymagań musi być wiążąca dla obu stron: klienta i dostawcy.

Jednym z celów akceptacji wymagań jest zapewnienie, że wymagania są stabilne, a wszelkie zmiany będą zarządzane poprzez formalne Żądania Zmian. Dlatego też formalna akceptacja zmniejsza ryzyko wprowadzenia nowych wymagań w trakcie lub po ukończeniu implementacji.

Uzgodnienie wymagań jest uznane za kompletne, gdy wszyscy istotni interesariusze podpisali dokumenty wymagań.

Zakończenie akceptacji wymagań powinno być zakomunikowanego zespołowi projektowemu i zazwyczaj jest kamieniem milowym projektu.

### 7.6.2 Korzyści z akceptacji wymagań (K1)

- Akceptacja zapewnia rozwój właściwego produktu (działania są oparte o uzgodnioną listę wymagań i obejmują tylko to, co jest potrzebne do uzyskania wartości dodanej dla interesariuszy)
- Zminimalizowane ryzyko nieporozumień pomiędzy klientem a dostawcą odnośnie zakresu projektu
- Podstawa dla dalszych prac projektowych

<b>8 Śledzenie Wymagań (K2)</b>	<b>60 minut</b>
---------------------------------	-----------------

*Cele nauczania dla Poziomu Podstawowego Inżynierii Wymagań*  
Cele określają wiedzę i umiejętności, uzyskiwane po ukończeniu każdego modułu.

### **8.1 Śledzenie w Projekcie (K2)**

- LO-8.1.1 Podać znaczenie możliwości śledzenia (K1)
- LO-8.1.2 Podać typowe powody zmian w wymaganiach (K1)
- LO-8.1.3 Opisać cel możliwości śledzenia (K2)
- LO-8.1.4 Rozpoznać różne rodzaje możliwości śledzenia (K1)

### **8.2 Zarządzanie Zmianą (K2)**

- LO-8.2.1 Opisać charakterystyki Zarządzania Zmianą (K2)
- LO-8.2.2 Podać skład Komitetu Kontroli Zmian (K1)

## 8.1 Śledzenie w Projekcie (K2)

20 minut

### Pojęcia

Śledzenie horyzontalne i wertykalne

#### 8.1.1 Rozwój wymagań (K1)

Wymagania nie są stabilne, ale rozwijają się trakcie cyklu życia projektu.

Powody, dla ciągłego rozwoju i proponowanych zmian mogą być następujące:

- Nowe pomysły
- Nowe potrzeby klienta (wynikające na przykład z nowych przepisów, zmian w biznesie, nowych produktów itp.)
- Kontynuowana praca (na przykład: następna faza projektu, ulepszanie i optymalizacja już zaimplementowanych funkcjonalności itp.)
- Nowe powiązania w projekcie (na przykład: integracja z nowymi systemami, nowe kanały dostępu: kanały mobilne lub internetowe itp.)

#### 8.1.2 Możliwość śledzenia (K2)

Możliwość śledzenia dostarcza rozwiązania do zarządzania rozwijającymi się wymaganiami i innymi, związanymi z nimi, artefaktami.

Możliwość śledzenia zapewnia kontrolę, że wszystkie ważne etapy procesu wytwarzania zostały przeprowadzone. Powinna być wykonana dwukierunkowo dla wszystkich artefaktów (na przykład: od wymagań do artefaktów projektowych oraz od artefaktów projektowych do wymagań). Możliwość śledzenia jest również ważna dla testowania, weryfikacji i walidacji.

Cele możliwości śledzenia:

- Analiza wpływu
- Analiza pokrycia
- Dowód implementacji
- Użycie wymagań (śledzenie wymagań jako dowód, iż wymagania są użyte oraz jak są użyte)

W celu zapewnienia dobrej możliwości śledzenia, ważne jest, by nadawać wymaganiom unikalne nazwy.

### 8.1.3 Typy możliwości śledzenia (K2)

- Śledzenie horyzontalne
  - Przedstawia zależności pomiędzy wymaganiami na tym samym poziomie (związki pomiędzy różnymi typami wymagań)
- Śledzenie wertykalne
  - Przedstawia zależności pomiędzy różnymi artefaktami (wymaganiami a specyfikacją rozwiązania lub specyfikacją wymagań, przypadkami testowymi, kodem, modułami, planami itp.)

<b>8.2 Zarządzanie Zmianą (K2)</b>	<b>20 minut</b>
------------------------------------	-----------------

## Pojęcia

Komitet Kontroli Zmian, Zarządzanie Zmianą, Zmiana, Żądanie Zmiany

### 8.2.1 Zmiany wymagań (K1)

Zmiany wymagań mogą być zgłoszone w dowolnym momencie w trakcie realizacji projektu oraz po wydaniu gotowego oprogramowania w środowisku produkcyjnym. Zmiany będą zdarzać się zawsze i ważne jest, aby je planować w kontekście procesu i czasu.

Źródłem zmian mogą być:

- Rozszerzenia istniejącej funkcjonalności
- Defekty znalezione w oprogramowaniu/dokumentacji
- Inne problemy wykryte na przykład podczas testowania, takie jak słaba wydajność itp.
- Zgłoszenie nowej funkcjonalności lub zmiana istniejącej funkcjonalności
- Zmiany wynikające z czynników zewnętrznych (zmiany organizacyjne, prawne)

### 8.2.2 Zarządzanie Zmianą (K2)

Proces Zarządzania Zmianą to proces wnioskowania, określania możliwości, planowania, implementowania i oceny zmian w systemie oprogramowania, dokumentach lub innych produktach projektu. Celem Zarządzania Zmianą jest umożliwienie i wsparcie przetwarzania zmian i zapewnienie możliwości śledzenia zmian.

Proces Zarządzania Zmianą obejmuje następujące czynności:

1. Identyfikacja potencjalnej zmiany
2. Wnioskowanie o nową funkcjonalność
3. Analiza Żądania Zmiany
4. Ocena zmiany
5. Planowanie zmiany
6. Implementacja zmiany
7. Przegląd oraz zamknięcie zmiany

## 8. Wdrożenie zmiany

W zależności od złożoności i wpływu, zmiana może mieć różne skutki dla systemu. Małe zmiany mogą wymagać drobnych modyfikacji, natomiast skomplikowane mogą radykalnie zmienić logikę systemu. Wszelkie zmiany powinny być dokładnie przeanalizowane w celu określenia ryzyk i oceny wartości modyfikacji względem przewidywanego ryzyka.

### 8.2.3 Żądanie Zmiany (K2)

Zmiana powinna być zgłoszona jako formalny dokument Żądania Zmiany (zwany także CR lub Request for Change (RFC)). Taki dokument zwykle opisuje powód zmiany, priorytet oraz pożądane rozwiązanie wraz z dodatkowymi informacjami, takimi jak:

- Dane osoby/działu lub innej jednostki zgłaszającej zmianę
- Data zgłoszenia
- Planowana data implementacji zmiany (jeśli dotyczy)
- Koszt zmiany

### 8.2.4 Komitet Kontroli Zmian (K1)

Komitet Kontroli Zmian (Change Control Board – CCB) sprawdza i podejmuje decyzje odnośnie zmian. CCB wspiera kontrolowany sposób implementacji zmiany, lub proponowanej zmiany, w produkcji lub usłudze.

Komitet Kontroli Zmian jest grupą, która w oparciu o dostarczone informacje (takie jak ryzyko związane ze zmianą, jej wpływ, nakład pracy wymagany do implementacji) podejmuje decyzje odnośnie tego, czy należy wdrożyć proponowane zmiany. CCB składa się z interesariuszy projektu lub ich reprezentantów.

Komitet Kontroli Zmian może składać się z następujących ról:

- Kierownictwo projektu
- Kierownictwo wytwarzania
- Zapewnienie jakości (zarządzanie jakością, zarządzanie testami)
- Kierownictwo biznesowe, jeśli dotyczy
- Klient, jeśli dotyczy

### 8.2.5 Cykl życia wymagania (K2)

W zależności od stopnia analizy lub/i implementacji, wymaganie będzie posiadało inny status. Cykl życia wymagania może być wyrażony poprzez następujące przykładowe statusy:

- Nowe (proponowane)
- Do przeglądu
- Zatwierdzone
- Sprzeczne
- Zaimplementowane
- Zmodyfikowane
- Usunięte
- Przetestowane
- Wdrożone

Różne podejścia oraz organizacje mogą używać różnych cykli życia (i statusów) wymagania. W wielu przypadkach, cykle życia wymagań, zmian oraz defektów są bardzo podobne i obsługiwane przez to samo narzędzie.

### 8.2.6 Rozróżnienie pomiędzy Zarządzaniem Defektami a Zarządzaniem Zmianą (K2)

Ważne jest, by rozróżnić pomiędzy zmianą a defektem. Defekt definiowany jest błąd w komponencie lub systemie, który może spowodować, że komponent lub system nie będzie zdolny do wykonywania wymaganych funkcji. Jest to odstępstwo od wymaganego stanu systemu. Zmiana jest modyfikacją istniejących lub żądaniem nowych cech, wymagań lub funkcji.

### 8.2.7 Wpływ zmiany na projekt (K2)

Zmiany w wymaganiach mogą mieć różne wpływy na projekt. Najbardziej powszechne to:

- Zmiana harmonogramu, budżetu, zasobów
- Prace wynikające ze zmiany (w zależności od fazy projektu):
  - Aktualizacja artefaktów analitycznych oraz projektowych (na przykład: specyfikacje)
  - Aktualizacja dokumentacji technicznej i użytkownika

- Zmiany w strategii testów oraz testach
- Aktualizacja Planu Testów
- Aktualizacja potrzeb/planu szkoleniowego
- Rozszerzenie/skrócenie zakresu prac programistycznych
- Zmiany zakresu przygotowania oraz wykonania testów

<b>9 Zapewnienie Jakości (K2)</b>	<b>30 minut</b>
-----------------------------------	-----------------

*Cele nauczania dla Poziomu Podstawowego Inżynierii Wymagań*  
Cele określają wiedzę i umiejętności, uzyskiwane po ukończeniu każdego modułu.

### **9.1 Czynniki Oddziałujące (K1)**

LO-9.1.1 Podać czynniki oddziałujące na Inżynierię Wymagań (K1)

### **9.3 Zapewnienie Jakości poprzez Testowalność (K2)**

LO-9.3.1 Wyjaśnić, w jaki sposób produkty Inżynierii Wymagań wspierają testowanie (K2)

LO-9.3.2 Opisać zastosowanie kryteriów akceptacji (K2)

LO-9.3.3 Opisać, w jaki sposób Inżynieria Wymagań kontrybuuje do testowania (K2)

### **9.4 Metryki (K2)**

LO-9.4.1 Podać definicję metryki (K1)

LO-9.4.2 Opisać, jakie metryki mogą być użyte w Inżynierii Wymagań (K2)

<b>9.1 Czynniki Oddziałujące (K1)</b>	<b>10 minut</b>
---------------------------------------	-----------------

### 9.1.1 Czynniki oddziałujące na Inżynierię Wymagań (K1)

Jakość produktów Inżynierii Wymagań zależy od następujących czynników:

- Produkt, który jest wytwarzany
- Środowisko, w którym produkt jest wytwarzany
- Dziedzina (złożoność dziedziny biznesowej, poziom innowacji, częstotliwość zmian w biznesie itp.)
- Czynniki prawne, bezpieczeństwa oraz środowiskowe
- Presja czasowa i finansowa (ograniczenia czasowe i kosztowe mogą zmniejszać możliwości uruchomienia procesów Inżynierii Wymagań)
- Czynniki kulturowe (język, edukacja itp.)
- Technologia i ograniczenia projektowe

Podczas planowania czynności zapewnienia jakości, należy uwzględnić takie czynniki.

<b>9.2 Weryfikacja wymagań w fazie pozyskiwania wymagań (K2)</b>	<b>20 minut</b>
--	-----------------

Weryfikacja powinna być stale wykonywana podczas wytwarzania rozwiązania. Do celów weryfikacji wymagań w fazie pozyskiwania wymagań można zastosować następujące techniki:

- Techniki przeglądów
- Symulacje
- Prezentacje
- Demonstracje
- Dema

Ważne jest, by weryfikacja była planowana od początku projektu.

<b>9.3 Zapewnienie Jakości poprzez Testowalność (K2)</b>	<b>20 minut</b>
--	-----------------

## Pojęcia

Kryteria Akceptacji, Testowalność

### 9.3.1 Inżynieria Wymagań a testowanie (K2)

Inżynieria Wymagań jest ściśle powiązana z testowaniem. Dobre przypadki testowe wymagają dobrych wymagań, które mogą być testowane. Z tego powodu zaangażowanie testerów w specyfikację jest bardzo ważne.

### 9.3.2 Kryteria Akceptacji (K2)

Zgodnie z nomenklaturą Prince2™, Kryteria Akceptacji, zwane również Kryteriami Sukcesu, są standardami wymaganymi, by spełnić oczekiwania jakościowe klienta i uzyskać akceptację finalnego produktu. Innymi słowy, są to kryteria nałożone na określoną funkcję, komponent lub dowolny inny składnik produktu oprogramowania lub innego produktu projektu, które muszą być spełnione, by usatysfakcjonować klienta i uzyskać jego akceptację odnośnie produktu

Kryteria Akceptacji powinny być uzgodnione przez obie strony – dostawcę oraz klienta – przed rozpoczęciem projektu (powinny być częścią dokumentacji kontraktowej) i będą stanowić bazę do Planu Jakości Projektu. Każde wysokopoziomowe wymaganie musi mieć co najmniej jedno kryterium akceptacji. Kryteria te są podstawą Testów Akceptacyjnych.

Każde kryterium musi być mierzalne a środki pomiaru muszą być realistyczne i zaakceptowane.

Przykładem Kryterium Akceptacji jest: „Aplikacja musi odpowiadać w czasie krótszym, niż jedna sekunda, w przeciwnym przypadku musi wyświetlać komunikat oczekiwania”.

### 9.3.3 Metody dla testowania (K2)

Produkty Inżynierii Wymagań wspierają testowanie poprzez dostarczanie tak zwanych podstaw testowych. Wymagania i ich specyfikacje mogą być podstawą testową.

Produkty inżynierii Wymagań mogą wspierać testowanie przez:

- Umożliwienie pokrycia funkcjonalnego (pokrycie wszystkich wymagań funkcjonalnych przypadkami testowymi; przypadki testowe są pisane tak, by pokryć wszystkie wymagania funkcjonalne)

Pokrycie funkcjonalne =  $\sum$  Wymagań testowanych przy użyciu przypadków testowych /  $\sum$  Wymagań

- Dostarczenie wiarygodnych podstaw testowych dla “testów czarno-skrzynkowych” lub testowania “opartego o specyfikację”, takich jak:
  - Wartości brzegowe dla kategorii danych wejściowych
  - Stany aplikacji
  - Ograniczenia logiczne i biznesowe itp.
- Umożliwienie takich technik testowych, jak: przedziały równoważności, analiza wartości brzegowych, testowanie przy użyciu tabeli decyzyjnych, testowanie przejść stanów, testowanie oparte o przypadki użycia.
- Przykład: przedziały równoważności – jest to technika testowania, która polega na dzieleniu danych wejściowych używanych w określonym module oprogramowania w partycje danych, z których można wyodrębnić przypadki testowe. Przypadki testowe projektuje się tak, by co najmniej raz pokryć każdą partycję.

Przykład przedziałów równoważności:

1. Prawidłowy przedział wieku użytkownika to od 1 do 99. Prawidłowy przedział określany jest jako prawidłowa partycja. Istnieją jeszcze dwie inne partycje zakresów nieprawidłowych. Pierwsza nieprawidłowa partycja to  $\leq 0$  a druga to  $\geq 100$ .
2. Prawidłowy zakres dla odpowiedzi w kwestionariuszu obejmuje następujące znaki: A, B, C, D. Nieprawidłowa partycja będzie obejmować wszystkie inne znaki tekstowe, włączając znaki specjalne.

Przypadki testowe powinny uwzględniać prawidłową partycję oraz wszystkie nieprawidłowe partycje.

Istnieją inne techniki testowe: analiza wartości brzegowych, testowanie przy użyciu tabeli decyzyjnych, testowanie przejść stanów, testowanie oparte o przypadki użycia. Powinny one być używane po analizie testowej wymagań i zastosowane zgodnie z treścią wymagań.

### 9.3.4 Wymagania a proces testowy (K2)

Wymagania są podstawową wejściową informacją dla procesu wytwarzania i testowania systemu. Dobrze zdefiniowane wymagania zmniejszają ryzyko porażki projektu (lub nawet więcej, produktu), ponieważ umożliwiają staranne testowanie. Stabilność wymagań umożliwia spełnienie terminów określonych zadań.

Ważne jest, aby pamiętać, że wymagania powinny być sprawdzone przez testy statyczne (co może angażować testerów) i zaakceptowane przez kierowników testów (ponieważ w niektórych przypadkach wymagania mogą okazać się nietestowalne). Testerzy pomagają poprawić jakość wymagań, wskazując słabe punkty i ewentualne wady. Testerzy powinni także uczestniczyć w weryfikacji wymagań w celu zapewnienia testowalności.

<b>9.4 Metryki (K2)</b>	<b>20 minut</b>
-------------------------	-----------------

## Pojęcia

Metryka

### 9.4.1 Metryka (K1)

Metryka jest skalą pomiaru i metodą używaną do pomiaru [ISO 14598].

Metryki umożliwiają wykonanie wymiernego stwierdzenia odnośnie statusu projektu i jakości.

Ważne jest, by pamiętać, iż rezultaty pomiaru (liczby zebrane podczas pomiaru) muszą być zawsze porównane z danymi odniesienia (odpowiednimi metrykami).

### 9.4.2 Metryki dla wymagań (K1)

Dla wymagań można zastosować następujące metryki:

- Koszty projektu
  - Liczba wymagań
- Śledzenie w projekcie
  - Liczba wymagań śledzonych do innych artefaktów
- Stabilność projektu
  - Liczba zmienionych wymagań
- Doskonalenie procesu
  - Powody zmian w wymaganiach (defekty, ulepszenia)
- Jakość specyfikacji
  - Pokrycie wymagań modelami
- Liczba defektów
  - Typ defektu
  - Liczba błędów w wymaganiach o różnych typach (defekty logiczne, spójności, danych itp.)

### 9.4.3 Pomiar jakości wymagań (K2)

Ocenę jakości wymagań umożliwiają następujące pytania:

- Czy wymagania są prawidłowe?
- Czy wymagania są zrozumiałe?
- Czy wymagania są wykonalne?
- Czy wymagania można śledzić?
- Czy wymagania są identyfikowalne?
- Czy wymagania są testowalne?

Niektóre formuły, które można użyć do pomiaru jakości wymagań:

$$\text{Jasność} = \frac{\sum \text{Wymagań bez zgłoszonych defektów i problemów}}{\sum \text{Wymagań}}$$

Miarą jakości wymagań może być także wskaźnik zmian (nie dotyczy projektów zwinnych). Im wyższa wartość wskaźnika zmian całego zbioru wymagań (co może być spowodowane przez wysiłki zmierzające do wyjaśnienia niezrozumiałych wymagań i/lub niepełny lub niespójny opis wymagań), tym bardziej projekt jest zagrożony. Dlatego w celu zarządzaniem ryzykiem w projekcie powinien być mierzony wskaźnik zmian.

<b>10 Narzędzia (K2)</b>	<b>40 minut</b>
--------------------------	-----------------

*Cele nauczania dla Poziomu Podstawowego Inżynierii Wymagań*

Cele określają wiedzę i umiejętności, uzyskiwane po ukończeniu każdego modułu.

**10.1 Korzyści z Narzędzi (K2)**

- LO-10.1.1 Wyjaśnić cel wsparcia narzędziowego w Inżynierii Wymagań (K2)
- LO-10.1.2 Opisać, jakie czynności w Inżynierii Wymagań mogą być wspierane przez narzędzia (K2)

**10.2 Kategorie Narzędzi (K2)**

- LO-10.2.1 Opisać wymagania dotyczące narzędzi w obszarze Inżynierii Wymagań (K2)
- LO-10.2.2 Wyjaśnić, co należy wziąć pod uwagę w kontekście kosztu narzędzi (K2)

<b>10.1</b>	<b>Korzyści Narzędzi (K2)</b>	<b>20 minut</b>
-------------	-------------------------------	-----------------

## Pojęcia

Narzędzia Inżynierii Wymagań

### 10.1.1 Zastosowanie narzędzi w Inżynierii Wymagań (K2)

Narzędzia do przechowywania i administracji wymagań ułatwiają Inżynierię Wymagań. Mogą one przejmować powtarzalne mechaniczne czynności lub zapewniać przegląd informacji. Możliwe jest zatem utrzymywanie trudnej dokumentacji zgodnej i aktualnej. Wybór narzędzia musi nastąpić przed wytwarzaniem produktu. W przeciwnym wypadku może to spowodować poważne problemy.

Narzędzia mogą wspierać następujące czynności w Inżynierii Wymagań:

- Identyfikację oraz przechowywanie wymagań
- Modelowanie wymagań (wraz z prototypowaniem)
- Dokumentowanie wymagań (tworzenie specyfikacji wymagań)
- Definiowanie i utrzymywanie śledzenia wymagań

### 10.1.2 Korzyści z używania narzędzi (K2)

- Zapewnienie, że wszystkie wymagania są przechowywane w jednym miejscu i dostępne dla wszystkich interesariuszy
- Wspieranie śledzenia wymagań (do przypadków testowych itp.) oraz umożliwienie weryfikacji odpowiedniego pokrycia wymagań
- Umożliwienie zarządzania wymaganiami zmian w łatwy sposób
- Poprawa jakości specyfikacji wymagań poprzez wymuszanie stosowania określonych szablonów dokumentów i notacji modelowania
- Oszczędność czasu dzięki automatyzacji niektórych działań (takich jak generowanie pełnych specyfikacji z narzędzia)

<b>10.2</b>	<b>Kategorie Narzędzi (K2)</b>	<b>20 minut</b>
-------------	--------------------------------	-----------------

## Pojęcia

Kategorie narzędzi

### 10.2.1 Kategorie Narzędzi (K2)

- Narzędzia do pozyskiwania wymagań
  - Mapy myśli
- Narzędzia do modelowania
  - Narzędzia UML
  - Narzędzia SysML
- Narzędzia do prototypowania
- Narzędzia do zarządzania wymaganiami
- Narzędzia do zarządzania defektami
- Narzędzia do zarządzania zmianami
- Narzędzia do zarządzania projektem

Koszt zakupu narzędzi jest bardzo zróżnicowany. Komercyjne narzędzia mogą być bardzo drogie, podczas gdy narzędzia open source są bezpłatne. Wybór narzędzia musi być zatem dokonany bardzo ostrożnie. Przed wybraniem narzędzia, powinna być prowadzona analiza. Analiza powinna brać pod uwagę:

- Jaka notacja do modelowania jest używana w organizacji i ma być wspierana przez narzędzie
- Czy organizacja planuje korzystać z innych notacji modelowania w przyszłości (w tym przypadku może być uzasadniony zakup narzędzia wspierającego planowaną notację razem z tymi już jest stosowanymi w organizacji)
- Wymagania organizacji dotyczące funkcjonalności wymaganych od narzędzia (zazwyczaj komercyjne narzędzia zapewniają bardziej złożone funkcje, niż open source)
- Koszty narzędzi biorąc pod uwagę, czy narzędzie ma być używane tylko do jednego konkretnego projektu czy będzie wykorzystywane we wszystkich/większości projektów

- Czy narzędzie może być zintegrowane z innymi niezbędnymi narzędziami (np. narzędziem do zarządzania defektami, narzędziem do zarządzania projektami, czy innymi - w zależności od potrzeb organizacji)
- Czy narzędzie umożliwia wymianę informacji z narzędziem wykorzystywanym przez organizację klienta (w niektórych przypadkach klient wykonuje własną analizę wymagań, a produkty szczegółowej analizy wymagań dostawcy powinny być następnie przeniesione do środowiska klienta)
- Łatwość obsługi i uczenia się (potencjalny koszt szkolenia), dostępność pomocy online, podręczników, samouczków, innego wsparcia

Szybki wybór może powodować wysokie koszty (K2):

- Koszt zakupu narzędzia, które nie spełnia wymagań użytkownika i nie spełnia swojego celu
- Koszt zakupu drogiego narzędzia, które jest używane tylko w jednym projekcie lub zakupu drogiego narzędzia, podczas gdy istnieją narzędzia open source o podobnej funkcjonalności
- Koszt szkolenia w przypadku zakupu narzędzia bez wystarczającego systemu pomocy (zwłaszcza w przypadku, gdy od narzędzia wymagane są tylko podstawowe funkcje)
- Koszt rozszerzenia zakupionego narzędzia o dodatkowe, wymagane przez użytkowników funkcje, nieobsługiwane przez narzędzie (gdy były też inne narzędzia o takiej funkcjonalności)
- Koszty integracji narzędzia z innymi narzędziami stosowanymi w organizacji

## 11 Literatura

- Beck, K.: *Extreme Programming*. Munich 2003
- Beck, K.: *Extreme Programming Explained: Embrace Change*. Boston 2000
- Beck, K.: *Test Driven Development. By Example*. Amsterdam 2002
- Beck, K.: *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Longman 1999
- Boehm, B.: *Software Engineering Economics*. Englewoods Cliffs, NJ 1981
- Bohner, S.A. and R.S. Arnold, Eds. *Software Change Impact Analysis*. Los Alamitos, California, USA, IEEE Computer Society Press 1996.
- Bundschuh, M.; Fabry, A.: *Aufwandschätzung von IT-Projekten*. Bonn 2004
- Cockburn, A.: *Agile Software Development*. Addison Wesley 2002
- Cockburn, A.: *Writing Effective Use Cases*. Amsterdam 2000
- Cohn M.: *Estimating With Use Case Points*, Fall 2005 issue of Methods & Tools
- Cotterell, M. and Hughes, B.: *Software Project Management*, International Thomson Publishing 1995
- Newman, W.M. and Lamming, M.G.: *Interactive System Design*, Harlow: Addison-Wesley 1995
- Davis A. M.: *Operational Prototyping: A new Development Approach*. IEEE Software, September 1992. Page 71
- Davis, A. M.: *Just Enough Requirements Management. Where Software Development Meets Marketing*, Dorset House, 2005, ISBN 0932633641
- DeMarco, T. et al.: *Adrenalin-Junkies und Formular-Zombies – Typisches Verhalten in Projekten*. Munich 2007
- DeMarco, T.: *Controlling Software Projects: Management, Measurement and Estimates*. Prentice Hall 1986
- DeMarco, Tom: *The Deadline: A Novel About Project Management*. New York 1997
- Dorfman, M. S.: *Introduction to Risk Management and Insurance (9 ed.)*. Englewood Cliffs, N.J: Prentice Hall 2007. ISBN 0-13-224227-3.
- Dynamic Systems Development Method Consortium. See: <http://na.dsdm.org>
- Ebert, Ch.: *Systematisches Requirements Management. Anforderungen ermitteln, spezifizieren, analysieren und verfolgen*. Heidelberg 2005
- Evans, E. J.: *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Amsterdam 2003
- Graham, D. et al: *Foundations of Software Testing*. London 2007
- Gilb, T.; Graham, D.: *Software Inspection*. Reading, MA 1993
- Gilb, T.: *What's Wrong with Requirements Specification*. See: [www.gilb.com](http://www.gilb.com)

Heumann, J.: *The Five Levels of Requirements Management Maturity*, see: [http://www.ibm.com/developerworks/rational/library/content/RationalEdge/feb03/ManagementMaturity\\_TheRationalEdge\\_Feb2003.pdf](http://www.ibm.com/developerworks/rational/library/content/RationalEdge/feb03/ManagementMaturity_TheRationalEdge_Feb2003.pdf)

Linstone H. A., Turoff M.: *The Delphi Method: Techniques and Applications*, Reading, Mass.: Addison-Wesley, ISBN 9780201042948, 1975

Hull, E. et. All: *Requirements Engineering*. Oxford 2005

IEEE Standard 610.12-1990 IEEE Standard Glossary of Software Engineering Terminology

IEEE Standard 829-1998 IEEE Standard for Software Test Documentation

IEEE Standard 830-1998 IEEE Recommended Practice for Software Requirements Specifications

IEEE Standard 1012-2004: IEEE Standard for Software Verification and Validation

IEEE Standard 1059-1993: IEEE guide for software verification and validation plans

IEEE Standard 1220-1998: IEEE Standard for Application and Management of Systems Engineering Process

IEEE Standard 1233-1998 IEEE Guide for Developing System Requirements Specifications

IEEE Standard 1362-1998 IEEE Guide for Information Technology-System Definition – Concept of Operations (ConOps) Document

ISO 9000

ISO/EIC 25000

ISO 12207

ISO 15288

ISO 15504

ISO 31000: Risk Management - Principles and Guidelines on Implementation

IEC 31010: Risk Management - Risk Assessment Techniques

ISO/IEC 73: Risk Management – Vocabulary

ISTQB: ISTQB Glossary of Testing Terms 2.1

ISTQB: Certified Tester, Foundation Level Syllabus, version 2011

Jacobsen, I. et al.: *The Unified Software Development Process*. Reading 1999

Jacobson, I. et al.: *Object-Oriented Software Engineering. A Use Case Driven Approach*. Addison-Wesley 1993

Kilpinen, M.S.: *The Emergence of Change at the Systems Engineering and Software Design Interface: An Investigation of Impact Analysis. PhD Thesis*. University of Cambridge. Cambridge, UK 2008.

Lauesen, S.: *Software Requirements: Styles and Techniques*. London 2002

Mangold, P.: *IT-Projektmanagement kompakt*. Munich 2004

- McConnell, S.: *Aufwandschätzung für Softwareprojekte*. Unterschleißheim 2006
- McConnell, S.: *Rapid Development: Taming Wild Software Schedules (1st ed.)*. Redmond, WA: Microsoft Press. ISBN 1-55615-900-5, 1996
- Paulk, M., et al: *The Capability Maturity Model: Guidelines for Improving the Software Process*. Reading, MA 1995
- Pfleeger, S. L.: *Software Engineering: Theory and Practice, 2nd edition*. Englewood Cliffs, NJ 2001
- Pfleeger, S.L. and J.M. Atlee: *Software Engineering: Theory and Practice*. Upper Saddle River, New Jersey, USA, Prentice Hall 2006.
- Pohl, K.: *Requirements Engineering. Grundlagen, Prinzipien, Techniken*. Heidelberg 2007
- Project Management Institute: *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*. PMI 2004
- Putnam, L.e H.; Ware M. *Measures for excellence: reliable software On time, within budget*. Yourdon Press. ISBN 0-135676-94-0, 1991
- Robertson, S.; Robertson, J.: *Mastering the Requirements Process*, Harlow 1999
- Rupp, C.: *Requirements-Engineering und Management. Professionelle, Iterative Anforderungsanalyse in der Praxis*. Munich 2007
- Sharp H., Finkelstein A. and Galal G.: *Stakeholder Identification in the Requirements Engineering Process*, 1999
- Sommerville, I.: *Requirements Engineering*. West Sussex 2004
- Sommerville, I.: *Software Engineering 8*. Harlow 2007
- Sommerville, I.; Sawyer, P.: *Requirements Engineering: A Good Practice Guide*. Chichester 1997
- Sommerville, I.; Kotonya, G.: *Requirements Engineering: Processes and Techniques*. Chichester 1998
- Spillner, A. et all: *Software Testing Foundations*. Santa Barbara, CA 2007
- SWEBOK - The Guide to the Software Engineering Body of Knowledge:  
<http://www.computer.org/portal/web/swebok/home>
- Thayer, R. H.; Dorfman, M.: *Software Requirements Engineering, 2nd edition*. Los Alamitos, CA 1997
- V-Modell® XT: <http://www.vmodellxt.de/>
- Wieggers, K.E.: *First Things First: Prioritizing Requirements*. Software Development, September 1999
- Wieggers, K. E.: *Software Requirements*. Redmond 2005
- Wieggers, K. E.: *More About Software Requirements: Thorny Issues and Practical Advice*. Redmond, Washington 2006
- Young, R. R.: *Effective Requirements Practices*. Addison-Wesley 2001

## 12 Index

- Akceptacja wymagań, 64, 84
- Analiza Obiektowa, 68, 74
- Analiza Trybów i Efektów Awarii, 34, 36
- Analiza wymagań, 29, 64, 82
- Burza mózgów, 49, 52
- Cel, 9, 46, 49, 59, 69
- CMMI, 28
- CoCoMo, 79, 81
- Customer, 43
- Cykl życia produktu, 23
- Dekompozycja funkcjonalna, 70, 72, 73
- Diagram aktywności, 74
- Diagram behawioralny, 74
- Diagram klas, 74
- Diagram komponentów, 74
- Diagram komunikacji, 74
- Diagram maszyny stanów, 74
- Diagram obiektów, 74
- Diagram pakietów, 74
- Diagram przypadków użycia, 74
- Diagram sekwencji, 74
- Diagram strukturalny, 74
- Diagram wdrożenia, 74
- Diagram wymagań, 74
- Dokument wymagań, 59
- Failure Mode and Effects Analysis, 36
- FMEA, 36, 37
- Formalizacja, 60, 65
- Historyjki Użytkownika, 62
- Identifying Requirements, 49
- Identyfikacja wymagań, 29, 51, 54, 57, 82
- IEC 31010, 106
- IEEE 1233, 19, 63
- IEEE 1362, 19, 63
- IEEE 610, 12, 19
- IEEE 830, 19, 59, 61, 63
- Impact Analysis, 105, 106
- Interesariusz, 39, 40, 48
- Inżynieria Wymagań, 3, 11, 12, 17, 20, 29, 31, 32, 39, 44, 93, 96
- ISO 12207, 20, 106
- ISO 15288, 20, 106
- ISO 15504, 20, 28, 106
- ISO 31000, 34, 106
- ISO 9000, 19, 106
- ISO 9126, 19
- ISO/EIC 25000, 106
- ISO/IEC 15504, 27, 28
- ISO/IEC 25000, 19, 55, 56
- ISO/IEC 73, 106
- Kierownik Wymagań, 39
- Klient, 28, 39, 40, 44, 53, 90
- Komitet Kontroli Zmian, 89, 90
- Konkluzja przez analogię, 78
- Kontrakt, 44
- Kontraktor, 39
- Kryteria akceptacji, 44, 61
- Kryteria Akceptacji, 96
- Krytyczność, 12, 17
- Kwestionariusz, 49, 50

Metoda Delphi, 78  
 Metryka, 99  
 Model konceptualny, 69, 71  
 Model kontekstowe, 70  
 Model przepływu danych, 70, 72, 73  
 Model rozwiązania, 64, 69, 71  
 Model V, 23  
 Model warunków, 70  
 Model wymagań, 69, 71  
 Model związków encji, 70, 73  
 Możliwość śledzenia, 25, 66, 87  
 Narzędzia, 101, 102, 103  
 Object-oriented Analysis, 74  
 Obserwacja polowa, 49  
 Plan Zarządzania Ryzykiem, 34, 36  
 Podpisywanie wymagań, 84  
 Priorytet, 12, 16  
 Priorytetyzacja, 68, 82  
 Procedura algorytmiczna, 79  
 Process Models, 23  
 Programowanie Ekstremalne, 23, 25  
 Project Management, 31, 105, 107  
 Punkty Funkcyjne, 79  
 Reprezentant klienta, 49, 51  
 Requirements Engineering, 22, 31, 35, 38, 106, 107  
 Risk, 31, 34, 35, 36, 105, 106  
 Risk Assessment, 36, 106  
 Risk Management, 31, 34, 35, 36, 105, 106  
 Równanie Putnama, 79  
 Rozwiązanie, 12, 15, 69  
 Ryzyko, 34, 35, 53  
 Ryzyko Produktowe, 34  
 Ryzyko Projektowe, 34  
 Samo-rejestracja, 49, 50  
 Skale priorytetowe, 83  
 Śledzenie horyzontalne, 87, 88  
 Śledzenie wertykalne, 88  
 Śledzenie Wymagań, 86  
 Software Requirements Specifications, 106  
 Specyfikacja, 13, 15, 24, 29, 52, 60, 61, 62, 63, 64, 65  
 Specyfikacja rozwiązania, 61  
 Specyfikacja Rozwiązania, 62  
 Specyfikacja wymagań, 15, 29, 52, 61, 65  
 Specyfikacja Wymagań, 60, 61, 63  
 SysML, 65, 68, 74, 76, 103  
 Szacowanie Kosztów, 77  
 Szacowanie Kosztu, 68  
 Tabela decyzyjna, 70  
 Terminowanie, 49, 53  
 Testowalność, 93, 96  
 UML, 65, 68, 74, 76, 103  
 Uzgadnianie wymagań, 29, 42  
 Verification and Validation, 106  
 Walidacja, 12, 17, 29, 66  
 Warsztaty, 49, 53, 54  
 Weryfikacja, 17, 66, 95  
 Wizja, 43, 46  
 Wymagania funkcjonalne, 13, 55, 56, 71  
 Wymagania нефunkcjonalne, 13, 55, 56, 71  
 Wymagania Procesowe, 12  
 Wymagania Produktowe, 12  
 Wymaganie, 11, 12, 66, 83

Wymaganie Funkcjonalne, 12

Wymaganie Niefunkcjonalne, 12

Wytwarzanie Wymagań, 17

Wytwórca Wymagań, 39

Wywiad, 49, 50

Żądanie Zmiany, 89, 90

Zapewnienie Jakości, 93, 96

Zarządzanie Projektem, 31, 32

Zarządzanie Ryzykiem, 31, 34, 35, 36

Zarządzanie Wymaganiami, 12, 17, 18

Zarządzanie Zmianą, 32, 86, 89

Zmiana, 89, 90, 91

Zmiany wymagań, 29, 89

Zobowiązanie, 12, 15