**I. Podstawy Testowania**

**Ogólnie 1.1 Co To Jest Testowanie:**

**Powszechnie uważa się, że testowanie polega wyłącznie na wykonywaniu testów, czyli uruchamianiu oprogramowania i sprawdzaniu uzyskanych rezultatów**

Testowanie oprogramowania to proces obejmujący czynności które wykraczają poza samo wykonywanie testów.

W skład procesu testowania wchodzą takie czynności jak:

-planowanie,

-analiza,

-monitorowanie testów i nadzór nad testami,

-projektowanie i implementacja testów,

-raportowanie o postępie i wynikach testów oraz dokonywanie jakości przedmiotu testów

Testowanie może wymagać uruchomienia modułu lub systemu – testowanie dynamiczne

Testowanie bez uruchamiania testowego obiektu – testowanie statyczne

**K1 1.1.1  Typowe Cele Testowania:**

-Zapobieganie defektom poprzez ocen produktów pracy takich jak: wymagania, historyjki użytkownika, projekt i kod;

-Weryfikacja czy zostały spełnione wszystkie wyspecyfikowane wymagania

-sprawdzenie czy przedmiot testów jest kompletny

 -walidowanie czy działa zgodnie z oczekiwaniami użytkowników i innych klientów

- budowanie zaufania do poziomu jakości przedmiotu testów

-wykrywanie defektów i awarii a tym samym zmniejszenie ryzyka związanego z niedostateczną jakością oprogramowania.

-dostarczanie klientom niezbędnych informacji do podejmowania decyzji

-przestrzeganie wymagań wynikających z umów, przepisów prawa

Cele testowania mogą się różnić w zależności od kontekstu testowanego modułu lub systemu - Różnice:

-W przypadku testowania modułowego jednym z celów może być wykrycie jak największej liczby awarii

W przypadku testowania akceptacyjnego jednym z celów może być potwierdzenie że system działa zgodnie z oczekiwaniami i spełnia wymagania. Innym celem może być dostarczenie klientom informacji na temat ryzyk

**K2 1.1.2. Testowanie a Debugowanie:**

Testowanie i debugowanie to dwie różne czynności. Wykonywanie testów pozwala ujawnić awarie, które są skutkiem defektów w oprogramowaniu natomiast debugowanie to czynności związane z wytwarzaniem oprogramowania która polega na znajdowaniu analizowaniu i usuwaniu tych defektów.

Następnie wykonywane jest **testowanie potwierdzające,** które pozwala sprawdzić, czy wprowadzone poprawki w rezultacie spowodowały usunięcie defektów.

Standard międzynarodowy ISO/ICE/IEEE 29119-1 zawiera bardziej szczegółowe informacje dotyczące pojęć związanych z testowaniem.

**Ogólnie 1.2. Dlaczego Testowanie Jest Niezbędne:**

Rygorystyczne testowanie modułów i systemów oraz związanej z nimi dokumentacji

-może pomóc w zmniejszaniu ryzyka wystąpienia awarii podczas eksploatacji oprogramowania.

-Wykrycie a następnie usunięcie defektów przyczynia się do podniesienia jakości modułów lub systemu.

-Testowanie oprogramowania może być niezbędne do spełniania wymagań wynikających z umów, przepisów prawa bądź norm czy standardów.

**K2 1.2.1. Znaczenie Testowania Dla Powodzenia Projektu:**

W historii informatyki znamy wiele przykładów gdy systemy zostały przekazane do eksploatacji lecz na skutek defektów uległy awarii lub z innych powodów nie zaspokoiły potrzeb interesariuszy.

**Dzięki odpowiednim technikom testowania częstotliwość występowania problemów można ograniczyć.**

Kilka przykładów zastosowania odpowiednich technik:

-Zidentyfikowanie i usunięcie defektów zmniejsza ryzyko wytworzenia niepoprawnych cech oprogramowania.

-Ścisłą współpraca między testerami a projektantami systemu na etapie prac projektowych pozwala lepiej zrozumieć obu stronom projekt systemu i sposób jego testowania.

-Ścisłą współpraca testerów  z programistami pozwala obu stronom lepiej zrozumieć kod i sposób jego testowania.

-Weryfikacja i walidacja oprogramowania przez testerów przed przekazaniem go do użytkowania pozwala wykryć defekty prowadzące do awarii.

Osiągnięcie postawionych celów testowania przyczynia się do ogólnego powodzenia procesu wytwarzania i pielęgnacji oprogramowania.

**K2. 1.2.2. Zapewnianie Jakości a Testowanie:**

Testowanie jest często utożsamiane z zapewnieniem jakości ale w rzeczywistości są to dwa oddzielne procesy które nazywają się zarządzaniem jakością.

**Zarządzanie jakością obejmuje wszystkie czynności nadzorowania działań** organizacji w dziedzinie jakości.

Elementami zarządzania jakością są między innymi zapewnianie jakości i kontrola jakości,

**Zapewnianie jakości skupia się na na realizowaniu właściwych procesów w celu uzyskania odpowiedniego poziomu jakości.**

**Duze znaczenie dla skutecznego zapewniania jakości ma wykorzystanie procesu analizy do wykrywania i usuwania przyczyn defektów.**

Kontrola jakości obejmuje szereg czynności włączając w to czynności testowe k

tóre wspierają osiągnięcie odpowiednich poziomów jakości w procesie wytwarzania i pielęgnacji oprogramowania.

**Działania związane z zapewnieniem jakości wspierają właściwe testowanie.**

**K2 1.2.3. Pomyłki, defekty i awarie:**

Awaria(failure): Odchylenie od spodziewanego zachowania albo wyniki działania oprogramowania.

Defekt, usterka, pluskwa(bug, falut): skutek błędu twórcy oprogramowania . Usterka może ale nie musi spowodować awarii. Defekt który wystąpi podczas uruchamiania systemu może spowodować awarie modułu lub systemu.

Błąd, Pomyłka(error): działanie człowieka powodujące powstawanie nieprawidłowego wyniku

 Przykład: nieprawidłowa wypłata odsetek jest awarią a nieprawidłowe obliczenia wykonywane przez kod są defektem wynikającym z pierwotnego defektu.

**K1 1.3. Siedem Zasad Testowania:**

1. Testowanie ujawnia usterki ale nie może dowieść ich braku
2. Testowanie gruntowne nie jest możliwe
3. Wczesne Testowanie oszczędza czas i pieniądze
4. Kumulowanie się defektów (zwykle większość defektów wykrytych podczas testowania ma swoje źródło w niewielkiej liczbie modułów)
5. Paradoks Pestycydów (ciągłe powtarzanie tych samych testów przestaje w pewnym momencie wykrywać nowe defekty, niezmienne testy tracą z czasem zdolność do wykrywania defektów, w niektórych przypadkach takich jak testowanie automatyczne paradoks pestycydów może być korzystny ponieważ pozwala potwierdzić że liczba defektów związanych z regresja jest niewielka)
6. Testowanie zależy od kontekstu (testowanie różni się w zależności od tego co testujemy np. systemy przemysłowe testuje się inaczej niż aplikacje mobilne sklepu internetowego)
7. Przekonanie o braku błędów jest błędem (niektóre organizacje oczekują iż testerzy będą w stanie uruchomić wszystkie możliwe testy i wykryć wszystkie defekty lecz jest to niemożliwe, co więcej błędnym przekonaniem jest iż znalezienie i naprawienie defektów uwalnia system od błędów)

**Ogólnie 1.4 Proces Testowy:**

Nie ma uniwersalnego procesu testowania oprogramowania ale istnieją typowe czynności testowe. Dóbr odpowiedniej strategii testowania zależy od wielu czynników. Organizacji w swojej strategii testowej może zdefiniować które czynności wchodzą w skład procesu testowania.

**K2. 1.4.1. Proces Testowy w Kontekście:**

**Przykładowe czynniki kontekstowe** wpływające na proces testowy w organizacji to :

-wykorzystanie modelu cykli życia oprogramowania i

-metodyki projektowe

-poziomy testów i typy testów

-czynniki ryzyka produktowego i projektowego

-dziedzina biznesowa

-ograniczenia operacyjne takie jak:

budżet i zasoby

harmonogramy

złożonośc procesu lub projektu

wymagania wynikające z umów i przepisów

-polityka testów i praktyk w organizacji

-wymagane standardy normy i standardy organizacji i poza nią

Dobrą praktyką jest zdefiniowanie mierzalnych kryteriów pokrycia dotyczące podstaw testów

Przykładem ilustrującym tę praktykę jest podstawa testów aplikacji mobilnej która może zawierać listę wymagań oraz listę wspieranych urządzeń mobilnych.

**K2. 1.4.2. Czynności i Zadania Testowe:**

-planowanie testów

-monitorowanie testów i nadzór nad testami

-analiza testów

-projektowanie testów

-implementacja testów

-wykonywanie testów

-ukończenie testów

Planowanie Testów: Obejmuje czynności których efektem jest zdefiniowanie celów testowania oraz podejścia do osiągania celów testowania.

Monitorowanie Testów i nadzór nad Testami:

Monitorowanie testów polega na ciągłym porównywaniu rzeczywistego z zaplanowanym postępem testowania.

Nadzór nad testami polega na podejmowaniu działań które są niezbędne do osiągania celów wyznaczonych w planie testów.

Interesariusze są informowani o postępie w realizacji planu testów za pomocą raportów które zawierają informacje o ewentualnych odchyleniach od planu oraz informacje pomagające podjęcie decyzji o wstrzymaniu testowania.

Analiza testów:

Analiza testów służy do ustalenia tego co należy przetestować

Główne czynności wykonywane w ramach analizy testów:

-dokonywanie analizy podstawy testów

-specyfikacja wymagań

-informacji dotyczących projektu i implementacji

-informacji dotyczących projektu i implementacji

-implementacji samego modułu lub systemu

-raportów z analizy ryzyka które mogą dotyczyć aspektów funkcjonalnych i niefunkcjonalnych

-dokonywanie oceny testowalności podstawy  testów i elementów testowych

-identyfikowanie cech i zbiorów cech które mają zostać przetestowane

-definiowanie warunków testowych

Zastosowanie technik czrnoskrzynkowych, białoskrzynkowych oraz technik opartych na doświadczeniu w ramach analizy testów pomaga zmniejszyć prawdopodobieństwo pominięcia warunków tetowych.

W niektórych przypadkach w wyniku analizy testów definiowane są warunki testowe.

Identyfikowanie defektów na etapie analizy testów jest istotną potencjalną korzyscia .

Czynności wykonywane w ramach analizy testów pozwalają zweryfikować czy wymagania są spójne i czy odzwierciedlają wymagania klientów

 BEHAVIIOR DRIVEN DEVELOPMENT BDD – Wytwarzanie sterowane zachowaniem i ACCEPTANCE TEST DRIVEN DEVELOPMENT ATDD wytwarzanie sterowane testami akceptacyjnymi które obejmuje generowanie warunków testowych i przypadków testowych na podstawie historyjek użytkownika i kryteriów akceptacyjnych przed rozpoczęciem tworzenia kodu.

Projektowanie Testów:

O ile analiza testów odpowiada na pytanie co należy przetestować o tyle projektowanie testów odpowiada na pytanie jak należy testować.

Podczas projektowania testów warunki testowe są przekształcane w przypadki testowe wysokiego poziomu.

Główne czynności wykonywane w ramach projektowania testów to:

-projektowanie przypadków testowych oraz określenie ich priorytetów

-identyfikacja danych testowych

-projektowanie środowiska testowego

Implementacja Testów:

Implementacja testów odpowiada na pytanie czy mamy wszystko co jest potrzebne do uruchomienia testów.

Podczas implementacji testów tworzy się i lub kończy testalia niezbędne do wykonania testów.

Główne czynności:

Opracowanie procedur testowych

Utworzenie zestawów testowych

Uporządkowanie zestawów testowych w harmonogram wykonywania testów

Zbudowanie środowiska testowego

Przygotowanie danych testowych

Wykonywanie Testów:

Podczas wykonywania testów uruchamiane są zestawy testowe zgodnie z harmonogramem testów.

Główne czynności przeprowadzane w ramach wykonywania testów to :

-zarejestrowanie danych identyfikacyjnych

-wykonywanie testów ręcznie lub przy użyciu narzędzi do wykonywania testów

-porównanie rzeczywistych wyników  z wynikami oczekiwanymi

-raportowanie defektów

-powtórzenie czynności testowych

-zweryfikowanie i zaktualizowanie możliwości dwukierunkowego  śledzenia powiązań między podstawą testów warunkami testów przypadkami testowymi procedurami testowymi a wynikiem

Ukończenie Testów:

Polega na zebraniu danych pochodzących z wykonywanych czynności testowych. Czynności związane z ukończeniem testów są wykonywane w momencie osiągnięcia kamieni milowych projektów takich jak: przekazanie systemu lub oprogramowania, zakończenie realizacji projektu, ukończenie testów danego poziomu.

Główne czynności wykonywane w ramach ukończenia testów to:

-sprawdzenie czy wszystkie raporty o defektach są zamknięte

-utworznie raportu z testów

-dla wersji końcowych zarchiwizowanie środowiska testowego, danych testowych

-przekazanie testaliów

-przeanalizowanie wniosków z ukończonych czynności testowych

-wykorzystanie zebranych informacji do zwiększenia dojrzałości procesu testowego.

**K2 1.4.3 Produkty Pracy Związane z Testowaniem:**

Produkty pracy związane z testowaniem powstają w ramach procesu testowego.

Punktem odniesienia dla produktów pracy związanych z testowaniem może być również standard międzynarodowy ISO/IEC/IEEE 29119-3

Produkt Pracy Planowania Testów:

Wśród pracy powstających na etapie planowania testów można zwykle wyróżniać jeden lub kilka planów testów. Plan testów zawiera informacje na temat podstawy testów. Ponadto określa się w nim kryteria wyjścia które będą stosowane w ramach monitorowania testów i nadzoru nad testami.

Produkty pracy monitorowania testów i nadzoru nad Testami:

Są różnego rodzaju raporty testów w tym raporty o postępie testów oraz sumaryczne raporty testów.

Raporty z testów powinny zawierać szczegółowe informacje na temat dotychczasowego przebiegu procesu testowego i powinny być przedstawiane w formie dostosowanej do potrzeb konkretnych odniorców.

Produkty Pracy Analizy Testów:

Produkty związane z analizą testow obejmują zdefiniowane i uszeregowane według priorytetów warunki testowe przy czym pożdana jest dwukierunkowa możliwość sledzenia powiązan miedzy tymi warunkami

Produkty Pracy Projektowania Testów:

W wyniku projektowania testów w oparciu o zdefiniowane w analizie testów warunki testowe powstją przypadki testowe i zbiory przypadków testoweych

W idealej sytuacji dla każdego przypadku testowego istnieje możlowść dwukierunkowego śledzenia powiązań między tymi przypadkami a pokrywanym przez niego warunkiem testowym.

Wśród rezultatów etapu projektowania testów można również wymienić:

-zaprojektowanie i zidentyfikowanie niezbędnych danych testowych

-zaprojektowanie środowiska testowego

-zidentyfikowanie infrastruktury i narzędzi

Produkt Pracy Implementacji testów:

Produkty pracy związane z implementacją testów to między innymi:

-procedury testowe oraz kolejnośc ich wykoanania

-zestawy testowe

-harmonogram wykonywania testów

W idealnym przypadku po zakończeniu implementacji testów można wykazac spełnienei pokrycia określonych w planie testów

Implementacja testów może także skutkować utworzniem i zweryfikowaniem danych testowych i środowiska testowego.

**Produkty pracy wykonywania testów**

 Do produktów pracy związanych z wykonywaniem testów zaliczają się między innymi:

• dokumentacja dotycząca statusu poszczególnych przypadków testowych lub procedur testowych (np.: gotowy do wykonania, zaliczony, niezaliczony, zablokowany, celowo pominięty itd.);

 • raporty o defektach (patrz podrozdział 5.6.); • dokumentacja wskazująca, które elementy testowe, przedmioty testów, narzędzia testowe i testalia zostały wykorzystane w ramach testowania.

Produkt Pracy Ukończenia Testów:

Produkty pracy związane z ukończeniem testów to między innymi sumaryczne raporty z testów lisat czynności do wykonania, żądania zmian, pozycje  wbacklogu, ostateczna wersja testaliów

**K2. 1.4.4. Śledzenie Powiązań Między Podstawą testów a Produktami Pracy Związanymi z Testowaniem.**

Produkty pracy związane z testowaniem i ich nazwy mogą być bardzo różne. Niezależnie od tych róznic w celu zapewniania skutecznego monitorowania i nadzoru istotne jest stworzenie i utrzymanie mechanizmu śledzenia powiązań między każdym elementem podstawy testów.

Sprawne śledzenie powiązań umożliwia nie tylko ocenę pokrycia testowego ale również:

-analizowanie wpływu zmian

-przeprowadzanie audytu zmian

-przeprowadzanie audytu testów

-spełnianie kryteriów związanych z zarządzaniem w obszarze IT

-tworzenie bardziej zrozumiałych raportów o statusie testów i sumarycznych raportów z testów

-przekazanie interesariuszom informacji o aspektach technicznych testowania

-udzielanie informacji potrzebnych do oceny jakości produktów

Niektóre narzędzia do zarządzania testami mają wbudowane modele produktów pracy związanej z testowaniem.

**Ogólnie 1.5. Psychologia Testowania:**

Wytwarzanie oprogramowania w tym testowanie to proces realizowany z udziałem ludzi w związku z czym duże znaczenie dla przebiegu testowania mają uwarunkowania psychologiczne.

**K1 1.5.1 Psychologia Człowieka a Testowanie:**

Identyfikowanie defektów podczas testowania statycznego bądź identyfikowanie awarii w trakcie testowania dynamicznego może być odbierane jako krytyka produktu lub jego autora.

Przykładem takiego zjawiska są oczekiwania programistów że ich kod będzie poprawny co prowadzi do tego iż ciężko jest im zaakceptować że ich kod jest błędny.

Dodatkowy czynnik utrudniający przyjęcie informacji zwrotnej z testów stanowi skłonność wielu osób do obwiniania osoby przynoszącej złe wiadomości a takie sytuacje są często rezultatem testowania.

Niektóre osoby mogą postrzegać testowanie jako czynność destrukcyjną nawet jeśli przyczynia się ona    do postępu w realizacji projektu i podnoszenia jakości produktów.

Aby ograniczyć podobne reakcje należy przekazywać informacje w sposób jak najbardziej konstruktywny.

ZASADy konfliktu DOBREJ  WYMIANY INFORMACJI:

-Należy zacząć od współpracy nie od kłótni

-Należy podkreślić korzyści wynikające z testowania. Dla organizacji wykrycie i usunięcie defektów podczas testowania oznacza oszczędność czasu i pieniędzy oraz zmniejszenie ryzyka związanego z jakością produktów

-informacje na temat rezultatów testów i inne wnioski należy przekazywać w sposób neutralny koncentrując się na faktach i nie krytykując deweloperów

-należy wczuć się w sytuacje drugiej osoby i zrozumieć dlaczego negatywnie ona reaguje na podane informacje

-należy upewnić się ze rozmówca rozumie przekazane informacje

Należy zadbać by testerzy przeszczegali przyjętych założeń a ich osobiste nastawienie w jak najmniejszym stopniu wpływało na wykonywaną prace

**K2. 1.5.2. Różnice w sposobie Myślenia Testerów i Programistów**

Programiści i testerzy często prezentują odmienny sposób myślenia

Sposób myślenia danej osoby wyznaczają przyjmowane przez nią założenia oraz preferowane sposoby podejmowania decyzji i rozwiązywania problemów

Wśród pożądanych cech osobowościowych testera należy wymienić: ciekawość, umiejętność krytycznego spojrzenia, dbałość o szczegóły  oraz pozytywne komunikowanie się w relacjach zawodowych.

Programiści są często bardziej zainteresowani projektowaniem i budowaniem rozwiązań niż analizowaniem ewentualnych problemów

Programiści często nie uświadamiają sobie popełnionych przez siebie błędów

Programiści mający odpowiednie nastawienie mogą testować własny kod

Niezależni testerzy mają inny punkt widzenia różny od punktów widzenia autorów produktów prac

**II. TESTOWANIE W CYKLU ŻYCIA OPROGRAMOWANIA**

**Ogólnie 2.1 Modele Cyklu Życia oprogramowania**

Model cyklu życia oprogramowania opisuje rodzaje czynności wykonywanych na poszczególnych etapach projektu wytwarzania oprogramowania oraz powiązania logiczne i chronologiczne między tymi czynnościami. Istnieje wiele różnych modeli cyklu życia oprogramowania a każdy z nich wymaga innego podejścia do testowania.

**K2. 2.1.1. Wytwarzanie Oprogramowania a Testowanie Oprogramowania**

relacje między czynnościami związanymi z wytwarzaniem oprogramowania a czynnościami testowymi

Znajomość powszechnie stosowanych modeli cyklu życia oprogramowania jest ważny element obowiązków każdego testera ponieważ pozwala uwzględnić właściwe czynności testowe takie jak

Kilka zasad dobrych praktyk testowania które dotyczą każdego modelu cyklu życia oprogramowania:

-dla każdej czynności związanej z wytwarzaniem oprogramowania istnieje odpowiadająca jej czynność testowania

-każdy poziom testów ma przypisane cele odpowiednie do tego poziomu

-analizę i projektowanie testów na potrzeby danego poziomu testów należy rozpocząć podczas wykonania odpowiadającej danemu poziomowi czynności związanej z wytwarzaniem oprogramowania

-testerzy powinni uczestniczyć w dyskusjach dotyczących definiowania i doprecyzowania wymagań i założeń projektu oraz w przeglądach produktów pracy

Bez względu na wybrany model cyklu życia oprogramowania czynności testowe powinny rozpocząć się w początkowym etapie tego cyklu zgodnie z zasadą wczesnego testowania

Modele cyklu życia oprogramowania, kategorie:

**-sekwencyjne modele wytwarzania oprogramowania**

**-iteracyjne modele wytwarzania oprogramowania**

**Sekwencyjny model wytwarzania oprogramowania** opisuje proces wytwarzania oprogramowania jako liniowy przepływ czynności następujących kolejno po sobie. Oznacza to że każda faza tego procesu powinna rozpoczynać się po zakończeniu fazy poprzedniej.

**W modelu kaskadowym(iteracyjnym)** czynności związane z wytwarzaniem oprogramowania wykonuje się jedna po drugiej. /czynności testowe występują dopiero wtedy gdy wszystkie inne czynności wytwórcze zostaną ukończone\

W PRZECIWIENSTWIE DO MODELI SEKWENCYJNYCH**w modelach iteracyjno-przyrostowyc**h działające **oprogramowanie może być dostarczone już w ciągu tygodni a nawet dni** jednak spełnienie wszystkich wymagań potrzeba miesięcy a nawet lat.

MODEL V -  w przeciwieństwie do modelu kaskadowego **zakłada integrację procesu testowania z całym procesem wytwarzania oprogramowania a tym samym wprowadza** **zasadę wczesnego testowania**

 W większości przypadków sekwencyjne modele wytwarzania oprogramowania pozwalają na wytworzenie oprogramowania posiadającego pełny zestaw funkcjonalności

Wytwarzanie iteracyjne polega na specyfikowaniu, projektowaniu, budowaniu i testowaniu wspólnie grup funkcjonalności w serii cykli o określonym czasie trwania.

Wśród przykładów modeli iteracyjnych można wyróżnić:

-Rational Unified Process(RUP): poszczególne iteracje trwają zwykle stosunkowo długo a przyrosty funkcjonalności są odpowiednio duże

-SCRUM: poszczególne iteracje trwają stosunkowo krótko(kilka godzin dni tygodni) a przyrosty są stosunkowo małe

-Kaban; wdrażany ze stałą lub zmienną długością iteracji umożliwia dostarczenie jednego udoskonalenia lub jednej funkcjonalności naraz bądź zgrupowanie większej liczby funkcjonalności

-Model spiralny: tworzy eksperymentalne elementy przyrostowe które następnie mogą zostać gruntownie przebadane a nawet porzucone na dalszych etapach wytwarzania oprogramowania.

Składowe komponenty systemów lub systemy wykorzystujące powyższe modele często odnoszą się do nachodzących na siebie i powtarzalnych w cyklu życia oprogramowania poziomu testów.

W idealnej sytuacji każda funkcjonalność jest testowana na wielu poziomach

W każdym z powyższych modeli powstaje rosnący system który może być dostarczany użytkownikom końcowym na zasadzie dodawania pojedynczych funkcjonalności w określonych iteracjach lub w bardziej tradycyjny sposób w formie dużych wydań.

**K1. 2.1.2. Modele Cyklu Życia Oprogramowania w Kontekście:**

powody, dla których konieczne jest dostosowanie modeli cyklu życia oprogramowania do kontekstu

**Modele cyklu życia oprogramowania należy dobierać i dopasować do kontekstu** wynikającego z charakterystyki projektu i produktu. W związku z tym w procesie wyboru i dostosowania odpowiedniego modelu **należy wziąć pod uwagę:**

-cel projektu

-typ wytwarzanego produktu

-priorytety biznesowe

-zidentyfikowane czynniki ryzyka produktowego i projektowego

Różne modele cyklu życia oprogramowania można łączyć ze sobą. Przykładem może być zastosowanie modelu V do wytwarzania i testowania systemów back-endowych i ich integracji oraz modelu zwinnego do wytwarzania i testowania interfejsów użytkownika i funkcjonalności.

W przypadku systemów związanych z Internetem rzeczy które składają się z wielu różnych obiektów takich jak: urządzenia, produkty, i usługi w odniesieniu do poszczególnych stosuje się zwykle oddzielne modele cyklu życia oprogramowania. Stanowi to duże wyzwanie w zakresie wytwarzania poszczególnych wersji systemów

Przykładem może być wytwarzanie i testowanie wewnętrznego systemu administracyjnego o niewielkim znaczeniu jak wytwarzanie systemu układu hamulcowego w samochodzie.

W zależności od kontekstu projektu konieczne może okazać się połączenie lub reorganizacja niektórych poziomów testów.

**POWODY DLA KTÓRYCH MODELE ROZWOJU OPROGRAMOWANIA MUSZĄ BYĆ DOSTOSOWANE DO KONTEKSTU PROJEKTU I CHARAKTERYSTYKI PRODUKTU:**

-różnica w ryzykach produktowych systemów

-wiele jednostek biznesowych może być częścią projektu lub programu

-krótki czas na dostarczanie produktu na rynek

**Ogólnie 2.2. Poziomy Testów:**

Poziomy testów to grupy czynności testowych które organizują się i którymi zarządza się wspólnie.

Każdy poziom testów jest instancją procesu testowego.

Poziomy testów opisane w sylabusie to:

***-testowanie modułowe***

***-testowanie integracyjne***

***-testowanie systemowe***

***-testowanie akceptacyjne***

Każdy poziom testów charakteryzuje atrybuty:

-cele szczegółowe

-podstawa testów

-przedmiot testów

-typowe defekty i awarie

-konkretne podejścia i odpowiedzialności

**K2 2.2.1 Testowanie modułowe**

Cele Testowania modułowego:

Testowanie modułowe skupia się na modułach które można oddzielnie przetestować.

Cele tego typu testowania to m.in.:

-zmniejszenie ryzyka

-weryfikacja zgodności zachowań funkcjonalnych i niefunkcjonalnych modułu z projektem i specyfikacją

-budowanie zaufania do jakości modułu

-wykrywanie defektów w module

-zapobieganie przedostaniu się defektów na wyższe poziomy testowania

Podstawy Testów:

Podstawa testów w ramach testowania modułowego to:

-projekt szczegółowy

-kod

-model danych

-specyfikacja modułów

Przedmioty Testów:

-moduły, jednostki lub komponenty

-kod i struktury danych

-klasy

-moduły baz danych

Typowe Defekty i Awarie:

-niepoprawna funkcjonalność

-problemy z przepływem danych

-niepoprawny kod i logika

Defekty są zwykle usuwane natychmiast

Konkretne podejścia i odpowiedzialności:

Testowanie modułowe jest zwykle wykonywane przez programistę będącego autorem kodu. Programiści często wykonują i piszą testy po napisaniu kodu danego modułu.

**K2. 2.2.2. Testowanie Integracyjne:**

Testowanie Integracyjne skupia się na **interakcjach między modułami lub systemami.** Cele testowania integracyjnego to:

-zmniejszenie ryzyka

-weryfikacja zgodności zachowań funkcjonalnych i niefunkcjonalnych

-budowanie zaufania do jakości interfejsów

-wykrywanie defektów

-zapobieganie przedostawaniu się defektów na wyższe poziomy testowania

 Dwa różne poziomy testowania integracyjnego w zależności od wielkości przedmiotów testów:

**Testowanie integracji** modułów skupia się na interakcjach i interfejsach między integrowanymi modułami. Testy tego typu wykonuje się po zakończeniu testowania modułowego

**Testowanie integracji systemów** skupia się na interakcjach i interfejsach między systemami, pakietami i mikrousługami. Tego typu testy mogą również obejmować interakcje z interfejsami dostarczanymi przez organizacje zewnętrzne.

Podsatawa testów:

-projekt oprogramowania i systemu

-specyfikacje interfejsow i protokołow komunikacyjnych

-przypadki użycia

-architektura na poziomie modułów systemu

-przepływy pracy

-definicje interfejsów zewnętrznych

Przedmioty Testów:

Do typowych przdmiotów testów zalicza się:

-podsystemy

-bazy danych

-infrastrukturę

-interfejsy

-interfejsy programowania aplikacji

-mikrousługi

Typowe Defekty i Awarie

Przykłądami  typowych defektów awarii wykrywanych w ramach testowania integracji modułów są:

-niepoprawne lub brakujące dane bądź niepoprawne kodowanie danych

-niepoprawne uszeregowanie lub niepoprawna synchronizacja wywołań interfejsów

-niezgodności interfejsów

-błędy komunikacji między modułami

- brak obsługi lub nieprawidłowa obsługa błędów komunikacyjnych między modułami

Przykładami typowych defektów awarii wykrywanych w ramach testowania integracji systemów są:

-niespójne struktury komunikatów przesyłanych między systemami

-niepoprawne lub brakujące dane

-niezgodnośc interfejsów

-błędy komunikajci między systemami

-brak obsługi lub nieprawidłowa obsługa błędów komunikacji między systemami

-niepoprawne założenia

-nieprzestrzeganie rygorystycznych przepisów dotyczących zabezpieczeń

Konkretne Podejścia i Odpowiedzialności:

Testowanie integracji modułów i testowanie systemów powinno koncentrować się na samej integracji.

Testowanie integracji modułów jest często obowiązkiem programistów natomiast za testowanie integracji systemów zwykle odpowiadają testerzy.

Usystematyzowane startegie integracji mogą być oparte na architekturze systemu.

Im szerszy zakres integracji tym trudniej jest wskazać konkretny moduł lub system w którym wystąpił defekt co może prowadzić do wzrostu ryzyka.

**K2 2.2.3. Testowanie Systemowe:**

Cele Testowania Systemowego:

Testowanie systemowe skupia się na **zachowaniu i możliwościach całego systemu lub produktu**.

Cele:

-zmniejszanie ryzyka

-walidacjia zgodności zachowań funkcjonalnych i niefunkcjonalnych systemu z projektem i specyfikacjami

-walidacja czy system jest kompletny i działa zgodnie z oczekiwaniami

-wykrywanie defektów

-zapobieganie przedostawaniu się defektów na wyższe poziomy testowania lub na produkcje

Podstawa Testów:

-specyfikacje wymagań

-raporty z analizy ryzyka

-przypadki użycia

-opowieści i historyjki użytkownika

-modele zachowan systemu

-diagramy stanów

-instrukcje obsługi systemu i podręczniki użytkowanika

Przedmioty Testów:

-aplikacje

-systemy łączące sprzęt i oprogramowania

-systemy operacyjne

-systemy podlegające testowaniu

-konfiguracje i dane konfiguracyjne systemu

Typowe Defekty i Awarie:

-niepoprawne oblicznia

-niepoprawne lub nieoczekiwane zachowania funkcjonalne i niefunkcjonalne systemu

-niepoprawne przepływy danych w systemie

-problemy z prawidłowymi działaniem systemu w środowisku testów systemowych

-problemy z prawidłowym działaniem systemu w środowisku

-niezgodności działania systemu

Konkretne podejścia i odpowiedzialności:

Testowanie systemowe powinno koncentrować się na kompleksowym zbadaniu zachowania systemu jako całości na poziomie funkcjonalnym i niefunkcjonalnym

Testowanie systemowe jest przeprowadzane przez niezależnych testerów

**K2. 2.2.4. Testowanie Akceptacyjne:**

Skupia się na **zachowaniu i możliwościach całego systemu lub produktu**.

 Cele:

-budowanie zaufania do systemu

-walidacjia czy system jest kompletny

-werfikacja zgodności zachowania funkcjonalnego i niefunkcjonalnego systemu z specyfikacja

Podczas testowania akceptacyjnego moą też zostać wykryte defekty ale nie są częst ich celem

Najczęstsze formy testowania akceptacyjnego to:

-testowanie akceptacyjne przez użytkownika

-produkcyjne testy akcpetacyjne

-testowanie alfa i beta

Testowanie Akceptacyjne Przez Uzytkownika(UAT): odbywa się w środowisku produkcyjnym i zwykle skupia się na walidacji czy system nadaje się do użytkowania przez przyszłych odbiorców. Głównym cele jest zbudowanie pewności że system spełnia wymagania.

Produkcyjne Testy Akcetacyjne(OAT):

Testy skupiają się na aspektach operacyjnych i mogą obejmować:

-testowanie mechanizmów tworzenia kopi zapasowych

-instalowanie odinstalowanie i aktualiowanie oprogramowania

-zarządzanie użytkownikami

-wykonanie czynności pielęgnacyjnych

-sprawdzanie czy występują słabe punkty zabezpieczeń

-wykonywanie testów wydajnościowych

Testowanie Akceptacyjne Zgodności z Umową i Przepisami Prawa:

Testowanie Alfa i Beta:

Testowanie alfa jest wykonywane w siedzibie organizacji wytwarzającej oprogramowanie wykonywane przez klientów.

Testowanie Beta wykonują klienci we własnych lokalizacjach

Cel: budowanie zaufania klientów

Podstawa Testów:

-procesy biznesowe

-wymagania użytkowników

-przepisy, umowy normy i standardy

-przypadki użycia

-wymagania systemowe

-dokumentacja systemu

-procedury instalacji

-raporty z analizy ryzyka

Typowe Przedmioty Testów:

-system podlegający testowaniu

-konfiguracja i dane konfiguracyjne systemu

-procesy biznesowe

-formularze

-raporty

Typowe Defekty i Awarie:

-niepoprawnie zaimplementowane reguły biznsowe

-niespełnienie przez system wymagań

-awarie niefunkcjonalne

Konkretne Podejścia i Obowiązki:

Testowanie akceptacyjne często spoczywa na klientach użytkownikach i właścicielach.

**Ogólnie 2.3 Typy Testów:** To grupa dynamicznych czynności testowych wykonywanych z myśla o przetestowaniu określonych charakterystyk systemu/oprogramowania. Celami testów są:

-Ocena funkcjonalności charakterystyk jakościowych

-Dokonanie oceny niefunkcjonalnych charakterystyk jakościowych w tym parametrów

-ustalenie czy struktura lub architektura modułu lub systemu jest poprawna

-dokonywanie skutków zmian

**K2. 2.3.1. Testowanie Funkcjonalne:** systemu polega na wykonywaniu testów które umożliwiają dokonanie oceny funkcji jakie system powinien realizować

**Funkcje opisują to co powinien robić system**

Testy funkcjonalne należy wykonywać na wszystkich poziomach testów

Testowanie funkcjonalne uwzględnia zachowanie oprogramowania

Do projektowania i wykonywania testów funkcjonalnych mogą być potrzebne specjalne umiejętności lub specjalna wiedza.

**K1. 2.3.2. Testowanie Niefunkcjonalne:**

Testowanie niefunkcjonalne pozwala sprawdzić to **jak dobrze zachowuje się dany system.**

Celem testowania niefunkcjonalnego jest dokonywanie oceny charakterystyk systemów i oprogramowania.

Klasyfikacje charakterystyk jakościowych oprogramowania zawiera ISO/IEC 25010

Wbrew mylemu przekonaniu testy niefunkcjonalne można i należy prowadzić/wykonywać na każdym poziomie testów.

NA potrzeby testowania niefunkcjonalnego można używać technik czarnoskrzynkowych

Staranność testowania niefunkcjonalnego można zmierzyć na podstawie pokrycia niefunkcjonalnego

Do projektowania i wykonywania testów niefunkcjonalnych mogą być potrzebne specjalne umiejętności lub wiedza

**K2. 2.3.3. Testowanie Białoskrzynkowe:**

W przypadku testowania białoskrzynkowego warunki testowe wyprowadza się na podstawie struktury wewnętrznej lub implementacji danego systemu

Staranność testowania białoskrzynkowego można zmierzyć na podstawie pokrycia strukturalnego.

**Powyższe rodzaje pokrycia nazywa się zbiorczo pokryciem kodu.**

**K2. 2.3.4. Testowanie Związane ze Zmianami:**

Po wprowadzeniu zmian mających na celu usunięcie defektów bądź dodanie zmodyfikowanej funkcjonalności należy przeprowadzić testy które potwierdzą iż modyfikacje nie spowodowały defektów, nieprzewidzianych zmian, niekorzystnych zachowań.

**Testowanie potwierdzające:** Celem testu potwierdzającego jest sprawdzenie czy pierwotny defekt został pomyślnie usunięty.

**Testowanie regresyjne:** polega na ponownym wykonaniu testów po wprowadzeniu zmian w celu ich wykrycia.

Testowanie potwierdzające i testowanie regresji można wykonać na każdym poziomie testów

Z uwagi na ciągłą ewolucję systemu testowanie potwierdzające i testowanie regresji są bardzo istotne.

**K22. 2.3.5. Poziomy Testów a Typy Testów:**

Przykłady testów funkcjonalnych:

-na potrzeby testowania moduł powinien obliczać odsetki składane

-na potrzeby testowania w jaki system sprawdza dolnośc kredytową

-na potrzeby testowania w jaki sposób pracownik banku zatwierdza lub odrzuca wniosek kredytowy

Przykłady testów niefunkcjonalnych:

-na potrzeby testowania śa testy wydajnościowe

-na potrzeby testowania testy związane z zabezpieczeniem systemu

-na potrzeby testowania

Przykłady testów białoskrzynkowe:

-na potrzeby testowania celem pełnego pokrycia instrukcji kodu i decyzji

-na potrzeby testowania sprawdzanie w jaki sposób interfejs przekazuje wyświetlane dane na ekranie

-na potrzeby testowania sprawdzeniem wszystkich typów zapytan

Przykłady testów związanych ze zmianami:

-na potrzeby testowania prokjektowane są automatyczne testy regresji dotyczące poszczególnych modułow

-na potrzeby testowania sa przeprowadzane testy potwierdzające skuteczność wporwadzanych poprawek

-na potrzeby testowania akceptacyjnego wykonywane sa wszystkie wcześniej niezaliczone testy

**Ogólne 2.4. Testy Pielęgnacyjne**

Po wdrożeniu oprogramowania system wymaga dalszej pielęgnacji.

Po dokonaniu każdej zmiany w fazie pielęgnacji należy wykonać testowanie pielęgnacyjne którego celem jest sprawdzenie czy zmiana została wprowadzona pomyślnie jak i wykrycie niezamierzonych skutków ubocznych.

Pielęgnacja może przebiegać planowo jaki  i nieplanowo czy doraźnie.

Na zakres testowania pielęgnacyjnego wpływają między innymi:

-poziom ryzyka związanego z zmianą

-wielkość dotychczasowego systemu

-wielkośc wprowadzonej zmiany

**K2. 2.4.1. Zdarzenia wywołujące pielęgnacyjne:**

Zdarzenia pielęgnacyjne można podzielić na nastepujace kategorie:

-Modyfikacja – obejmuje zmiany korekcyjne, awaryjne, środowiska operacyjnego, uaktualnienia oprorgramowania, poprawki usuwające defekty

-Migracyjna- obejmuje przejście z jednej platformy na inna

-Wycofanie – kiedy aplikacja lub system jest wycofywany może to wymagac testowania migracji lub archwizacji danych TU wyminiamy: testowanie procedur odzyskania / pozyskania zarchiwzowania danych

W przypadku systemów związanych z internetem testowanie pielęgnacyjne może być konieczne po wprowadznieu zmodyfikowanych elelmentów

**K2. 2.4.2 Analiza Wpływu Związana z Pielęgnacja:**

**Analiza wpływu pomaga zdecydować, czy wprowadzenie danej poprawki w pielęgnowaniu systemie jest zasadne**

Pozwala ocenić zmiany wprowadzone w wersji pielęgnacyjne pod kątem skutków zamierzonych jak i spodziewanych skutków ubocznych.

Analizę wpływu można przeprowadzić przed dokonaniem zmiany aby ustalić czy zmianę te należy faktycznie wprowadzić

Przeprowadzenie analizy wpływu może być utrudnione jeśli:

-specyfikacje są nieaktualne lub niedostępne

-przypadki testowe nie sa aktualne

-wsparcie narzędziowe nie istnieje

-zaangażowane osoby nie dysponują wiedza

-podczas wytwarzania oprogramowania poświęcono zbyt mało czasu charakterystyce jakościowej

**Ogólnie. 3.1. Podstawy Testowania Statycznego:**

W przeciwieństwie do testowania dynamicznego które wymaga uruchomienia oprogramowania  testowanie statyczne opiera się na ręcznym badaniu produktów pracy bądź dokonaniu oceny kodu przy użyciu odpowiednich narzędzi.

Analiza statyczna jest w szczególności ważna w przypadku systemów komputerowych krytycznych ze względów bezpieczeństwa ale staje się coraz bardziej istotna i powszechnie stosowana w innych kontekstach.

**K1. 3.1.1. Produkty Pracy Badane Metodą Testowania Statycznego:**

Przy użyciu technik testowania statycznego można zbadać niemal wszystkie produkty pracy np.:

**-specyfikacje**

**-kryteria akceptacji-architekturę i specyfikacje projektowe**

**-kod**

**-testalia**

**-podręczniki użytkownika**

**-strony internetowe**

**-umowy, plany projektów**

**-konfiguracje i infrastrukturę**

Analizę statyczną można zastosować efektywnie do każdego produktu pracy o strukturze formalnej dla którego istnieje odpowiednie narzędzie do jej przeprowadzenia.

**K2.3.1.2. Zalety Testowania Statycznego**

**Po wdrożeniu techniki analizy statycznej wykrywane są usterki w kodzie, które nie zostałyby wykryte w przypadku wykonywania tylko testów dynamicznych.**

Techniki testowania statycznego dostarczają wielu korzyści. W przypadku ich stosowania w początkowym etapie cyklu życia oprogramowania pozwalają one wykryć defekty jeszcze przed rozpoczęciem testowania dynamicznego.

Zastosowanie technik testowania statycznego do wykrywania defektów i ich niezwłocznego usuwania jest prawie zawsze dużo tańsze niż wykrywanie defektów w przedmiocie testów i usuwania ich metodą testowania dynamicznego.

Dodatkowe zalety testowania statycznego:

-efektywne wykrywanie i usuwanie defektów jeszcze przed wykonaniem testów dynamicznych

-identyfikowanie defektów które trudno jest wykryć metodą testowania dynamicznego

-zapobieganie wystąpienia defektów w projekcie  i kodzie

-zwiększenie wydajności prac programistycznych

-obniżenie kosztów i zmniejszenie czasochłonności wytwarzania oprogramowania

-obniżenie kosztów i zmniejszenie czasochłonności testowania

-obniżenie łącznego kosztu zapewniania jakości w całym cyklu życia oprogramowania

**K2.3.1.3. Różnice Między Testowaniem Statycznym a Dynamicznym:**

Testowanie statyczne i dynamiczne mogą mieć te same cele takie jak dokonywanie oceny jakości produktów pracy czy jak najwcześniejsze identyfikowanie defektów.

GŁÓWNA  RÓŻNICA MIĘDZY TESTOWANIEM STATYCZNYM A DYNAMICZNYM POLEGA NA TYM ŻE TESTOWANIE STATYCZNE POZWALA WYKRYĆ DEFEKTY BEZPOŚREDNIO W PRODUKTACH PRACY, A NIE NA PODSTAWIE SPOWODOWANYCH PRZEZ TE DEFEKTY AWARII ZIDENTYFIKOWANYCH PODCZAS URUCHAMIANIA OPROGRAMOWANIA.

Zdarza się że defekt pozostaje nie wykryty przez długi czas ponieważ nie powoduje awarii.

Testowanie statyczne może być wykorzystywane do zwiększenia wewnętrznej jakości natomiast dynamiczne koncentruje się głównie na zewnętrznych zachowaniach systemu/oprogramowania.

Przykłady typowych defektów które są łatwiejsze do wykrycia i usunięcia przy zastosowaniu metod testowania statycznego:

-defekty w wymaganiach

-defekty w projekcie

-defekty w kodzie

-odchylenia od standardów

-niepoprawne specyfikacje interfejsów

-słabe punkty zabezpieczeń

**K2.3.2.Proces Przeglądu:**

Przeglądy mogą mieć różny charakter: od nieformalnego po formalny.

Cechą charakterystyczną przeglądów nieformalnych jest to iż nie przebiegają one zgodnie ze zdefiniowanym procesem i nie trzeba ich dokumentować.

Przeglądy  formalne są przeprowadzane zgodnie z udokumentowanymi procedurami  i muszą być obowiązkowo dokumentowane.

To na czym koncentruje się przegląd zależy od uzgodnionych celów przeglądu.

Szczegółowy  opis przeglądu produktów pracy zawiera międzynarodowy standard ISO/IEC/20246

**K2.3.2.1. Proces Przeglądu (FORMALNEGO) Produktów Pracy:** obejmuje następujące czynności:

**W ramach rozpoczęcia przeglądu recenzenci otrzymują listy kontrolne przygotowane do przeglądu.**

**PLANOWANIE**: -określenie zakresu prac, -oszacowanie nakładu pracy i czasu, -wybór osób, -określenie kryteriów wejścia i wyjścia, -sprawdzenie czy zostały spełnione kryteria wejścia

**ROZPOCZĘCIE PRZEGLĄDU**: -rozesłanie produktów prac, -wyjaśnienie uczestnikom zakresów celów, - udzielenie odpowiedzi na ewentualne pytania

**PRZEGLĄD INDYWIDUALNY:** -dokonanie przeglądu całości, - odnotowanie potencjalnych defektów

**PRZEKAZANIE INFORMACJI O PROBLEMACH I ANALIZA PROBLEMÓW:** -przekazanie informacji  o potencjalnych defektach, -przeanalizowanie defektów, -dokonanie oceny wniosków z przeglądu

**USUNIĘCIE DEFEKTÓW I RAPORTOWANIE:** - tworzenie raportów o defektach, -usunięcie defektów, -sprawdzenie czy zostały spełnione kryteria wyjścia

**K1.3.2.2. Role i Obowiązki w Przeglądzie Formalnym:**

W typowym formalnym przeglądzie wyróżnia się następujące role:

**AUTOR:**

-tworzy produkty pracy będące przedmiotem przeglądu

-usuwa defekty w produkcie

**KIEROWNICTWO:**

-odpowiada za zaplanowanie przeglądu

-podejmuję decyzje o przeprowadzeniu przeglądu

-wyznacza pracowników i określa budżet

-monitoruje na bieżąco opłacalność przeglądu

-wykonuje decyzje kontrolne

**FACYLITATOR(Moderator):**

-dba o sprawny przebieg spotkań związanych z przeglądem

-występuje w roli mediatora

-jest często osobą od której zależy powodzenie przeglądu

**LIDER PRZEGLĄDU:**

-ponosi ogólną odpowiedzialność za przegląd

-decyduje o tym kto ma wziąć udział w przeglądzie

**PRZEGLĄDAJĄCY:**

-mogą być ekspertami merytorycznymi

--mogą reprezentować różne punkty widzenia

**PROTOKOLANT:**

-gromadzi informacje o potencjalnych defektach

-rejestruje nowe potencjalne defekty

W przypadku niektórych przeglądów jedna osoba może pełnić kilka ról. Bardziej szczegółowe role opisane w standardzie ISO/IEC 20246

**K2.3.2.3.Typy Przeglądów:**

Przeglądy mogą być przeprowadzane w różnych celach ale jednym z głównych celów jest ujawnianie usterek. W wykryciu defektów mogą pomóc przeglądy wszystkich typów.

**Przegląd nieformalny** (np. sprawdzenie koleżeńskie)**:**

Cel główny: wykrycie potencjalnych defektów. / Wygenerowanie nowych pomysłów, rozwiązań.

Przegląd nie odbywa się zgodnie z formalnym(udokumentowanym) procesem

Przegląd nie musi obejmować spotkania

Rezultaty przeglądu mogą zostać udokumentowane

**Przejrzenie:**

Cele Główne: wykrycie potencjalnych defektów, podniesienie jakości oprogramowania, rozważanie alternatywnych implementacji, dokonanie oceny zgodności z normami/standardami.

-Spotkanie związane z przeglądem prowadzi zwykle autor produktu pracy

-Wymagany jest udział protokolanta

-Przegląd może mieć formę scenariuszy

-Do tego przeglądu są tworzone dzienniki (log) potencjalnych defektów i raporty z przeglądu

-Charakter przeglądu może w praktyce być różny- od nieformalnego po bardzo formalny

**Przegląd techniczny:**

Cele główne: Uzyskanie konsensus, wykrycie potencjalnych defektów, zwiększenie zaufania, wyg.nowych pomysłów.

-Przeglądający powinni mieć takie same kompetencje techniczne jak autor oraz powinni być ekspertami technicznymi

-Wymagany jest udział protokolanta

-spotkanie związane z przeglądem jest opcjonalne

**Inspekcja:**

Cele główne: wykrycie potencjalnych defektów, dokonanie jakości produktu pracy i zwiększenie zaufania do niego, zapobieganie w przyszłości podobnych defektów.

Przegląd odbywa się zgodnie z zdefiniowanym procesem opartym na regułach i listach kontrolnych

Występują jednoznacznie określone role

Obowiązują określone kryteria wejścia i wyjścia

Obecność protokolanta jest obowiązkowa

Autor nie może być liderem przeglądu

Mogą być tworzone dzienniki potencjalnych defektów i raporty z przeglądu

**K2/K3.3.2.4. Stosowanie Techniki Przeglądu**

Przykładowe techniki przeglądu indywidualnego w kontekście różnych typów przeglądów:

**Przegląd ad hoc**: to powszechnie stosowana technika której stosowanie nie wymaga intensywnych przygotowań.

Przegląd ad hoc: przeglądający otrzymują niewiele wskazówek dotyczących sposobu wykonywania zadania.  Ponadto ta technika wiąże się z ryzykiem wielokrotnego zgłaszania tych samych problemów.

**Przegląd oparty na liście kontrolnej:** to usystematyzowana technika zgodnie z którą przeglądający wykrywa problemy na podstawie list kontrolnych rozsyłanych w momencie rozpoczęcia przeglądu.

Lista kontrolna zawiera zestaw pytań odpowiadających potencjalnym defektom.

Najważniejszą zaletą techniki opartej na listach kontrolnych jest systematyczne pokrycie najczęściej występujących defektów.

**Scenariusze i przebiegi próbne :**  Przeglądający otrzymują ustrukturyzowane wytyczne dotyczące sposobu zapoznawania się z produktem pracy. Dzięki scenariuszom przeglądający mogą skuteczniej identyfikować określone typy defektów niż przy użyciu prostych list kontrolnych.

**Czytanie oparte na perspektywie:** W przypadku czytania opartego na perspektywie podobnie jak w przypadku przeglądu opartego na rolach przeglądający przyjmują indywidualne punkty widzenia różnych interesariuszy.

Badania empiryczne wykazały że czytanie oparte na perspektywie jest zazwyczaj najskuteczniejszą techniką przeglądu wymagań i produktów pracy o charakterze technicznym.

**Przegląd oparty na rolach:** to technika w której przeglądający oceniają produkt pracy z perspektywy poszczególnych ról przypisanych interesariuszom. W tym przeglądzie obowiązują te same zasady co w czytaniu opartym na perspektywie.

**K2.3.2.5. Czynniki Powodzenia Związane z Przeglądami:**

Kluczem do pomyślnego przeprowadzenia przeglądu jest staranny dobór typu przeglądu i stosowanych technik

Czynniki o charakterze organizacyjnym

Czynniki o charakterze kadrowym

Czynniki o charakterze organizacyjnym:

-każdy przegląd ma jednoznaczne cele

-stosowane typy przeglądów sprzyjają osiągnięciu założonych celów

-stosowane są różne techniki przeglądu

-stosowane listy kontrolne są aktualne

-uczestnicy maja wystarczająco czasu na przygotowanie

-przeglądy są planowane z wyprzedzeniem

-przeglądy są spójne z polityką jakości

Czynniki o charakterze kadrowym:

-w przeglądy są zaangażowane odpowiednie osoby

-testerzy są uznawani za ważnych uczestników przeglądu

-przeglądane są małe fragmenty dokumentacji

-informacje o wykrytych defektach są przyjmowane do wiadomości

-przeglądy odbywają się w atmosferze wzajemnego zaufania

**IV. TECHNIKI TESTOWANIA**

**Ogólnie 4.1. Kategorie Technik Testowania:**

Każda technika testowania ma z założenia **pomagać w identyfikowaniu warunków testowych, przypadków testowych i danych testowych.**

Przy wyborze technik należy wsiąść pod uwagę czynniki takie jak:

złożoność modułu lub systemu;

• obowiązujące przepisy i normy;

• wymagania klienta lub wymagania wynikające z umów;

• poziomy ryzyka i jego rodzaj;

• dostępną dokumentację;

• wiedzę i umiejętności testerów;

• dostępne narzędzia;

 • czas i budżet;

• model cyklu życia oprogramowania;

 • typy defektów spodziewane w modułach i systemach.

Niektóre techniki testowania lepiej się sprawdzą w określonych sytuacjach.

Użycie  technik testowania w analizie projektowaniu i implementacji testów może mieć zróżnicowany charakter od bardzo nieformalnego po bardzo formalny.

**K2.4.1.1. Kategorie Technik Testowania i Ich Cechy Charakterystyczne:**

Techniki testowania podzielono na: czarnoskrzynkowe, białoskrzynkowe i oparte na doświadczeniu.

Techniki czarnoskrzynkowe – **bazują na analizie podstaw testów.** Techniki z tej grupy można stosować zarówno w testowaniu funkcjonalnym jak i niefunkcjonalnym. Techniki czarnoskrzynkowe koncentrują się na danych wejściowych i wyjściowych przedmiotu testów bez odwoływania się do jego struktury wewnętrznej

Techniki białoskrzynkowe – **podstawą białoskrzynkowych technik jest analiza architektury, szczegółowego projektu, struktury wewnętrznej lub kodu przedmiotu testów.** W przeciwieństwie do czarnoskrzynkowych technik białoskrzynkowe techniki koncentrują się na strukturze i przetwarzaniu wewnątrz przedmiotu testów.

Techniki testowania oparte na doświadczeniach – pozwalają wykorzystać doświadczenie developerów testerów i użytkowników do projektowania, implementowania i wykonywania testów. Techniki te często są wykorzystywane z technikami czarno skrzynkowymi i biało skrzynkowymi.

**K2. 4.2 CZARNOSKRZYNKOWE TECHNIKI TESTOWANIA**

**K3.4.2.1. Podział na Klasy Równoważności:**

Technika podziału na klasy równoważności **polega na dzieleniu danych na grupy** w taki sposób aby każda grupa zawierała elementy przetwarzania w ten sam sposób.

(polega na dzieleniu danych na grupy w  taki sposób by każda grupa

Klasy równoważności można wyznaczyć zarówno dla wartości poprawnych jak i niepoprawnych:

-wartości poprawne to wartości które powinny zostać zaakceptowane przez moduł lub system

-wartości niepoprawne to wartości które moduł lub system powinien odrzucić.

-każda wartość musi należeć do jednej i tylko jednej klasy równoważności

**Warunkiem uzyskania stuprocentowego pokrycia** przy korzystaniu z tej technik**i jest pokrycie przez przypadki testowe wszystkich zidentyfikowanych klas równoważności.**

Podział na klasy równoważności można stosować na wszystkich poziomach testów.

**K3.4.2.2. Analiza Wartości Brzegowych:**

Technika analizy wartości brzegowych jest rozszerzeniem techniki podziału na klasy równoważności ale może być stosowana tylko w przypadku uporządkowanych klas zawierających dane liczbowe lub sekwencyjne.

Wartościami brzegowymi klasy równoważności są jej wartość minimalna i maksymalna.

Technika ta służy zwykle do testowania wymagań które odwołują się przedziału liczb.

Analiza Wartości Brzegowych można stosować na wszystkich poziomach testów.

*Akceptowalny zakres wartości całkowitych mieści się przedziale od 1 do 5 włącznie. W tym przypadku możemy więc wyróżnić trzy klasy równoważności: niepoprawna (wartości za niskie), poprawna, niepoprawna (wartości za wysokie). Dla poprawnej klasy równoważności wartości brzegowe to 1 i 5. Dla niepoprawnej klasy równoważności (za wysokie) wartość brzegowa to 6. Dla niepoprawnej klasy równoważności (za niskie) jest tylko jedna wartość brzegowa 0, ponieważ jest to klasa z jednym tylko przedstawicielem.*

**K3. 4.2.3. Testowanie w Oparciu o Tablicę Decyzyjną:**

jest dobrym sposobem na modelowanie złożonych reguł biznesowych które muszą zostać zaimplementowane w systemie.

Wartość warunków i akcji przedstawia się zwykle jako wartości logiczne (prawda/fałsz) lub dyskretnie(czerwony/zielony/niebieski)

Warunki:

-„T” oznacza że warunek został spełniony

-„N” oznacza że warunek nie został spełniony

Akcje:

-„X” oznacza że akcja powinna zostać wykonana

-Puste pole oznacza że akcja nie powinna zostać wykonana

Pełna tablica decyzyjna zawiera tyle kolumn ile jest niezbędnych do pokrycia wszystkich kombinacji warunków.

Minimalnym wymogiem  w przypadku testowania w oparciu o tablicę decyzyjną jest zawsze utworzenie co najmniej jednego przypadku testowego dla każdej reguły decyzyjnej w tablicy.

**Zaletą testowania w oparciu o tablicę decyzyjną jest możliwość zidentyfikowania wszystkich** ważnych **istotnych kombinacji warunków.**

Ponadto ta metoda pomaga znaleźć ewentualne luki w wymaganiach.

**K3.4.2.4 Testowanie Przejść Pomiędzy Stanami:**

Moduł lub system mogą różnie reagować na dane wejściowe w zależności od warunków bieżących lub historycznych.

Przejście jest inicjowane przez zdarzenie. Wynikiem zdarzenia jest przejście z stanu do stanu.

Diagramy przejść między stanami przedstawiają zwykle tylko poprawne przejścia.

Technikę testowania między stanami stosuje się w przypadku aplikacji wyposażonych w menu.

Jest ona również rozpowszechniona w branży oprogramowania wbudowanego.

**K3.4.2.5. Testowanie Oparte na Przypadkach Użycia:**

Testy można wyprowadzić z przypadków użycia. **Opisują one interakcje z elementami oprogramowania weryfikując wymagania funkcjonalności.** Z przypadkami użycia związane są pojęcia „aktorów” oraz „podmiotów”.

Każdy przypadek użycia określa konkretne działanie które podmiot może wykonywać we współpracy z jednym lub kilkoma autorami. Interakcje między aktorami a przedmiotem mogą powodować zmianę stanu podmiotu.

Testy projektuje się tak aby umożliwiały sprawdzenie wszystkich zdefiniowanych zachowań.

**Ogólnie 4.3. Białoskrzynkowe Techniki Testowania:**

Testowanie białoskrzynkowe opiera się na strukturze wewnętrznej przedmiotu testów.

Techniki białoskrzynkowe mogą być stosowane na wszystkich poziomach testów jednakże najczęściej stosuje się na poziomie testów modułowych.

**K2. 4.3.1. Testowanie i Pokrycie Instrukcji Kodu.**

Testowanie instrukcji służy do sprawdzenia potencjalnie wykonywalnych instrukcji zawartych w kodzie. Pokrycie mierzy się procentowo jako iloraz liczby instrukcji wykonanych przez testy zazwyczaj wyrażane w procentach

**K2. 4.3.2. Testowanie i Pokrycie Decyzji :**

Testowanie decyzji służy do **sprawdzenia decyzji zawartych w kodzie** oraz kodu wykonywanego na podstawie wyników decyzji. W tym celu tworzy się przypadki testowe które odzwierciedlają przepływy sterowania występujące po punkcie decyzyjnym.

W przypadku instrukcji IF tworzy się jeden przypadek dotyczący spełnienia warunku i jeden przypadek dotyczący niespełnienia warunku

W przypadku instrukcji CASE niezbędne są przypadki testowe odpowiadające wszystkim możliwym wynikom.

**K2.4.3.3. Korzyści Wynikające z Testowania Instrukcji i Testowania Decyzji:**

Uzyskanie stuprocentowego pokrycia kodu gwarantuje że wszystkie instrukcje wykonywalne zawarte w kodzie zostały przetestowane co najmniej raz, nie gwarantuje że przetestowana została cała logika decyzyjna.

Uzyskanie stuprocentowego pokrycia oznacza że wykonano wszystkie wyniki decyzji czyli przetestowano wyniki zarówno „prawda” jak i „fałsz”.

Uzyskanie stuprocentowego pokrycia decyzji gwarantuje stuprocentowe pokrycie instrukcji kodu.

**Ogólnie. 4.4. Techniki Testowania Oparte na Doświadczeniu.**

W ramach technik opartych na doświadczeniu przypadki testowe projektuje się z wykorzystaniem umiejętności  i intuicji testerów oraz ich doświadczenia z podobnymi aplikacjami i technologiami.

**K2. 4.4.1 Zgadywanie błędów:** to technika pozwalająca przewidzieć wystąpienie błędów, defektów i awarii na podstawie wiedzy testera dotyczącej między innymi:

-dotychczasowego działania aplikacji

-typowych popełnianych pomyłek

-awarii które wystąpiły w innych aplikacjach

Listy pomyłek i awarii można opracowywać na podstawie własnego doświadczenia danych dotyczących defektów i awarii.

**K2. 4.4.2. Testowanie Eksploracyjne:** polega na projektowaniu, wykonywaniu i rejestrowaniu testów nieformalnych oraz dokonywaniu ich oceny w sposób dynamiczny podczas ich wykonywania.

Rezultaty dostarczają wiedzy na temat modułu lub systemu.

Testowanie eksploracyjne jest najbardziej przydatne w przypadku niepełnych lub niewłaściwie sporządzonych specyfikacji bądź pod presją czasu.

Może być uzupełnieniem innych bardziej formalnych technik testowania.

**K2.4.4.3. Testowanie w Oparciu o Listę Kontrolną**

W testowaniu w oparciu o listę kontrolną testerzy projektują, implementują i uruchamiają testy tak aby pokryć warunki testowe wymienione w liście kontrolnej.

Listy kontrolne można opracowywać na podstawie własnego doświadczenia =, danych dotyczących potencjalnych defektów i awarii, oczekiwań użytkowników.

**V. ZARZĄDZANIE TESTAMI**

**Ogólnie 5.1. organizacja Testów**

**K2.5.1.1. Niezależne Testowanie:**

Zadania związane z testowaniem mogą wykonywać zarówno osoby pełniące konkretne role w procesie testowym jak i osoby pełniące inne role(np.: klienci)

Niektóre poziomy niezależności testowania:

*-brak niezależnych testerów??*

-niezależni deweloperzy lub testerzy pracujący w zespole deweloperskim lub projektowym

-niezależny zespół testowy działający w ramach organizacji

-niezależni testerzy będący przedstawicielami działów biznesowych lub społeczności użytkowników

-niezależni testerzy spoza organizacji pracujących u klienta

Sposób zapewniania niezależności testowania zależy od wybranego modelu cyklu życia oprogramowania.

Potencjalne korzyści wynikające z niezależności testowania:

-prawdopodobieństwo wykrycia przez niezależnych testerów innego rodzaju awarii niż te wykryte przez deweloperów

-możliwość zweryfikowania lub obalenia przez niezależnych testerów założeń przyjętych przez interesariuszy

Potencjalne wady:

-izolacja testerów od zespołu deweloperów może prowadzić do braku komunikacji

-niebezpieczeństwo utraty przez programistów poczucia odpowiedzialności za jakość

-niebezpieczeństwo traktowania testerów jako zło konieczne

-ryzyko że niezależni testerzy nie będą dysponowali ważnymi informacjami

Wiele organizacji i jest w stanie osiągnąć korzyści z niezależnego testowania jednocześnie unikając wpisanych w nie wad.

**K1. 5.1.2. Zadania Kierownika Testów i Testera:**

Czynności i zadania wykonywane przez osoby pełniące obie role zależą od kontekstu projektu i produktu, umiejętności konkretnych osób oraz specyfikacji organizacji.

Kierownik testów ponosi ogólną odpowiedzialność za proces testowy i sprawne kierowanie czynnościami związanymi z testowaniem. Ponadto w wielu projektach kierownik testów może mieć pod sobą kilka zespołów testowych z którymi kierują liderzy.

Zakres obowiązków kierownika:

-opracowywanie lub dokonywanie przeglądu strategii testów i polityki testów

-planowanie testów w tym uwzględnienie kontekstu

-sporządzanie i aktualizowanie planu testów

-koordynowanie realizacji testów i planu testów

-prezentowanie  punktu widzenia testerów

-inicjowanie procesów analizy projektowania, implementacji i wykonania testów

-przygotowanie i dostarczanie raportu z postępu testów

-dostosowanie planu testów do rezultatów

-podejmowanie decyzji o implementacji środowisk testowych

Zadania testera to między innymi:

-dokonywanie przeglądu planów testów i uczestniczenie w ich opracowaniu

-analizowanie, dokonywanie przeglądu i opracowanie wymagań

-identyfikowanie i dokumentowanie warunków testowych

-projektowanie konfigurowanie i weryfikowanie środowisk testowych

-projektowanie i implementowanie przypadków testowych

-przygotowywanie i pozyskiwanie danych testowych

-automatyzacja testowania w zależności od potrzeb

Na poziomie testowania modułowego i integracji modułów testerami są często programiści,

Na poziomie testowania akceptacyjnego analitycy biznesowi, eksperci merytoryczni i użytkownicy,

Na poziomie testowania systemowego i integracji systemów -członkowie niezależnego zespołu testowego

Na poziomie **produkcyjnych testów akceptacyjnych** - operatorzy lub administratorzy systemu.

**5.2-Planowanie i Szacowanie Testów:**

**K2.5.2.1. Cel i Treść Planu Testów:** Plan testów przedstawia w sposób skrótowy czynności testowe wykonywane w ramach projektów **wytwarzania i pielęgnacji oprogramowania.**

Planowanie testów to czynności o charakterze ciągłym która jest wykonywana przez cały cykl życia oprogramowania.

Na proces planowania wpływają:

-polityka i strategia testów obowiązująca w danej organizacji

-cykle życia i metody wytwarzania oprogramowania

-zakres testowania

-cele

-czynniki ryzyka ograniczenia

-krytyczność

-testowalność oraz dostępność zasobów

Planowanie testów może obejmować następujące działania:

-określanie zakresu i celów testowania oraz związanego z nim ryzyka

-określenie ogólnego podejścia do testowania

-integrowanie i koordynowanie czynności testowych

-podejmowanie decyzji dotyczących tego co należy przetestować

-planowanie czynności związanych z analizą  projektowaniem, implementacją, wykonywaniem i oceną testów

-określanie budżetu potrzebnego na czynności testowe

**K2.5.2.2. Strategia Testów i Podejście do Testowania:**

Do typowych strategii testów zalicza się:

-Strategia analityczna. Opiera się na analizie określonego czynnika.

-Strategia Oparta na Modelu: testy projektuje się na podstawie konkretnego wymaganego aspektu produktu

Strategia Metodyczna: systematyczne stosowanie z góry określonego zbioru testów lub warunków testowych

Strategia Zgodna z Procesem: polega na analizie, projektowaniu i implementacji przypadków testowych na podstawie zewnętrznych  reguł i standardów

Strategia Kierowania. Podstawą tej strategii są przede wszystkim porady wskazówki i instrukcje interesariuszy, ekspertów

Strategia Testowa Minimalizująca Regresję: To strategia motywowana chęcią uniknięcia regresji już istniejących funkcjonalności

Strategia Reaktywna – W przypadku tej strategii testowanie jest ukierunkowane bardziej na zdarzenia niż na realizację ustalonego z góry planu

Właściwa strategia testowa jest zazwyczaj kombinacją powyższych strategii.

**K2.5.2.3. Kryteria Wejścia i Wyjścia**

W celu zapewnienia skutecznego nadzoru nad jakością oprogramowania i testowania

zaleca się określenie kryteriów wskazujących kiedy należy rozpocząć daną czynność testowa oraz kiedy można ją uznać za zakończoną.

Typowe kryteria wejścia to między innymi:

-dostępność testowanych wymagań

-dostępność elementów testowych

-dostępność środowiska testowego

-dostępność niezbędnych narzędzi testowych

-dostępność danych testowych i innych niezbędnych zasobów

Typowe kryteria wyjścia to:

-zakończenie wykonywania testów

-osiągnięcie odpowiedniego poziomu pokrycia

-nie przekroczenie uzgodnionego limitu nieustających defektów

-uzyskanie wystarczająco wysokich wskaźników niezawodności

Nawet w przypadku niespełnienia kryteriów wyjścia czynności testowe mogą zostać skrócone z powodu wykorzystania całego budżetu upływu czasu lub presji z wprowadzaniem na rynek

**K2.5.2.4. Harmonogram Wyrównywania Testów**

Harmonogram powinien uwzględniać takie czynniki jak: priorytety, zależności, konieczności wykonania testów potwierdzających regresji oraz najbardziej efektywną kolejność wykonywania testów.

O ile jest to możliwe testy powinny być wykonywane w kolejności

Jeżeli występują wzajemne zależności między kilkoma przypadkami testowymi należy ustalić ich od priorytetów.

**K2.5.2.5. Czynniki Wpływające na Pracochłonność Testowania.**

Jednym z elementów zarządzania testami jest szacowanie pracochłonności czyli przewidywanie nakładów pracy związanych z testowaniem które są niezbędne do osiągnięcia celów testowania.

Charakterystyka produktu, charakterystyka procesu wytwarzania , czynniki ludzkie

Charakterystyka produktu:

-czynniki ryzyka związane z produktem

-jakość specyfikacji

-wielkość produktu

-złożoność dziedziny produktu

-wymagania dotyczące charakterystyk jakościowych

-wymagany poziom szczegółowości dokumentacji testów

-wymagania dotyczące zgodności z przepisami prawa

Charakterystyka procesu wytwarzania oprogramowania:

-stabilność i dojrzałość organizacji

-stosowany model wytwarzania oprogramowania

-podejście do testowania

-używane narzędzia

-proces testowy

-presja czasu

Czynniki Ludzkie:

-umiejętność i doświadczenie zaangażowanych osób

-umiejętność współpracy zespołu i umiejętności kierownika

Rezultaty testów:

-liczba i ważność wykrytych defektów

-liczba wymaganych poprawek

**K2.5.2.6.Techniki Szacowania Testów:**

Dwie najczęściej stosowane techniki do oszacowania testów:

**-technika oparta na miarach**- szacowanie pracochłonności testowania na podstawie miar z wcześniejszych projektów

Przykład: wykresy spalania – rejestrowanie i raportowanie przy ich użyciu nakłady pracy są wykorzystywane do określania prędkości zespołu czyli ilości pracy jaką zespół może wykonać w następnej iteracji.

**-technika ekspercka –** szacowanie pracochłonności testowania na podstawie doświadczenia osób odpowiedzialnych za zadania związane z testowaniem

Przykład: modele usuwania defektów które przewidują rejestrowanie i raportowanie liczby defektów oraz czasu niezbędnego do ich usunięcia.

**5.3.Ogólnie -  Monitorowanie Testów i Nadzór nad Testami**

Celem monitorowania testów jest gromadzenie i udostępnianie informacji pozwalających uzyskać wgląd w przebieg czynności testowych.

Informacje objęte monitorowaniem mogą być zbierane ręcznie lub automatycznie powinny być wykorzystywane do oceny postępu testów.

Nadzór nad testami obejmuje wszelkie działania zarządcze i korygujące podejmowane na podstawie zgromadzonych w raportach informacji.

**K1.5.3.1.Miary Stosowane w Testowaniu:**

Podczas wykonywania testów i po ich zakończeniu można zbierać miary pozwalające oszacować:

-postęp realizacji harmonogramu i budżetu

-bieżącą jakość przedmiotu testów

-adekwatność wybranego podejścia do testowania

-skuteczność czynności testowych z punktu widzenia realizacji celów

Powszechnie stosowane są następujące miary dotyczące testów:

**-poziom wykonania przypadków testowych**

**-informacje o defektach**

-**pokrycie testowe wymagań**

-koszty testowania

-procent wykonania zaplanowanych prac związanych z przygotowaniem przypadków testowych

-procent wykonania zaplanowanych z przygotowaniem środowiska testowego

**K2.5.3.2. Cel, Treść i Odbiorcy Raportów z Testów:**

Raporty z testów służą do podsumowania i przekazywania informacji na temat czynności testowych zarówno w trakcie ich wykonania jak i po ich zakończeniu.

Na etapie monitorowania i nadzoru nad testami kierownik testów  sporządza raporty o postępie interesariuszy oraz sumaryczny raport z testów.

Typowy raport z postępu testów może zawierać:

-status czynności testowych i postępu realizacji planu testów

-czynników zakłócających wykonywanie prac

-testów zaplanowanych w następnym okresie raportowania

-jakości przedmiotu testów

Po spełnieniu kryteriów wyjścia kierownik testów sporządza sumaryczny raport z testów który zawiera między innymi:

-podsumowanie wykonanych testów

-informacje na temat zdarzeń

-informacje o odstępstwach od planu

-informacje o statusie testowania i jakości produktu

--miary które blokowały lub nadal blokują przebieg testów

-informacje na temat ryzyka

-informacje o wytwarzanych produktach związanych z testowaniem

Treść raportu zależy od projektu, wymagań organizacyjnych i cyklu życia oprogramowania.

Raporty z testów muszą być dostosowane zarówno do kontekstu projektu jak i do potrzeb docelowych odbiorców. Typ i ilość informacji dla odbiorców może być inny i rózny od tych zamieszczonych w raporcie sumarycznym dla kierownictwa.

**5.4 Zarządzanie Konfiguracją:**

Celem zarządzania konfiguracją jest zapewnienie i utrzymanie integralności modułu lub systemu i testaliów oraz wzajemnych relacji między nimi przez cały cykl życia projektu.

W kontekście testowania zarządzania konfiguracji:

-wszystkie przedmioty testów zostały zidentyfikowane

-wszystkie testalia zostały zidentyfikowane

-wszystkie zidentyfikowane dokumenty i elementy oprogramowania były przywołane w sposób jednoznaczny w dokumentacji testaliów.

Procedury zarządzania konfiguracją wraz z niezbędną infrastruktura należy zidentyfikować i zaimplementować na etapie planowania testów.

**5.5.Czynniki Ryzyka a Testowanie:**

**K1.5.5.1. Definicja Ryzyka.**

Ryzyko jest możliwością wystąpienia w przyszłości zdarzenia o niepożądanych konsekwencjach.

O poziomie ryzyka decyduje prawdopodobieństwo wystąpienia niekorzystnego zdarzenia oraz jego wpływ

**K2.5.5.2. Czynniki Ryzyka Produktowego i Projektowego:**

Ryzyko produktowe występuje wszędzie tam gdzie produkt pracy może nie zaspokoić uzasadnionych potrzeb użytkowników czy interesariuszy.

Przykładami czynników ryzyka produktowego mogą być następujące problemy:

-niewykonywalne przez oprogramowanie zakładanych funkcji zgodnie z specyfikacją.

-niedostateczne spełnienie przez architekturę systemu określonych wymagań niefunkcjonalnych

-niepoprawne wykonywanie konkretnych obliczeń w niektórych okolicznościach.

-błędy w kodzie struktury sterowania pętlą

-zbyt długi czas odpowiedzi w systemie transakcyjnym wysokiej wydajności

-niezgodność informacji zwrotnych

Ryzyko projektowe obejmuje sytuacje których zaistnienie może mieć negatywny wpływ na możliwość osiągnięcia celów projektu. Przykłady ryzyka projektowego to:

-problemy związane z projektem

-problemy organizacyjne

-problemy natury politycznej

-problemy techniczne

-problemy związane z dostawcami

Ryzyko projektowe może dotyczyć zarówno czynności związanych z wytwarzaniem oprogramowania jak i jego testowaniem.

**K2.5.5.3. Testowanie Oparte na Ryzyku a Jakość Produktu**

Wiedzę na temat ryzyka można wykorzystać do odpowiedniego ukierunkowania działań wykorzystywanych podczas testowania. Na podstawie ryzyka można podejmować decyzje co do tego gdzie i kiedy należy rozpocząć testowanie.

Podejście do testowania oparte na ryzyku umożliwia prewencyjne obniżanie poziomu ryzyka produktowego.

Jednym z elementów tego podejścia jest analiza ryzyka produktowego która obejmuje identyfikowanie czynników tego typu ryzyka oraz szacowanie prawdopodobieństwa wystąpienia ich.

Podejście oparte na ryzyku przeprowadzonej analizy produktowego umożliwia:

-wskazanie odpowiednich technik testowania

-określenie konkretnych typów które należy wykonać

-ustalenie priorytetów testowanie w sposób sprzyjający jak najwcześniejszemu wykryciu

-ustalenie czy w celu zmniejszania ryzyka należy wykonać inne czynności niezwiązane z testowaniem

Aby zminimalizować prawdopodobieństwo awarii produktu w ramach zarządzania wyzyskiem wykonuje się usystematyzuje czynności obejmujące:

-analizowanie potencjalnych problemów i regularne dokonywanie ich ponownej oceny

-ustalenie które czynniki ryzyka są istotne i wymagają podjęcia działań

-podejmowanie działań mających na celu załagodzenie ryzyka

-tworzenie planów awaryjnych na wypadek faktycznego wystąpienia określonych czynników ryzyka

Ponadto testowanie pozawal zidentyfikować nowe czynniki ryzyka wskazać czynniki wymagające złagodzenia oraz zmniejszyć niepewności związane z ryzykiem.

**5.6.Zarządzanie Defektami:**

Jednym z celów testowania jest znajdowanie defektów w związku z czym wszystkie znalezione defekty należy zalogować. Sposób zgłoszenia defektu zależy od kontekstu testowania danego modułu lub systemu, poziomu testów oraz wybranego modelu wytwarzania oprogramowania.

W procesie zarządzania defektami może się okazać że w niektórych raportach opisane zostały rezultaty fałszywie pozytywne a nie rzeczywiste awarie spowodowane defektami.

Przykładem takiej sytuacji może być niezaliczenie testu na skutek przerwania połączenia sieciowego.

W związku z tym testerzy powinni starać się ograniczyć do minimum liczbę rezultatów fałszywie pozytywnych które są zgłaszane jako defekty.

Typowe raporty o defekcie ma na celu:

-dostarczenie deweloperom i innym interesariuszom informacji na temat wszelkich zaistniałych zdarzeń

-umożliwienie kierownikom testów śledzenia jakości produktu pracy i wpływu na postęp testowania

-przedstawienie sugestii dotyczących usprawnienia procesów wytwarzania i testowego

Raport o defekcie przedkładany podczas testowania dynamicznego zawiera zwykle następujące szczegółowe informacje:

-identyfikator

-tytuł i krótkie podsumowanie zgłaszanego defektu

-data raportu o defekcie zgłaszająca jednostka i autor zgłoszenia

-identyfikacja elementu testowego

-faza cyklu życia oprogramowania w której zaobserwowano defekt

-opis defektu umożliwiający jego odtworzenie i usunięcie

-oczekiwane i rzeczywiste rezultaty

-zakres i stopień wpływu

-priorytety usunięcia defektu

-staus raportu o defekcie

-wnioski zalecenia i zgody

-kwestie globalne

-historia zmian

-odwołania do innych elementów

Niektóre z powyższych szczegółowych informacji mogą zostać automatycznie uwzględnione lub objęte zarządzanie w przypadku korzystania z narzędzi do zarządzania defektami.

VI. NARZĘDZIA WSPOMAGAJĄCE TESTOWANIE

**K2.6.1.Uwarunkowania Związane z Narzędziami Testowymi:**

Narzędzia testowe służą do wspierania jednej lub wielu czynności testowych. Dostępne są między innymi następujące narzędzia

-narzędzia używane bezpośrednio podczas testowania, takie jak narzędzia do wykonywania testów i narzędzia do przygotowania danych testowych

-narzędzia ułatwiające zarządzanie wymaganiami

-narzędzia służące do analizy i oceny

-inne narzędzia wspomagające testowanie

**K2.6.1.1.Klasyfikacja Narzędzi Testowych:**

Zależnie od kontekstu narzędzi testowe mogą służyć realizacji jednego lub kilku celów:

-zwiększenie efektywności wykonywania czynności testowych poprzez zautomatyzowanie powtarzających się zadań

-podniesienie wydajności czynności testowych poprzez wspieranie czynności manualnych w trakcie całego procesu testowego

-podniesienie jakości wykonywanych czynności testowych poprzez zwiększenie spójności testowania

-zautomatyzowanie czynności których nie można wykonać ręcznie

-zwiększenie niezawodności testowania

Narzędzia mogą być klasyfikowane według kilku kryteriów

Niektóre narzędzia obsługują tylko lub głównie jedną czynność natomiast inne pozwalają wykonywać kilka czynności.

Niektóre typy narzędzi testowych mogą mieć charakter inwazyjny co oznacza że mogą wpływać na rzeczywisty wynik testu.

Niektóre narzędzia posiadają funkcje które są bardziej odpowiednie dla deweloperów.

Narzędzia wspomagające zarządzanie testowaniem i testaliami: mogą obejmować swoimi działaniem wszystkei czynności testowe wykonywane przez cały cykl życia oprogramowania.

Narzędzia wspomagające testowanie statyczne

Narzędzia wspomagające projektowanie i implementacja testów

Narzędzia wspomagające wykonywanie i logowanie testów

Narzędzia wspomagające pomiar wydajności i analizę dynamiczną: są niezbędne do wykonania czynności związanych z testowaniem wydajnościowym i obciążeniowym

Narzędzia wspomagające wyspecjalizowane czynności testowe: są pomocne w testowaniu charakterystyk niefunkcjonalnych

**K1.6.1.2 Korzyści i czynniki ryzyka związane z automatyzacją testowania:**

Sam zakup narzędzia nie gwarantuje jeszcze sukcesu.

Potencjalne korzyści wynikające z zastosowania narzędzi wspomagających testowanie:

-ograniczenie powtarzalnych czynności wykonywanych ręcznie

-zwiększenie spójności i powtarzalności

-bardziej obiektywna ocena rezultatów testów

-łatwiejszy dostęp do informacji testowania

Potencjalne ryzyka związane z zastosowaniem narzędzi wspomagających:

-nierealistyczne oczekiwania wobec narzędzia

-niedoszacowanie czaso i pracochłonności oraz kosztów początkowego wdrożenia narzędzia

-niedoszacowanie nakładów pracy

-nadmierne uzależnienie od narzędzia

-zaniedbanie kontroli wersji produktów pracy

-możliwość zakończenia przez dostawcę działalności

**K1.6.1.3 Szczególne uwarunkowania związane z narzędziami do wykonywania testów i zarządzania testami:**

Aby sprawnie zaimplementować narzędzia do wykonywania testów i zarządzania testami należy uwzględnić kilka czynników:

**Narzędzia do wykonywania testów:** umożliwiają uruchamianie przedmiotu testów i zarządzania testami w organizacji, należy uwzględnić kilka istotnych czynników:

-Rejestrowanie działań wykonywanych ręcznie

-Podejście do testowania sterowanego danymi

-Testowanie oparte na słowach kluczowych

Narzędzia do testowania opartego na modelu pozwalają zarejestrować specyfikację funkcjonalną w postaci modelu takiego jak diagram aktywności.

**Narzędzia do zarządzania testami:** muszą często komunikować się z innymi narzędziami lub arkuszami kalkulacyjnymi: wcelu generowani użyteczności, utrzymania możliwości śledzenia, tworzenia powiązań z informacjiami o wersjach przedmiotów testów.

6.2 Skuteczne korzystanie z narzędzi:

**K1.6.2.1.Główne zasady wyboru narzędzi:**

Najważniejsze kwestie należy wziąć pod uwagę w wyborze narzędzia dla organizacji to:

-dokonywanie oceny dojrzałości własnej organizacji jej mocnych i słabych stron

-wskazanie możliwości udoskonalenia procesu testowego

-zapoznanie się z technologiami stosowanymi w ramach przedmiotu testów

-dokonanie oceny dostawcy

-zidentyfikowanie wewnętrznych wymagań

-uwzględnienie zalet i wad poszczególnych modeli licencyjnych

-oszacowanie stosunku kosztów do korzyści

**K1.6.2.2. Projekty pilotażowe związane z wprowadzeniem narzędzia w organizacji.**

Po dokonaniu wyboru narzędzia i pomyślnym przeprowadzeniu dowodu słuszności można rozpocząć wprowadzenie tego narzędzia w organizacji.

Pierwszym etapem tego procesu jest zwykle projekt pilotażowy który ma na celu:

-szczegółowe zapoznanie się z narzędziem

-ustalenie na ile dane narzędzie jest dopasowane do istniejących procesów

-podjęcie decyzji w zakresie standardowych sposobów użytkowania

-oszacowanie czy spodziewane korzyści będzie można osiągnąć odpowiednim kosztem

**K1.6.2.3.Czynniki sukcesu związane z narzędziami:**

Wśród czynników sukcesu związanych z obsługą narzędzi w organizacji należy wymienić:

-dostosowanie i udoskonalanie procesów pod kątem współpracy z narzędziem

-zapewnienie użytkownikom narzędzia odpowiednich szkoleń

-wprowadzenie mechanizmu zbierania informacji

-monitorowanie użytkowania narzędzia

-zapewnianie użytkownikom narzędzia niezbędnej pomocy technicznej

-zbieranie doświadczenia od wszystkich użytkowników

Należy również zadbać o integrację techniczną i organizacyjną narzędzia w każdym cyklu życia oprogramowania.