Metodyka testowania systemów wchodzących w skład SIG

INNOWACYJNA GOSPODARKA
NARODOWA STRATEGIA SPÓJNOŚCI

GŁÓWNY URZĄD GEODEZJI I KARTOGRAFII

UNIA EUROPEJSKA
EUROPEJSKI FUNDUSZ
ROZWOJU REGIONALNEGO

Spis treści

Spis rysunków

Spis tabel

1 Wprowadzenie

1.1 Cel dokumentu

Celem dokumentu jest przedstawienie wytycznych metodycznych dla procesu testowania w ramach procesu wytwórczego realizowanego w SIG. Dokument, zwany dalej Metodyką, uzupełnia i doprecyzowuje istniejące w SIG standardy, w szczególności Podręcznik SIG.

1.2 Zakres dokumentu

Dokument opisuje proces testowania oprogramowania. Dokument składa się a następujących rozdziałów:

- Rozdział 1 Wprowadzenie – rozdział zawiera opis celu dokumentu, słownik pojęć oraz wykaz dokumentów powiązanych;
- Rozdział **Błąd! Nie można odnaleźć źródła odwołania.** Testy w ramach procesu wytwórczego – rozdział przedstawia rodzaje testów realizowanych w ramach procesu wytwórczego;
- Rozdział 3 Produkty procesu testowania – rozdział przedstawia i opisuje produkty do wytworzenia na użytek przeprowadzenia testów;
- Rozdział 4 Procedura testów – rozdział wskazuje obowiązkowe do przestrzegania procedury;
- Rozdział 5 Środowisko testów – rozdział opisuje proces i charakterystykę środowisk infrastrukturalnych koniecznych do przeprowadzenia testów;
- Rozdział 6 Dane testowe – rozdział opisuje proces przygotowania danych testowych koniecznych na użytek przeprowadzenia testów;
- Rozdział 7 Podejście TDD – rozdział opisuje metodykę Test Driven Development;
- Rozdział 8 Inne zagadnienia związane z procesem testowania.

1.3 Zastosowane konwencje, słownik użytych pojęć

Termin	Definicja
Testowanie (oprogramowania)	Element procesu wytwarzania oprogramowania. Jego celem jest badanie, czy dane oprogramowanie posiada określone cechy określone w specyfikacji (i wyrażone w niej np. poprzez

	wymagania). Testowanie dostarcza także wiedzy o ryzyku odbioru systemu.
Przypadek testowy	Przypadek testowy to opis pewnej liniowej ścieżki interakcji z systemem, która, przy założeniu wykorzystania określonych danych i przy określonym stanie początkowym systemu i jego otoczenia (tzw. warunkach początkowych), jednoznacznie doprowadza do sekwencji określonych, obserwowalnych stanów systemu.
Scenariusz testowy	Scenariusze i przypadki testowe są działaniami zapisanymi w Planie testów i przeznaczonymi do wykonania w ramach testów weryfikujących spełnienie wymagań stawianych danemu rozwiązaniu.
Wymaganie funkcjonalne	Cecha jakościowa oprogramowania zgodna z taksonomią normy ISO-9126 lub norm jej równoważnych określona w OPZ dotyczącym systemu
Wymaganie pozafunkcjonalne	

Tabela 1. Pojęcia i skróty

1.4 Problemy związane z procesem testowania

Poniżej przedstawione zostały główne, zidentyfikowane problemy związane z procesem testowania systemów wchodzących w skład SIG:

1. Brak danych testowych;
2. Niedostępność systemów zewnętrznych;
3. Niedoszacowanie czasu przeprowadzenia testów;
4. Brak gotowości oprogramowania;
5. Brak kompletności przekazywanego Planu testów;
6. Problemy z testowaniem integracji z systemem Geoportal;
7. Problemy z testowaniem wydajności systemu;
8. Brak określenia wymaganych rodzajów testów;
9. Brak wytycznych do opracowywania scenariuszy testowych;
10. Nie wykorzystywanie testów automatycznych do testów regresji.

1.5 Dokumenty powiązane

Tam gdzie nie zaznaczono inaczej Metodyka posługuje się terminologią i konstrukcjami pojęciowymi opisanymi w:

- Podręcznik SIG;
- Information Technology Infrastructure Library (ITIL), https://www.axelos.com/best-practice-solutions/itil;
- Unified Modeling Language (UML), http://www.omg.org/spec/UML/2.5/;

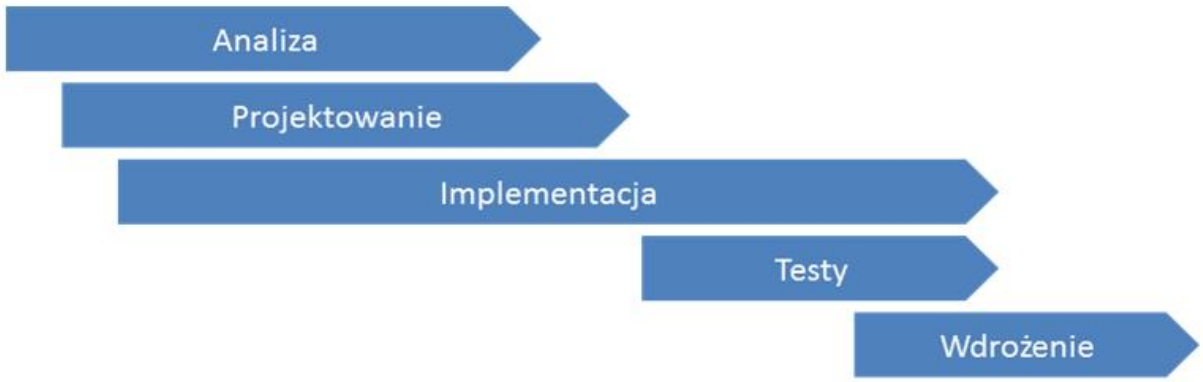
- RFC 2119 (Key words for use in RFCs to Indicate Requirement Levels) http://www.ietf.org/rfc/rfc2119.txt.

2 Testy w ramach procesu wytwórczego

Obecnie w ramach SIG przyjęto proces wytwórczy w kształcie opisanym poniższym rysunkiem:

Rysunek 1. Proces wytwórczy w ramach SIG

Testy są elementem podprocesu Dostarczania, który przedstawia Rysunek 2.



Rysunek 2. Umiejscowienie testów w ramach podprocesu dostarczania

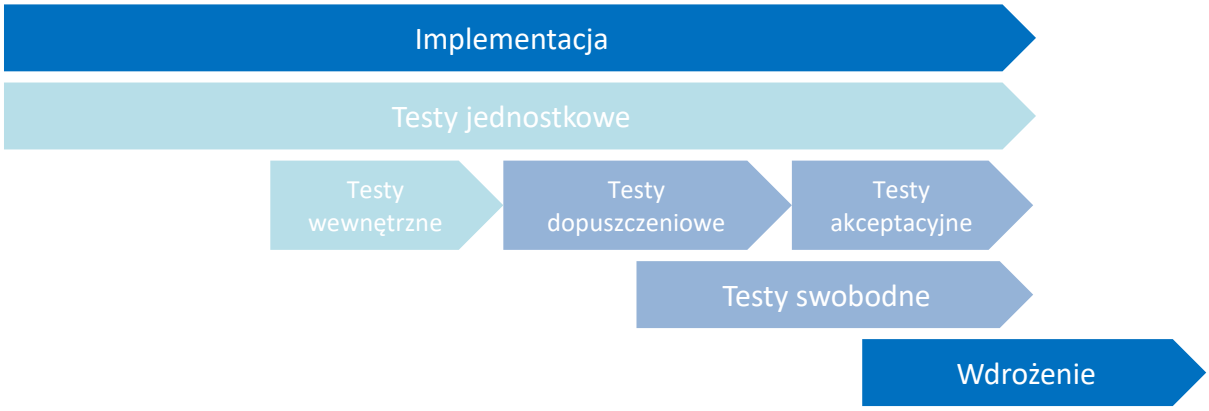
Powyższy schemat dla podprocesu Dostarczania przedstawia przebieg całego procesu lub jednej z jego iteracji (w przypadku przyjęcia podziału przedsięwzięcia na iteracje). Należy zwrócić uwagę na nachodzenie na siebie poszczególnych podprocesów. Uwzględniając różnorodność projektów, istotne jest zachowanie kolejności rozpoczynania i kończenia poszczególnych procesów. Proces Testów rozpoczyna się w trakcie trwania procesu Implementacji i nie powinien zaczynać się przed zakończeniem Procesu Projektowania (nie dotyczy to testów jednostkowych, które powinny być przygotowywane od początku procesu Implementacji). Proces Testów kończy się wraz z procesem Implementacji, a może trwać w trakcie procesu Wdrożenia.

W przypadku zastosowania w danym przedsięwzięciu iteracyjnego cyklu wytwórczego, elementy podprocesu Dostarczania powtarzane są w kolejnych iteracjach dla określonych oczekiwanych cech systemu – zbiorów wymagań lub funkcjonalności. Podobnie powtarzana, wraz z innymi elementami podprocesu Dostarczania, jest faza testowania. Zakres kolejnych iteracji testów powinien być przyrostowy, tzn. w określonej iteracji testowane są wszystkie elementy przetestowane w poprzednich iteracjach oraz nowe, wytworzone w obecnej iteracji. Powtarzanie raz wykonanych testów (scenariuszy lub przypadków testowych) może być uzupełniane o nowe elementy (np. wynikające z rozwoju istniejących funkcjonalności), a więc testy cech systemu z poprzednich iteracji nie muszą być jedynie testami regresyjnymi.

W procesie testów przeprowadzana jest weryfikacja oprogramowania. Rozróżnia się następujące rodzaje testów, które powinny zostać wykonane:

- Testy wewnętrzne,
- Testy dopuszczeniowe,
- Testy akceptacyjne,
- Testy jednostkowe,
- Testy swobodne.

Poniższy rysunek umiejscawia rodzaje testów w ramach faz towarzyszących testom podprocesów procesu wytwórczego oraz prezentuje zależności początek - koniec między rodzajami testów.



Rysunek 3. Umiejscowienie rodzajów faz w ramach procesu wytwórczego

2.1 Testy jednostkowe

Testy jednostkowe (ang. unit tests) zorientowane są na weryfikację określonych elementów oprogramowania (na różnym poziomie ziarnistości – metod, klas, modułów). Testy jednostkowe

towarzyszą podprocesowi implementacji (patrz: Rysunek 3), który powinien przebiegać zgodnie z zasadami podejścia TDD (ang. Test-Driven Development) – patrz rozdział Podejście TDD.

Realizacja testów jednostkowych leży po stronie Wykonawcy, Zamawiający obserwuje jedynie wyniki realizowanych testów.

2.2 Testy wewnętrzne

Testy wewnętrzne obejmują weryfikację rozwiązania pod względem realizacji przez nie wymagań funkcjonalnych i są prowadzone przez Wykonawcę (bez udziału Zamawiającego) w środowisku Wykonawcy. Wykonawca przekazuje Raport z testów wewnętrznych przed rozpoczęciem tury testów dopuszczeniowych.

2.3 Testy dopuszczeniowe

Testy dopuszczeniowe obejmują weryfikację rozwiązania pod względem realizacji przez nie wymagań funkcjonalnych. Testy dopuszczeniowe są przeprowadzane przez Wykonawcę w obecności Zamawiającego lub są przeprowadzane przez Zamawiającego.

2.4 Testy akceptacyjne

Testy akceptacyjne obejmują weryfikację oprogramowania w zakresie funkcjonalności oraz wymagań pozafunkcjonalnych poprzez wykonanie testów specyficznych dla testowania poszczególnych wymagań pozafunkcjonalnych, w tym uwzględnienia testów wydajnościowych (w tym przeciążeniowych), bezpieczeństwa i automatycznych. W ramach testów wydajnościowych powinny być przeprowadzone testy przeciążeniowe. Testowanie przeciążeniowe jest prowadzone w celu oceny zachowania systemu w sytuacji maksymalnego obciążenia. Testy obciążeniowe powinny obejmować badanie zachowania systemu przy dużej liczbie jednocześnie działających użytkowników oraz wywoływania wielu zapytań jednocześnie, przez dłuższy okres czasu. Kryteria akceptacji testów przeciążeniowych powinny być określone w OPZ lub uzgodnione przy opracowywaniu strategii testów. Testy automatyczne mogą być wykorzystywane przy testach regresji. Wdrożenie może rozpocząć się przy cząstkowych wynikach testów akceptacyjnych, jednak do jego skutecznego wykonania konieczne jest zakończenie testów akceptacyjnych.

Testy akceptacyjne są przeprowadzane przez Wykonawcę w obecności Zamawiającego lub są przeprowadzane przez Zamawiającego.

2.5 Testy swobodne

Testy swobodne przeprowadzane są przez Zamawiającego, najczęściej bez obecności Wykonawcy. Testy swobodne nie wymagają przygotowania dodatkowych produktów (patrz: Produkty procesu testowania). Celem testów swobodnych jest weryfikacja funkcjonalności rozwiązania oraz jego cech pozafunkcjonalnych. Rezultaty testów swobodnych są przedmiotem osobnych ustaleń projektowych.

Należy zwrócić uwagę na stabilność testowanego oprogramowania w sytuacji, gdy testy swobodne realizowane są przed zakończeniem testów dopuszczeniowych. W przypadku, gdy w wyniku testów dopuszczeniowych oprogramowanie będzie podlegało modyfikacjom, może zajść potrzeba powtórzenia części lub całości działań wykonanych w ramach testów swobodnych. Dlatego w pewnych przypadkach celowym może być rozpoczęcie testów swobodnych po zakończeniu testów dopuszczeniowych.

Dobór rodzajów testów jest zależny od specyfiki danego projektu, np. w sytuacji realizacji procesu, którego wynikiem jest dostarczenie rozwiązania pilotażowego akceptowalna jest rezygnacja z testów dopuszczeniowych. Wykonanie innych, dodatkowych rodzajów testów, może być przedmiotem ustaleń specyficznych dla konkretnych przedsięwzięć. Przykładem mogą być testy integracyjne, których sposób prowadzenia jest ściśle powiązany z architekturą rozwijanego rozwiązania, zarówno biznesową (np. w zakresie współpracy podmiotów biorących udział w testach), jaki i aplikacyjną czy techniczną (w kontekście sposobu implementacji integracji).

2.6 Wytyczne procedury testowania

W ramach prowadzonych testów wymaga się spełnienia następujących warunków:

1) Pozytywny wynik testów wewnętrznych jest warunkiem koniecznym do rozpoczęcia testów dopuszczeniowych;

2) Pozytywny wynik testów dopuszczeniowych jest warunkiem koniecznym do rozpoczęcia testów akceptacyjnych;

3) Pozytywny wynik testów akceptacyjnych jest warunkiem koniecznym do odbioru oprogramowania i wykonania wdrożenia masowego na środowisku produkcyjnym;

4) Testy akceptacyjne są prowadzone na jednej wersji oprogramowania tzn.:

- cała tura testów akceptacyjnych musi być przeprowadzona na jednej wersji oprogramowania;
- po rozpoczęciu testów akceptacyjnych dopuszcza się implementację, która polega na usunięciu błędów kosmetycznych;

W szczególnych przypadkach, za zgodą Zamawiającego, dopuszcza się odstępstwo od powyższej procedury.

3 Produkty procesu testowania

W ramach procesu testowania powinny powstawać produkty opisane niniejszym rozdziałem.

3.1 Strategia testowania

Strategia testowania powstaje w postaci dokumentu tekstowego zgodnego z szablonem dostarczonym przez Zamawiającego. Celem Strategii testowania jest udokumentowanie decyzji dotyczących procesu testowania w danym przedsięwzięciu. Strategia testowania doprecyzowuje Metodykę tam, gdzie Metodyka nie przesądza konkretnych rozwiązań lub gdzie specyfika projektu wymaga odpowiednich ustaleń na etapie realizacji projektu. Dokument musi powstać przed zaplanowaniem i wykonaniem testów w projekcie, jednak później może podlegać aktualizacjom.
Strategia testowania jest pierwszym produktem procesu testowania.

Przykładowe elementy jakie muszą znaleźć się w Strategii testowania:

1) W przypadku, gdy rozwiązanie zawiera kilka aplikacji składowych, określenie zakresu testów (zbiorów wymagań z Projektu Funkcjonalnego) weryfikowanych dla każdej z nich;

2) Przedstawienie zakresu faz testów dla projektów prowadzonych etapowo/iteracyjnie;

3) Techniki weryfikacji specyficznych wymagań pozafunkcjonalnych;

4) Zakres i charakter danych testowych;

5) Uproszczenia przyjęte podczas weryfikacji (np. dla rozwiązań multiplatformowych);

6) Opis dodatkowych rodzajów testów, jakie należy przeprowadzić (np. testy integracyjne, testy zgodności ze standardami, testy regresji, testy maksymalnego obciążenia);

7) Definicje i limity kategorii błędów, jeżeli są one inne od zawartych w Metodyce (patrz: Kategorie błędów);

8) Decyzje dotyczące środowiska testów dla testów akceptacyjnych.

Przykład Strategii testowania w określonym przedsięwzięciu stanowi Załącznik 1 do niniejszego dokumentu. „Strategia procesu testowania dla Umowy na Budowę oraz rozwój e-usług i narzędzi w ramach projektów CAPAP, ZSIN Faza II i K-GESUT wraz ze szkoleniami” została przygotowana na podstawie niniejszej Metodyki testowania, a także obejmuje ekstrakt elementów samej Umowy związanych z zagadnieniem testów (będących zobowiązaniem wszystkich jej stron). Należy mieć na uwadze, że powyższy przykład wskazuje modelowe podejście do opracowania Strategii testów w ramach konkretnej Umowy/przedsięwzięcia, może stanowić ukierunkowanie takiego podejścia, natomiast nie oznacza, że każde inne przedsięwzięcie realizowane w ramach niezależnej Umowy powinno skutkować – w ramach własnej Strategii testowania – analogicznym sposobem traktowania oraz zawierać identyczne elementy.

3.2 Plan testów

Plan testów powstaje w postaci dokumentu tekstowego zgodnego z szablonem dostarczonym przez Zamawiającego. Wytworzenie planu testów obowiązuje dla (przynajmniej) testów dopuszczeniowych i akceptacyjnych. Plan testów nie może zostać zaakceptowany, jeśli nie istnieje zaakceptowany dokument Strategii testowania.

Plan testów musi zapewniać przetestowanie wszystkich przypadków użycia i wymagań przypisanych do danego obszaru w Strategii testowania, a także umożliwić sprawdzenie zgodności wytworzonego systemu z Projektem interfejsu użytkownika. Plan testów musi również uwzględniać zapisy z niniejszego dokumentu oraz SOPZ odnośnie testowania, w tym również integracji i migracji danych.

Plan testów zawiera przynajmniej:

1) Zakres testów – określenie obszaru rozwiązania podlegającego pod dany Plan testów. Zakres może być opisany przez określenie zapisów specyfikacji weryfikowanych w ramach danego zestawu testów (np. wymagań, przypadków użycia). W ramach danej fazy nie musi być weryfikowane całe wymaganie, czy przypadek użycia, takie wyłączenia muszą być jednak jednoznacznie określone;

2) Scenariusze testowe (patrz: Scenariusze i przypadki testowe) wraz ze wskazaniem szacowanego czasu wykonywania poszczególnych scenariuszy;

3) Przypadki testowe (patrz: Scenariusze i przypadki testowe);

4) Procedurę zgłaszania błędów;

5) Opis środowiska przeznaczonego do testów (konfiguracja środowiska, wykaz niezbędnych zasobów do przeprowadzenia testów) – patrz: Środowisko testów;

6) Plan realizacji testów – kolejność wykonania testów, ewentualne zaplanowane powtórzenia i ich wariacje, zrównoleglenia, ograniczenia czasowe itd.;

7) Dane testowe – może to być określenie charakteru jakościowego lub ilościowego danych lub (wariant preferowany) dołączenie danych testowych do Planu w formie załączników;

8) Raport z pokrycia testami wymagań oraz przypadków użycia pokazujący mapowanie przypadków testowych na wymagania i przypadki użycia;

9) Opisaną w odpowiedniej dla danego typu testów strukturze metodę weryfikacji wymagań i przeprowadzenia testów w zakresie (element dotyczący testów akceptacyjnych):

    a) Bezpieczeństwa;

    b) Wydajności;

c) Automatyzacji procesu testowania (testy automatyczne).

Elementy opisane w punkcie 9) powyżej mogą stanowić osobne plany testów (np. Plan testów bezpieczeństwa) lub załączniki do planu testów akceptacyjnych. Struktura dla planów testów wskazanych w punkcie 9) powinna powielać strukturę głównego planu testów, w szczególności musi tak czynić w zakresie punktów 1), 4), 6), 7).

Weryfikacja wymagań zapisanych w Planie testów nie musi obejmować wymagań spełnianych jednoznacznie w wyniku zastosowania odpowiedniej architektury, technologii lub dopuszczonych przez Zamawiającego rozwiązań organizacyjnych.

3.3 Raport z testów

Raport z testów powstaje jako produkt wytwarzany w ramach testów i dokumentujący wykonane testy. Raport testów powstaje jako dokument tekstowy zgodny z szablonem dostarczonym przez Zamawiającego. Wytworzenie planu testów obowiązuje dla (przynajmniej) testów dopuszczeniowych i akceptacyjnych.

Raport z testów powinien powstać w oparciu o Plan testów.

Raport z testów zawiera przynajmniej:

1) Zakres testów odpowiadający zapisom Planu testów;

2) Wyniki testów;

3) Plan działań następczych związany z wynikami testów.

4 Procedura testów

Testy nie mogą się rozpocząć wcześniej niż moment akceptacji (odbioru) Planu testów oraz jeśli nie zostało przygotowane środowisko testowe opisane w Planie testów.

W ramach testów wykonywane są zadania testowe (scenariusze i przypadki testowe, działania weryfikujące wymagania pozafunkcjonalne). Po wykonaniu zadań wynikających z Planu testów następuje podsumowanie.

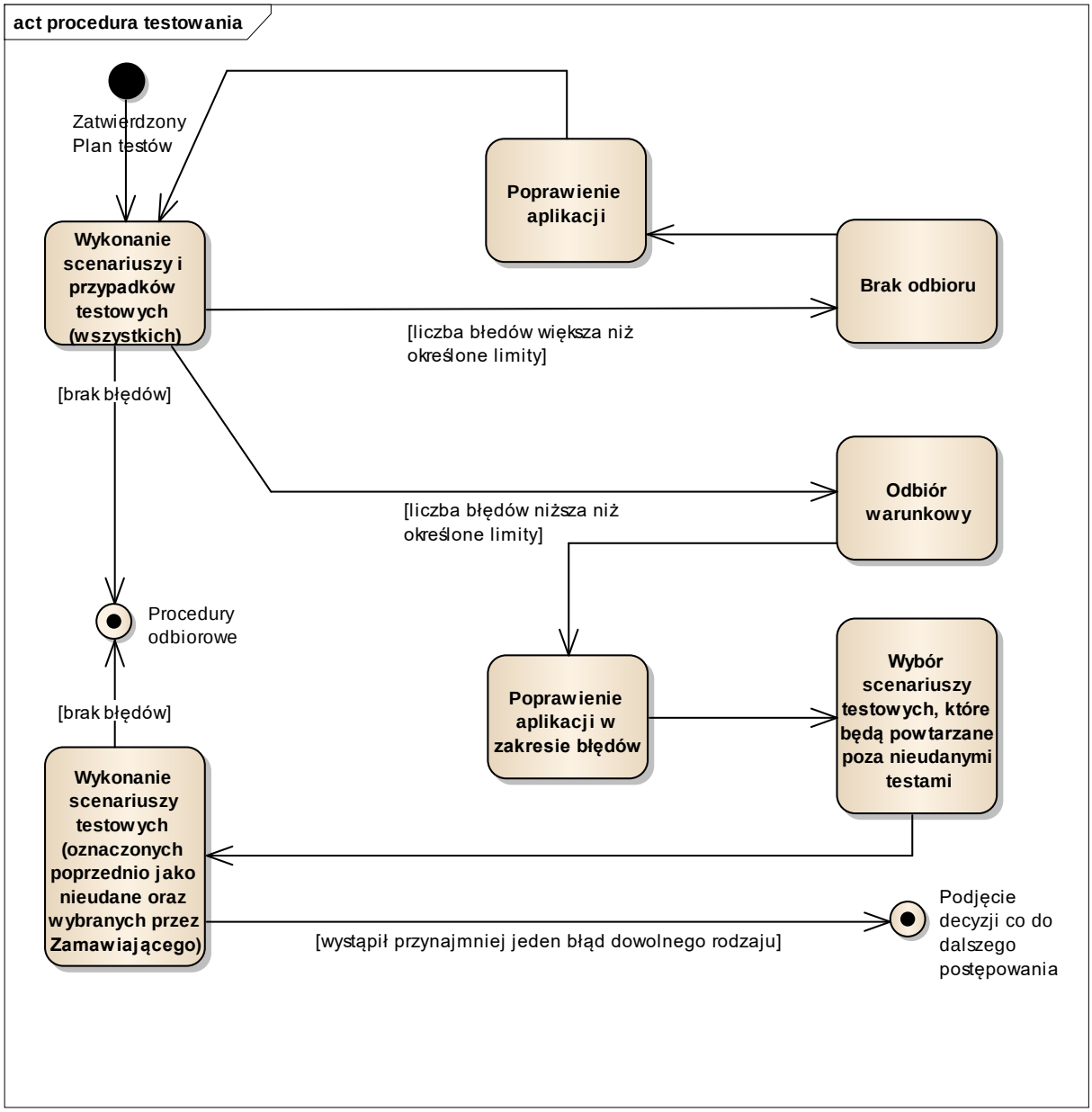
W przypadku braku stwierdzonych błędów następuje przejście do kolejnego etapu. Dla testów dopuszczeniowych jest to zgoda na przystąpienie do fazy testów akceptacyjnych (z ograniczeniami wynikającymi z harmonogramu projektu, gotowości Planu testów i środowiska dla tych testów). W przypadku testów akceptacyjnych w przypadku braku błędów następuje przejście do procedur odbiorowych.

Jeżeli w trakcie testów wystąpiły błędy, istotna jest ich liczba oraz charakter (patrz: Kategorie błędów). W przypadku, gdy błędy nie przekraczają limitów określonych w Planie testów dla zdefiniowanych kategorii błędów wymagana jest poprawa oprogramowania i jego ponowna weryfikacja. Dodatkowo, wybierane są działania z Planu testów, które powinny zostać powtórzone, mimo że wskazały na pozytywną weryfikację oprogramowania. Powtarzane są działania wybrane przez Zamawiającego. Powtarzanie działań ma na celu zapewnienie, że wprowadzane w wyniku błędów poprawki nie powodują błędów w innych elementach oprogramowania. W przypadku wystąpienia jakiegokolwiek błędu podczas poprawianych testów musi zostać podjęta decyzja co do dalszego postępowania.

W przypadku, gdy liczba błędów przekracza określone wcześniej limity następuje poprawienie aplikacji, a następnie dana faza testów rozpoczynana jest od nowa (wykonywane są wszystkie testy zapisane w Planie testów). Należy zwrócić uwagę, że taka sytuacja może powodować zmiany na poziomie projektu (np. dodatkowe ustalenia projektowe, zmiany harmonogramu), jednak nie powinna wpływać na zakres testów. Wyjątkiem są tu sytuacje, gdy źródło błędów jest niezależne od Wykonawcy (np. brak gotowości systemu zewnętrznego do integracji).

W każdym z powyższych przypadków powinien powstać Raport z testów opisujący wyniki przeprowadzonych testów. W przypadku, gdy Raport opisuje testy powtarzane w wyniku wykrytych błędów, powstaje kolejna wersja dokumentu Raportu, na bazie stworzonych poprzednio.

Na diagramie przedstawionym na Rysunek 4 zawarto przebieg procedury testowej dla testów akceptacyjnych.



Rysunek 4. Przebieg procedury testowej dla testów akceptacyjnych

5 Środowisko testów

W kontekście testów należy wyróżnić następujące środowiska:

1) **Testowe** – środowisko stanowiące odzwierciedlenie środowiska produkcyjnego systemu w zakresie typów elementów architektury sprzętowo-programowej (przez istnienie np. serwera aplikacyjnego czy serwera bazodanowego). Dopuszczalne jest by było to środowisko o ograniczonej wydajności (mniejsza liczba maszyn czy procesorów, mniejszy rozmiar pamięci operacyjnej) i pojemności (mniejszy rozmiar pamięci masowej, słabsze).

Odpowiedzialność za to środowisko regulowana jest odpowiednimi zapisami umów i ustaleniami projektowymi, jednak rekomenduje się podział odpowiedzialności:

a) warstwa sprzętowa i oprogramowanie serwerowe – Zamawiający,

b) warstwa oprogramowania specyficznego dla danego rozwiązania oraz konfiguracja całości środowiska – Wykonawca.

2) **Deweloperskie** – środowisko specyfikowane w kontekście testów z punktu widzenia wymagań metodyki TDD. Za to środowisko powinien w całości odpowiadać Wykonawca.

3) **Preprodukcyjne / integracyjne / szkoleniowe** – środowisko stanowiące możliwie pełne odzwierciedlenie środowiska produkcyjnego. Nazwa tego środowiska zależy od jego przeznaczenia:

a) **preprodukcyjne** – gdy stanowi miejsce działań poprzedzających wprowadzenie rozwiązań na środowisko produkcyjne. Działania te nie muszą być bezpośrednio związane z testami i mogą obejmować np. działania administracyjne takie jak weryfikacja procedur (np. weryfikację poprawności procedury instalacji);

b) **integracyjne** – gdy środowisko służy weryfikacji poprawności integracji z systemami zewnętrznymi;

c) **szkoleniowe** – gdy jest miejscem prowadzenia szkoleń.

Istotnymi zagadnieniami w kontekście środowisk tej kategorii jest ich dostępność. Z uwagi na znaczny koszt repliki sprzętowo-programowej środowiska produkcyjnego, środowisko takie może być wykorzystywane do różnych celów i mogą zdarzyć się sytuacje, gdy jest potrzeba jednoczesnego wykorzystania go np. do testów i do szkoleń. Należy też zwrócić uwagę na fakt, że środowiska tego typu mogą być udostępniane (w całości lub w części) podmiotom zewnętrznym, co wymaga odpowiedniej obsługi organizacyjnej (komunikacja między podmiotami, współdzielenie danych, także wrażliwych, zachowanie spójności konfiguracji itd.). Za środowiska z tej kategorii odpowiada Zamawiający.

4) **Produkcyjne** – środowisko, gdzie wdrażane jest docelowe rozwiązanie. Na tym środowisku nie mogą być realizowane testy żadnego rodzaju i w żadnym zakresie. Za to środowisko odpowiada Zamawiający.

Środowisko w jakim realizowane są testy powinno zostać opisane w Planie Testów i jest to jedyne miejsce, gdzie to środowisko jest specyfikowane (w szczególności niedopuszczalne jest specyfikowanie środowiska testowego w Planie Wdrożenia, natomiast dopuszczalne jest przygotowanie oddzielnego Planu Wdrożenia środowiska testów). Opis środowiska powinien określać:

1) Charakter środowiska (np. testowe, integracyjne);

2) W zakresie części serwerowej środowiska:

   a) Infrastrukturę sprzętowo-programową;

   b) Przeznaczenie elementów infrastruktury (np. określenie jakim elementom architektonicznym docelowego systemu odpowiadają);

   c) Adresy sieciowe maszyn;

   d) Wymagane zaślepki serwisów zewnętrznych (o ile są wymagane przez zakres testów);

   e) Adresy usług testowych systemów zewnętrznych (o ile są wymagane przez zakres testów);

3) W zakresie maszyn roboczych:

   a) Infrastrukturę sprzętowo-programową ze szczególnym uwzględnieniem oprogramowania dodatkowego pozwalającego wykonać testy (np. przeprowadzić symulowane wywołanie usług, zweryfikować czasy wykonania operacji, otworzyć pliki w specyficznych formatach, przeprowadzić testy za pomocą różnych przeglądarek internetowych, zweryfikować zawartość logów i baz danych);

   b) Liczność maszyn roboczych;

   c) Charakteryzować urządzenia peryferyjne (rodzaje urządzeń wskazujących, drukarki, czytniki kart, rozdzielczość ekranów itd.).

Testy dopuszczeniowe muszą być realizowane w środowisku testowym. Miejsce wykonania testów akceptacyjnych powinno przedmiotem decyzji Zamawiającego lub ustaleń projektowych (określonych w Strategii testów). Zaleca się, aby testy akceptacyjne wszystkich iteracji projektu, z wyjątkiem ostatniej, realizowane były na środowisku testowym. Ostatnia iteracja testów, bezpośrednio poprzedzająca wdrożenie produkcyjne, powinna być przeprowadzona na środowisku integracyjnym. Wyniki testów wydajnościowych realizowanych na środowisku testowym powinny być traktowane z uwzględnieniem ograniczeń wydajnościowych tego środowiska. Testy jednostkowe realizowane są w środowisku deweloperskim.

6 Dane testowe

Dane testowe to wszelkie informacje wykorzystywane jako dane wejściowe dla systemu podczas testów lub używane do porównań w celu stwierdzenia poprawności uzyskania przez system danego stanu.

Zarówno charakter danych testowych, jak i ich wolumen powinny odpowiadać typowi testów, dla których dane są przygotowywane. Oznacza to, że na potrzeby testów dopuszczeniowych wymagane

jest przygotowanie danych zawierających pojedyncze obiekty umożliwiające wykonanie czynności z planu testów (np. scenariuszy testowych). Z kolei dla testów akceptacyjnych zakres danych obejmuje zarówno testy funkcjonalne, jak pozafunkcjonalne, a więc dane muszą zostać przygotowane w ilości pozwalającej ocenić np. wydajność systemu lub hierarchizację danych ze względu na uprawnienia. W przypadku braku wystarczającej ilości danych, należy posiadane dane powielić i ewentualnie rozmieścić w przestrzeni w celu uzyskania symulacji docelowego obciążenia testowanego systemu. Przy czym należy zadbać o poprawność i spójność tych danych aby ich ewentualne wady nie zakłóciły wyników testów wydajnościowych.

Dane niezależnie od ich przeznaczenia powinny możliwie blisko odzwierciedlać przypadki rzeczywiste. Uproszczone obiekty są dopuszczalne tylko dla przejrzystości weryfikacji szczególnych sytuacji (np. weryfikacji wybranych scenariuszy z przypadków użycia). Dane wykonywane podczas testów muszą być odpowiednio oczyszczone (np. przez procesy depersonalizacji i anonimizacji) z informacji wrażliwych (takich jak dane osobowe, czy informacje niejawne).

Za przygotowanie danych odpowiada Wykonawca, który może zwrócić się do Zamawiającego o wskazanie istotnych z punktu widzenia biznesowego sytuacji powodujących powstanie szczególnych struktur danych lub o przekazanie wzorcowych, przykładowych danych jakie będzie przetwarzał system (o ile takie istnieją i Zamawiający zgodzi się je udostępnić). Przygotowane dane testowe podlegają akceptacji przez Zamawiającego.

7 Podejście TDD

W trakcie realizacji podprocesu implementacji i testów stosowana powinna być metodyka Test Driven Development (TDD), co w szczególności oznacza:

1) Budowanie przez Wykonawcę testów jednostkowych (ang. Unit Tests) dla przygotowywanego rozwiązania programistycznego;

2) Stosowanie przez Wykonawcę narzędzia, które pozwala na automatyczne uruchomienie opracowanych testów jednostkowych w trakcie budowania binariów (budowania kodu wynikowego) i dostarczenia raportu z wynikami testów;

3) Stosowanie przez Wykonawcę narzędzia, które pozwala na dostarczenie raportu na temat procentowego pokrycia linii kodu źródłowego testami jednostkowymi – ile procent linii kodu źródłowego (nie uwzględniając komentarzy – ang. Non-Comment Lines of Code (NCLOC) jest testowanych automatycznie (ang. code coverage);

4) Narzędzie opisane w punktach 2) i 3) powinno pozwalać na śledzenie historii procesu budowy;

5) Dla dostarczonego rozwiązania rezultaty wszystkich testów jednostkowych powinny być pozytywne (ang. passed);

6) Dla dostarczonego rozwiązania procentowe pokrycie linii kodu źródłowego testami jednostkowymi powinno wynosić minimum 80%.

Wykonawca zobowiązany jest do zainstalowania w środowisku Zamawiającego (np. w CODGiK) bądź do udostępnienia Zamawiającemu poprzez Internet środowisk deweloperskich zawierające wskazane powyżej narzędzia, w szczególności pozwalających na sprawdzenie zgodności kodu źródłowego z testami jednostkowymi. Wykonawca zobowiązany jest również do udzielenia Zamawiającemu instruktażu mającego na celu zdobycie umiejętności pozwalających na samodzielne sprawdzenie zgodności kodu źródłowego oraz przekaże przypadki testowe wraz z opisami ich przejścia.

Proces zliczania kodu pokrytego testami polega na rejestrowaniu wykonania kolejnych linii kodu (instrukcji) w ramach metod i klas podczas uruchomienia testów jednostkowych. Dopuszczalna jest prezentacja innych metryk pokrycia testami (np. na poziomie instrukcji, klas lub metod), jednak podstawową metryką jest pokrycie linii kodu. Należy zauważyć, że testy jednostkowe również opisane są kodem, który jednak nie stanowi kodu testowanej aplikacji. W przypadku wadliwej konfiguracji procesu TDD kod testów jednostkowych jest zliczany jako podlegającym testom (ponieważ jest podczas testów „wykonywany"). Powoduje to zafałszowanie wyników badania pokrycia i jest niedopuszczalne.

8 Inne zagadnienia związane z procesem testowania

Niniejszy rozdział przedstawia zagadnienia związane z testowaniem, które w szczegółach nie zostały ujęte w pozostałych częściach dokumentu.

8.1 Scenariusze i przypadki testowe

Wykonanie przypadku testowego oznacza:

1) Zapewnienie, jako startowego, stanu systemu zgodnego z warunkami początkowymi;

2) Przeprowadzenie interakcji z systemem zgodnie z kolejnością opisaną w krokach przypadku testowego i przy wykorzystaniu odpowiednich danych;

3) Zweryfikowanie zaistnienia stanów systemu opisanych przez przypadek.

Jeżeli nie jest możliwe spełnienie warunków zapisanych w jednym z powyższych punktów, w szczególności nie można doprowadzić do stanu opisanego w warunkach początkowych, nie jest możliwe wykonanie określonego kroku interakcji lub nie zaistniał/nie było możliwe stwierdzenie zaistnienia stanu systemu wynikającego z opisu przypadku, wynik wykonania przypadka testowego jest negatywny. W przeciwnej sytuacji wynik jest uznawany za pozytywny.

Ziarnistość przypadków testowych powinna być taka, by jeden przypadek odpowiadał jednemu scenariuszowi (przebiegowi) przypadku użycia i ewentualnie dodatkowej funkcjonalności innych przypadków użycia włączanej/rozszerzanej w ramach tego scenariusza.

Każdy przypadek testowy powinien być opisany przez następujące atrybuty:

1) Nazwa;
2) Identyfikator;
3) Opis (nie może być powtórzeniem nazwy);
4) Weryfikowane wymagania lub przypadki użycia (w przypadku weryfikacji częściowej wymagane jest wskazanie elementów weryfikowanych – np. „wymaganie WM.0001, pkt 1 i 2”, „przypadek użycia PU.007, scenariusz B”);
5) Warunki wstępne;
6) Uporządkowana lista kroków (atomowych czynności wykonywanych podczas testów) powiązanych z danymi testowymi oraz oczekiwanymi dla danego kroku rezultatami (stanami systemu).

Scenariusz testowy to specyfikacja procedury testowej oparta o powiązane ze sobą przypadki testowe. Powiązanie przypadków oznacza ich wykonanie w ustalonej kolejności, w sposób sekwencyjny lub w sposób zakładający alternatywy lub cykle regulowane możliwymi do kontrolowania warunkami. Scenariusze testowe posiadają, podobnie jak przypadki, warunki początkowe i oczekiwane rezultaty wykonania. Wynikają one wprost ze składowych przypadków testowych oraz z dodatkowych uwarunkowań. Jeżeli któryś z przypadków wskazanych w scenariuszu zostanie wykonany z wynikiem negatywnym, wynik wykonania scenariusza jest negatywny. W takiej sytuacji można kontynuować wykonanie pozostałych przypadków, jednak wynik pozostaje negatywny, niezależnie od wyniku ich wykonania. Przypadek testowy otrzymuje wynik negatywny w przypadku, gdy nie udało się w trakcie wykonywania przypadku testowego osiągnąć oczekiwanych rezultatów (wynik przypadku testowego może być pozytywny pomimo wystąpienia błędów niewpływających na możliwość osiągnięcia zakładanych rezultatów, np. błędów kosmetycznych lub średnich).

Ziarnistość scenariuszy testowych powinna być taka, by jeden scenariusz testowy odpowiadał przynajmniej jednemu scenariuszowi (przebiegowi) przypadku użycia, a maksymalnie wszystkim przebiegom przypadków użycia zgromadzonych w jednym pakiecie i dodatkowej funkcjonalności innych przypadków użycia włączanej/rozszerzanej w ramach tych przypadków użycia.

Każdy scenariusz testowy powinien być opisany przez następujące atrybuty:

1) Nazwa;
2) Identyfikator;
3) Opis (nie może być powtórzeniem nazwy);

4) Weryfikowane wymagania lub przypadki użycia (w przypadku weryfikacji częściowej wymagane jest wskazanie elementów weryfikowanych – np. „wymaganie WM.0001, pkt 1 i 2”, „przypadek użycia PU.007, scenariusz B”);

5) Warunki wstępne;

6) Uporządkowana lista kroków (wynikających z przypadków testowych) powiązanych z danymi testowymi oraz oczekiwanymi dla danego kroku rezultatami (stanami systemu) oraz dodatkowymi warunkami opisującymi wykonanie danych kroków (dotyczy cykli i alternatyw). Warunki te nie powinny być czynnościami, które polegają na wywołaniu funkcji systemu podlegających testowaniu w ramach danej iteracji testów. Dla ułatwienia realizacji testów w ramach scenariuszy testowych kroki scenariuszy mogą być reprezentowane przez opisujące je listy kroków, powiązane z nimi dane testowe i warunki.

8.2 Kategorie błędów

Poniższa sekcja opisuje podstawowe, rekomendowane kategorie, na jakie można dzielić błędy wykrywane podczas testowania.

Kategorie te są następujące:

- Błąd krytyczny;
- Błąd poważny;
- Błąd średni;
- Błąd kosmetyczny.

Wykorzystanie poniższej taksonomii błędów wymaga zdefiniowania na poziomie strategii testów pojęcia istotnej funkcjonalności. Podstawowe brzmienie tej definicji to: „Funkcjonalność systemu, która realizuje wymaganie lub przypadek użycia z przypisanym najwyższym statusem.”

W przypadku, gdy w procesie testowania wykonywane są inne testy niż testy oprogramowania (np. testy poprawności technicznej interfejsów, testy zasilenia bazy danych) w Strategii testowania należy odpowiednio rozbudować definicje poszczególnych kategorii błędów.

8.2.1 Błąd krytyczny

Błąd krytyczny występuje, gdy zajdzie jedna z poniższych sytuacji:

1) Całkowity brak odpowiedzi systemu na sygnały;

2) Brak działania lub implementacji istotnej funkcjonalności;

3) Błędy innych typów uniemożliwiające korzystanie z systemu lub jego istotnej funkcjonalności;

4) Brak spełnienia wymagania pozafunkcjonalnego;

5) Brak spełnienia zapisu formalnego umowy związanego z cechami pozafunkcjonalnymi, wyglądem lub funkcjonalnością systemu (np. brak logotypów, niezgodność ze standardami i normami przywołanymi w umowie).

8.2.2 Błąd poważny

Błąd poważny występuje, gdy zajdzie jedna z poniższych sytuacji:

1) Nieprawidłowe działanie systemu inne niż dotyczące istotnej funkcjonalności systemu;

2) Wydajność systemu uniemożliwiająca zrealizowanie scenariuszy testowych w czasie określonym w Planie testów (znaczne przekroczenie czasów realizacji scenariuszy testowych wobec założonych w Planie testów);

3) Powtarzające się błędy kosmetyczne i średnie utrudniające realizowanie testów (np. okazjonalne wylogowanie użytkownika, zrywanie połączeń, uciążliwe komunikaty, konieczność nieplanowanego wielokrotnego powtarzania tych samych czynności);

4) Komunikat wprowadzający użytkownika w błąd co do wykonania istotnej funkcjonalności.

8.2.3 Błąd średni

Błąd średni występuje, gdy wykryty zostanie błąd oprogramowania niewpływający na możliwość korzystania z funkcjonalności. Przykłady błędów średnich:

1) Niezgodna z wymaganiami konieczność wywołania tej samej funkcji wielokrotnie w celu uzyskania pojedynczego rezultatu;

2) Przy wywołaniu funkcjonalności pojawia się komunikat o błędzie, jednak system później realizuje działania związane z funkcjonalnością;

3) Komunikat wprowadzający użytkownika w błąd co do wykonania innej niż istotna funkcjonalności;

4) Przekroczony czas reakcji systemu w stosunku do wymaganej jego wydajności.

8.2.4 Błąd kosmetyczny

Błąd kosmetyczny występuje w sytuacji, gdy zaistnieje niezgodność sposobu prezentacji informacji w systemie niezgodna z wymaganiami i nie ma to wpływu na wykorzystanie jakiejkolwiek funkcjonalności lub spełnienie standardów (jest to np. defekt ergonomiczny). Przykłady błędów kosmetycznych:

1) Niewłaściwy rozmiar okna wymagający jego zmiany w celu realizacji przypadka testowego;

2) Literówki w interfejsie użytkownika;

3) Komunikat zgodny co do wartości informacyjnej jednak niezgodny ze specyfikacją co do brzmienia.

8.3 Limity błędów

Rekomendowane limity błędów, których niespełnienie powoduje działania opisane w rozdziale Procedura testów, są następujące:

1) Dla testów dopuszczeniowych

Kategoria błędów	Limit
Krytyczny	0
Poważny	1% (zaokrąglając w dół) przypadków testowych
Średni	10% (zaokrąglając w dół) przypadków testowych
Kosmetyczny	Bez limitu

Tabela 2. Rekomendowane limity błędów w trakcie testów dopuszczeniowych

2) Dla testów akceptacyjnych

Kategoria błędów	Limit
Krytyczny	0
Poważny	0
Średni	5% (zaokrąglając w dół) przypadków testowych
Kosmetyczny	Bez limitu

Tabela 3. Rekomendowane limity błędów w trakcie testów akceptacyjnych

Powyższe limity błędów mogą podlegać modyfikacjom w zależności od specyfiki danego systemu.

8.4 Ujęcie testów w harmonogramie projektu

Ze względu na możliwe występnie błędów w trakcie testów, w harmonogramie realizacji zamówienia powinien zostać uwzględniony dodatkowy czas na ewentualnie wydłużenie się testów. Sugerowane jest zapewnienie dodatkowego czasu odpowiadającego czasowi niezbędnemu na przeprowadzenie wszystkich scenariuszy testowych zakładając ich przebieg z wynikiem pozytywnym.

9 Załączniki

1. „Strategia procesu testowania dla Umowy na Budowę oraz rozwój e-usług i narzędzi w ramach projektów CAPAP, ZSIN Faza II i K-GESUT wraz ze szkoleniami.”