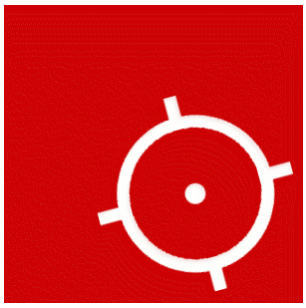



Podstawą tego rozdziału jest Foundation Level Syllabus wydany przez ISTQB.

## Poziomy testowania

Dla każdego poziomu testowania możemy wyróżnić:
- ogólny cel
- odnośnik do tworzenia przypadków testowych (np. podstawa testów)
- cel testowy (np. co będzie testowane)
- typowe defekty i błędy, które mogą zostać znalezione, wymagania kontrolerów testowych
- narzędzie wspomagające
- specyficzne podejście
- odpowiedzialność.

**Testowania komponentów**

Ten rodzaj testowania poszukuje defektów w oprogramowaniu oraz weryfikuje funkcjonalność (np. moduły, programy, obiekty, klasy itp.) dających się osobno testować komponentów. Może to być zrobione w oddzieleniu od reszty systemu, w zależności od cyklu rozwoju oprogramowania. Zaślepki (ang. stub), sterowniki i symulatory mogą zostać do tego celu użyte. Testowanie komponentów może zawierać testowanie funkcjonalności oraz pewnych nie funkcjonalnych parametrów, takich jak zachowanie zasobów (np. wycieki pamięci) lub testowanie odporności na atak, ale także testowanie strukturalne (np. pokrycie gałęzi). Przypadki testowe tworzone są na podstawie specyfikacji komponentu, projektu oprogramowania lub modelu danych. Najczęściej, o testowaniu komponentów mówimy, gdy mamy dostęp do kodu źródłowego i wsparcie środowiska tworzenia oprogramowania, tj. narzędzia debuggujące oraz pomoc programisty, piszącego kod. Defekty zazwyczaj naprawiane są zaraz po znalezieniu, bez formalnego raportowania.
Jedną z możliwości stosowania testów komponentowych jest przygotowanie i zautomatyzowanie przypadków testowych przed właściwym programowaniem. Takie podejście nazywamy "na początku test" lub programowanie kierowane testami. Taki sposób rozwoju oprogramowania jest wysoko iteratywny i bazowany na cyklicznym powstawaniu przypadków testowych, a następnie budowaniu i integracji małych kawałków kodu i wykonywanie testów komponentowych aż do uzyskania poprawnych wyników.

**Testowania integracyjne**

Następnym etapem testowania jest sprawdzenie interfejsów pomiędzy komponentami i oddziaływania różnych części systemu, takich jak systemu operacyjnego, systemu plików, oprzyrządowania i połączeń między systemami.
Możemy wyróżnić kilka poziomów integracji i mogą być one przeprowadzone na obiektach o różnych rozmiarach. Przykład:
+ testowanie integracji komponentów sprawdza zależności między komponentami i wykonywane jest po testowaniu komponentów
+ testowanie integracji systemu sprawdza oddziaływanie między różnymi systemami i może być wykonane po testowaniu systemu.

W przypadku, gdy dział rozwoju oprogramowania kontroluje jedynie jedną stronę interfejsu, końcowy produkt może mieć problemy z współpracą i z innymi systemami. W wielu przypadkach może się

Wyłączne prawa autorskie do tego dokumentu posiadają „testerzy.pl". Rozpowszechnianie dla celów komercyjnych jak i niekomercyjnych jest dozwolone tylko pod warunkiem podania źródła.

okazać, że stosowane jest wiele systemów i platform, co będzie miało kolosalny wpływ na zasięg naszego produktu.
Im większy jest rozmiar integracji tym trudniej zlokalizować przyczynę występowania błędu. Zwiększa to znacznie ryzyko i czas naprawy.
Strategie integracji są bazowane na architekturze systemu (np. z góry na dół; z dołu do góry), funkcjonalności, sekwencjach wykonywania wymiany danych, lub innych aspektach systemu lub komponentu. W celu zredukowania ryzyka późnego wykrycia błędu, integracja powinna raczej być wykonywana stopniowo, a nie metodą wszystko na raz (tzw. integracja "wielkiego wybuchu"). Również testowanie parametrów niefunkcjonalnych (np. wydajności) może być dodane do testów integracyjnych.
Na każdym poziomie integracji tester koncentruje się wybiórczo na danym obszarze. Przykładowo, gdy integrujemy moduł A z modułem B, testera interesuje komunikacja między nimi, a nie funkcjonalność poszczególnych modułów. W tym przypadku zarówno funkcjonalna jak i strukturalna metoda może zostać użyta.
W doskonałym świecie tester powinien rozumieć architekturę i wpływ planowania integracji, jeśli testy integracyjne planowane są przed stworzeniem komponentów lub systemu, mogą one zostać zbudowane zgodnie z wymaganiami w najbardziej wydajny sposób testowania.

**Testowanie systemów**

Testowanie systemów obejmuje sprawdzenie zachowania całości systemu/produktu w porównaniu z założeniami wstępnymi projektu lub programu.
W testowaniu systemu, środowisko testowe powinno być zbliżone do tego, w jakim nasz produkt będzie pracował realnie, aby zminimalizować ryzyko wystąpienia błędów specyficznych dla danego środowiska.
Testowanie systemowe może być oparte na ryzyku lub/i specyfikacji wymagań, procesach biznesowych, przypadkach użycia, lub innych, wysoko poziomowych zasad zachowania systemu, zależności z systemem operacyjnym i zasobami systemowymi.
Testowanie systemowe powinno uwzględniać zarówno funkcjonalne jak i niefunkcjonalne wymagania systemu. Wymagania mogą być definiowane jako tekst lub/i modele. Testerzy powinni umieć pracować z niekompletnymi lub nieudokumentowanymi wymaganiami. Testowanie systemowe funkcjonalnych wymagań rozpoczynamy wybierając najbardziej odpowiednią technikę opartą na specyfikacji (czarnoskrzynkową). Dla przykładu, tabela decyzji może zostać stworzona dla kombinacji efektów opisanych w zasadach biznesowych. Techniki oparte na strukturze (białoskrzynkowe) mogą być użyte do oceny wnikliwości testowania w odniesieniu do elementów struktury, takich jak struktura menu lub nawigacji na stronie internetowej.
Testowanie tego typu zazwyczaj wykonywane jest przez niezależny zespół testowy

**Testowanie akceptacyjne**

Testy akceptacyjne zazwyczaj są w gestii klienta lub użytkownika systemu, ale również współudziałowcy mogą być zaangażowani w ten proces.
Celem testów akceptacyjnych jest sprawdzenie niezawodności systemu, części systemu lub specjalnych niefunkcjonalnych parametrów systemu. Znajdywanie defektów nie jest głównym celem tego testowania. Testy akceptacyjne mogą ocenić zdolność systemu do lokowania i użycia, jednak nie jest to końcowy etap testowania, przykładowo testy integracyjne wysokiego poziomu mogą nastąpić po testach akceptacji.
Testy akceptacyjne mogą być czymś więcej niż tylko pojedynczym poziomem testowym np.:
+ oprogramowanie zakupione w sklepie może zostać przetestowane akceptacyjnie, gdy zostanie zainstalowane lub zintegrowane
+ testy akceptacyjne użyteczności komponentu może zostać przeprowadzone podczas testów komponentowych
+ testy akceptacyjne nowych poprawek mogą zostać przeprowadzone zanim system zostanie przetestowany.

Testy akceptacyjne zazwyczaj dzielą się na testy:

**użytkownika -** weryfikuje dopasowanie systemu do potrzeb użytkowników

**operacyjne -** akceptacja systemu przez administratorów systemu zawierające: sprawdzenie kopii zapasowej i zdolności do przywrócenia funkcjonalności po wystąpieniu problemów, zarządzanie użytkownikami, zadania serwisowe, cykliczne sprawdzenie bezpieczeństwa

**kontraktowe i regulacyjne -** testowanie kryteriów wytworzenia oprogramowania specyfikowanego dla klienta. Kryteria akceptacyjne powinny być zdefiniowane po uzgodnieniu

Wyłączne prawa autorskie do tego dokumentu posiadają „testerzy.pl". Rozpowszechnianie dla celów komercyjnych jak i niekomercyjnych jest dozwolone tylko pod warunkiem podania źródła.

kontraktu. Testy regulacyjne są wykonywane w zgodzie z rządowymi lub legislacyjnymi uregulowaniami.

**alfa i beta (testowanie polowe) -** ludzie tworzący oprogramowanie dla klienta kupującego produkt z półki w sklepie oczekują od niego informacji zwrotnej zanim produkt pojawi się na rynku. Testowanie alfa odbywa się w organizacji tworzącej oprogramowanie. Testowanie beta dokonuje się po stronie odbiorcy oprogramowania. Obydwa typy testowania przeprowadzane są przez potencjalnych odbiorców produktu.
Dopuszczalne są różne nazwy dla tego typu testowania w zależności od specyfiki wytwórcy np. fabryczne testy akceptacyjne, testy akceptacyjne działu itp.

testerzy.pl

Wyłączne prawa autorskie do tego dokumentu posiadają „testerzy.pl". Rozpowszechnianie dla celów komercyjnych jak i niekomercyjnych jest dozwolone tylko pod warunkiem podania źródła.

## Typy testowania

Aktywności testowe mogą być ukierunkowana na weryfikację systemów (lub ich części) pod kątem różnych, specyficznych celów.
Testowanie jest skoncentrowane na konkretnych celach testowych, które można podzielić na testowanie funkcjonalne, testowanie parametrów niefunkcjonalnych jakości, takich jak wiarygodność lub używalność, struktura lub architektura oprogramowania lub systemu, lub powiązane ze zmianami, np. potwierdzenie, że defekt został naprawiony (testy potwierdzające) i szukanie niepożądanych zmian (testy regresji).
Model oprogramowania może być rozwijany i/lub używany w strukturalnym i funkcjonalnym testowaniu. Dla przykładu, w funkcjonalnym testowaniu: modelu procesu, model zmiany stanów lub zwykły język specyfikacji; dla testowania strukturalnego model kontroli lub model struktury menu.

**Testowanie funkcjonalne - Czarna skrzynka**

Funkcje, jakie system, podsystem lub komponent wykonują mogą być opisane jako wymagania specyfikacji, przypadki użycia lub specyfikacja funkcjonalności, mogą też pozostać nieudokumentowane. Funkcje definiuje się jako czynności wykonywane przez system.
Testy funkcjonalne oparte są na tych funkcjonalnościach lub funkcjach (opisanych w dokumentacji lub zrozumiane przez testera) i mogą być wykonane na każdym poziomie testowania (np. testy komponentów oparte są na specyfikacji komponentów).
Techniki oparte na specyfikacji mogą być użyte do zdefiniowania warunków testowych i przypadków testowych bezpośrednio z funkcjonalności oprogramowania lub systemu. Testowanie funkcjonalne rozważa zewnętrzne zachowanie oprogramowania (czarna skrzynka).
Testowanie funkcjonalne bezpieczeństwa to sprawdzanie funkcji (np. firewall) związanych z wykryciem niebezpieczeństw i zagrożeń takich jak wirusy, mających za zadanie uszkodzenie systemu.

**Testowanie niefunkcjonalne - określanie parametrów**

Testy niefunkcjonalne obejmują, ale nie są ograniczone do, test wydajności, test obciążeń, test stresu, test używalności, test współpracy, test serwisowy, test niezawodności i test zdolności do pracy w na różnych platformach. Ten rodzaj testowanie określa jak system działa.
Testy niefunkcjonalne mogą być wykonane na wszystkich poziomach testowych. ¤¤¤¤¤Pojęcie niefunkcjonalnego testowania opisuje testy wymagane do pomiarów charakterystyk systemu i oprogramowania, które może być oceniony na skali, takich jak czas odpowiedzi podczas testowania wydajności. Te testy mają odniesienie w modelu jakości np. ISO 9126.

**Testowanie strukturalne - biała skrzynka**

Testy strukturalne mogą być wykonane na wszystkich poziomach testowych. Techniki strukturalne są najlepsze, gdy używa się ich po technikach opartych na specyfikacji, w celu wsparcia pomiarów dokładności testowania poprzez ocenę pokrycia typu struktury.
Pokrycie jest mierzone wykonaniem kodu za pomocą szeregu testów i wyrażone w procentach pokrytych czynników. Jeśli pokrycie wynosi mniej niż 100% znaczy to, że więcej przypadków testowych może zostać zaprojektowanych dla tych czynników, które zostały pominięte, aby zwiększyć pokrycie.
Na wszystkich poziomach testów, ale szczególnie w testowaniu komponentów i integracji komponentów, możemy używać narzędzi do pomiarów pokrycia elementów kodu, tj. deklaracji i decyzji. Testowania strukturalne może być oparte na architekturze systemu np. hierarchii odwołań.
Testowanie strukturalne może również być użyte dla systemu, integracji systemu i poziomu testów akceptacyjnych (np. modeli biznesowych lub struktury menu).

**Testowanie regresyjne/potwierdzające**

Kiedy defekt zostaje znaleziony i naprawiony wtedy oprogramowanie powinno być ponownie przetestowane, aby upewnić się, że defekt został usunięty. Taką czynność nazywamy testami potwierdzającymi. Naprawianie błędów jest czynnością twórców oprogramowania a nie testerów.

Testy regresji są powtarzalnymi testami na już przetestowanym programie, po modyfikacjach, w celu wykrycia innych defektów wprowadzonych lub nie odkrytych podczas "naprawy". Mogą one być znajdować się w testowanym oprogramowaniu jak i w innym powiązanym lub niepowiązanym z nim komponencie. Zostaje wykonany, gdy oprogramowanie, lub środowisko, zostaje zmienione - testy

Wyłączne prawa autorskie do tego dokumentu posiadają „testerzy.pl". Rozpowszechnianie dla celów komercyjnych jak i niekomercyjnych jest dozwolone tylko pod warunkiem podania źródła.

regresji. Głównym zadaniem tych testów jest ocena ryzyka a nie znalezienia defektów w oprogramowaniu, które poprzednio działało.
Testy powinny być powtarzalne, jeśli mają służyć do potwierdzenia lub towarzyszyć testom regresyjnym.
Testy regresyjne mogą zostać wykonane na wszystkich poziomach testowych i zajmują się funkcjonalnością, zdolnościami niefunkcjonalnymi i testowaniem strukturalnym. Zbiory testów regresji są wykonywane wiele razy i ogólnie spowalniają rozwój oprogramowania, są więc świetnym kandydatem na automatyzację.

**Testowanie serwisowe**

Raz wydane oprogramowanie jest często w użyciu przez lata lub dekady. Podczas tego czasu system i jego środowisko podlega korekcjom, jest zmieniane lub rozszerzane. Testowanie serwisowe dokonywane jest na istniejącym systemie operacyjnym i jest wyzwalana modyfikacjami, migracjami lub niejakim starzeniem się oprogramowania lub systemu.
Modyfikacje zawierają: zaplanowane poprawki (np. kolejne odsłony), zmiany awaryjne, zmiany środowiska pracy takich jak planowane poprawki do systemów operacyjnych lub baz danych, lub też łatki na system operacyjny spowodowane znalezionymi dziurami lub błędami.
Testy serwisowe dla migracji (np. z jednej platformy na inną) powinny zawierać testy operacyjne nowego środowiska, tak samo jak zmiany w oprogramowaniu.
Testy serwisowe dla starzejącego się systemu powinny uwzględniać testy migracji danych lub ich archiwizowania, gdy wymagany jest długi okres przechowywania danych.
Dodatkowo oprócz testowania tego, co zostało zmienione, serwis testowy uwzględnia wydajne testy regresji części systemu, które nie zostały zmienione. Zakres testów serwisowych jest związany z ryzykiem zmian, rozmiar istniejącego systemu i rozmiar zmian. W zależności od zmian, serwis testowy może zostać wykonany na każdym poziomie testów i dla każdego typu.
Determinowanie jak zmiany mogą wpłynąć na istniejący system nazywa się analizą wpływu i używa się ich do wsparcia decyzji jak wiele testów regresji przeprowadzić.
Testy serwisowe mogą być trudne, jeśli specyfikacja jest przeterminowana lub został zagubiona.

**Statyczne techniki testowania**

**Przeglądy i procesy testowe**

Statyczne techniki testowe nie uruchamiają oprogramowania, które jest testowane, są one ręczne (przeglądy) lub automatyczne (analiza statyczna).
Przeglądy są typem testowania oprogramowania (włączając w to kod) i mogą być przeprowadzone zanim dynamicznie uruchomi się produkt. Błędy wykryte dzięki przeglądowi we wczesnej fazie cyklu tworzenia oprogramowania są zazwyczaj znacznie tańsze do usunięcia niż te wykryte podczas wykonywania testów (np. defekty znalezione w wymaganiach).
Przegląd może zostać wykonany w całości jako czynności manualne, ale jest jeszcze wsparcie narzędzi. Główna czynność manualna to sprawdzenie przeglądanego produktu i skomentowanie go.
Każdy produkt może zostać przetestowany poczynając od wymagań specyfikacji, specyfikację projektową, kod, test plany, test specyfikacje, przypadki testowe, skrypty testowe, przewodnik użytkownika lub strony internetowe.
Korzyści przeglądów to wczesne wykrycie defektów i ich korekta, poprawa produktywności rozwoju oprogramowania, zredukowanie czasu potrzebnego na tworzenie oprogramowania, zredukowanie czasu i kosztów testowania, zredukowanie kosztów cyklu życia, mniej błędów i poprawa komunikacji.
Przegląd może znajdywać defekty pominięte podczas dynamicznych testów np. w wymaganiach.
Przegląd, analiza statyczna i dynamiczne testowanie mają te same cele - znajdywanie błędów.
Wypełniają się: różne techniki mogą znaleźć różne typy defektów efektywniej i wydajniej. Gdy technik dynamicznych raczej znajdują problemy spowodowane przez defekty to przeglądy znajdują same defekty.
Typowe defekty są łatwiejsze do znalezienia poprzez przegląd niż przez testy dynamiczne: odchylenia od standardu, złe wymagania, wady projektowe, niedostateczna serwisowalność i zła specyfikacja interfejsów.

**Proces przeglądu**

Wyróżnia się przeglądy od bardzo nieoficjalnych do bardzo sformalizowanych (np. ustruktyryzowane i regulowane). ¤¤Oficjalność procesu przeglądu jest związana z czynnikami takimi jak dojrzałość procesu rozwoju oprogramowania, legalnych lub uregulowanych wymaganiach lub konieczności przeprowadzenia audytu.

Wyłączne prawa autorskie do tego dokumentu posiadają „testerzy.pl". Rozpowszechnianie dla celów komercyjnych jak i niekomercyjnych jest dozwolone tylko pod warunkiem podania źródła.

Różne formy przeglądu są przeprowadzone w zależności od ustalonych celów przeglądu (np. znajdywanie defektów, zrozumienie, lub dyskusja i decyzja o znalezieniu wspólnego zdania).

**Fazy**

Wyróżnia się następujące fazy w zakresie przeglądu:
+ planowanie - wyznaczenie obsługi, zdecydowanie o rolach, zdefiniowanie kryteriów wejścia i wyjścia dla bardziej formalnych typów przeglądów (np. inspekcja); wybór, które części dokumentu sprawdzić.
+ rozpoczęcie - dystrybucja dokumentów; wyjaśnienie uczestnikom przeglądu celów, procesów i dokumentów; sprawdzenie osiągnięcia kryterium wejścia (dla bardziej formalnych typów przeglądu)
+ indywidualne przygotowanie - praca wykonana przez różnych uczestników przed spotkaniem przeglądowym, wynotowanie ewentualnych błędów, pytań i komentarzy
+ spotkanie przeglądowe: dyskusja lub rejestracja, zakończone dokumentem podsumowującym (formalne przeglądy). Uczestnicy spotkania mogą wynotować defekty, wyartykułować swoje rekomendacje jak radzić sobie z błędami, lub podjąć decyzję o defektach.
+ poprawki - naprawienie znalezionych błędów, zazwyczaj zrobione przez autora
+ kontynuacja - sprawdzenie czy defekty zostały naprawione lub ocenione, zebranie metryk i sprawdzenie kryterium wyjścia (formalne przeglądy)

Role i odpowiedzialność

Typowe role w formalnych przeglądach:
+ kierownik - decyduje o wykonaniu przeglądu, rezerwuje czas w projekcie i określa czy cel przeglądów został osiągnięty
+ moderator - osoba, która prowadzi przegląd dokumentacji lub zbioru dokumentacji, włączając planowanie przeglądu, prowadzenie spotkania, i kontynuacja po spotkaniu. Gdy konieczne, moderator może być mediatorem pomiędzy różnymi punktami widzenia i często jest osobą wspierająca sukces przeglądu
+ autor - osoba posiadająca odpowiedzialność za dokument(y), które mają zostać przeglądnięte
+ przeglądający - osoby ze specjalnym technicznym lub biznesowym przygotowaniem (nazywani również sprawdzający lub inspektorami), którzy po koniecznych przygotowaniach, zidentyfikują i opiszą znalezisk (np. defekty) w produkcie będącym przeglądanym. Przeglądający powinni być wybrani dla reprezentowania różnych perspektyw, reprezentowania różnych ról w procesie przeglądu i brać udział w spotkaniach przeglądowych
+ zapisujący - dokumentujący wszystkie poruszone problemy i otwarte tematy, które zostały zasygnalizowane podczas spotkania

Sprawdzając dokumentację pod różnymi perspektywami i używając wcześniej przygotowanych punktów kontrolnych można podnieść efektywność przeglądu np. punkty kontrolne użytkownika, serwisanta, testera czy lista punktów do sprawdzenia typowych problemów wymagań.

**Typy przeglądów**

Pojedynczy dokument może być tematem więcej niż jednego przeglądu. Jeśli więcej niż jeden jest użyty, kolejności może się różnić. Dla przykładu nieoficjalne przeglądy mogą być przeprowadzone przed technicznymi, a inspekcje wymagań mogą poprzedzać sprawdzenie ich przez klienta.
Główne parametry, opcje i cele typowych przeglądów:
+ nieformalne - niezdefiniowany proces, programowanie w parach lub techniczne prowadzenie przeglądów projektów i kodu, może być udokumentowane, może się różne w zależności od przydatności przeglądającego, główny cel to tanio osiągnąć korzyści
+ przejście przez - spotkanie prowadzone przez autora, przygotowany scenariusz, grupa kolegów, sesja bez ustalonego końca, opcjonalnie przed spotkaniem przygotowanie przeglądów, raport przeglądu, lista znalezisk z opisem, może być różne w praktyce, główny cel: uczenie, osiąganie zrozumienia, znajdywanie defektów
+ techniczne - udokumentowane, zdefiniowany proces detekcji defektów zawierający uwagi koleżeńskie i ekspertów technicznych, może być przeprowadzone bez obecności kierownictwa, w doskonałym przypadku powinno być prowadzone przez moderatora, a nie autora, przygotowane spotkanie, opcjonalnie przygotowana lista punktów kontrolnych, raport poprzeglądowy, lista znalezisk i udział kierownictwa, może być różna w zależności od okoliczności, główny cel: dyskusja, podejmowanie decyzji, sprawdzanie alternatyw, znajdowanie defektów, rozwiązywanie technicznych problemów, sprawdzenie zgodności z specyfikacją i standardami.

Wyłączne prawa autorskie do tego dokumentu posiadają „testerzy.pl". Rozpowszechnianie dla celów komercyjnych jak i niekomercyjnych jest dozwolone tylko pod warunkiem podania źródła.

+ inspekcja - prowadzona przez przygotowanego merytorycznie moderatora, sprawdzenie przez kolegów, zdefiniowane role, zawiera metryki, proces formalny oparty na zasadach i punktach kontrolnych z kryterium wejścia i wyjścia, przygotowanie przed spotkaniem, raport z inspekcji, lista znalezisk, formalny proces sprawdzenia wykonania założeń spotkania, opcjonalnie proces ulepszania, główny cel znajdowanie defektów.

**Cechy skutecznego przeglądu**

Aby skutecznie przeprowadzić przegląd należy:
+ każdy przegląd musi mieć wcześniej zdefiniowane cele
+ odpowiedni ludzie są zaangażowani w cele przeglądu
+ znajdywanie defektów jest mile widziane i wyrażane w sposób obiektywny
+ czynniki ludzkie i aspekt psychologiczny są brane pod uwagę (np. stworzenie środowiska do zdobywania doświadczenia przez wszystkich uczestników przeglądu)
+ techniki przeglądów są odpowiednie i pasujące do typu i poziomu oprogramowania i uczestników przeglądu
+ lista punktów kontrolnych jest użyta odpowiednio w celu zwiększenia efektywności identyfikacji defektów.
+ trening w temacie technik przeglądów zostaje dostarczony do organizacji, szczególnie dla praktyk sformalizowanych
+ kierownictwo wspiera dobry proces przeglądu (np. rezerwując czas potrzebny dla przeglądu w planie projektu)
+ jest wola uczenia się i wprowadzania procesu ulepszania

**Statyczna analiza przy użyciu narzędzi**

Celem analizy statycznej jest znajdowanie defektów w kodzie źródłowym oprogramowania i modelach oprogramowania. Analiza statystyczna jest wykonywana bez uruchamiania oprogramowania narzędziem testowym, w przeciwieństwie do technik dynamicznych. Analiza statyczna może zlokalizować defekty, które są trudne do znalezienia podczas testowania. Podobnie jak przeglądy, analiza statystyczna znajduje błędy niż ich konsekwencje. Narzędzia analizy statycznej analizuje kod programu (np. sprawdzenie przepływu danych) i generuje raport w postaci HTML lub XML.
Zalety analizy statycznej:
+ wczesne wykrycie defektów przed wykonaniem testów
+ wczesne ostrzeżenia o podejrzanym zachowaniu kodu lub projektu, poprzez sprawdzenie metryk takich jak pomiary komplikacji
+ identyfikacja defektów trudnych do znalezienia w dynamicznym testowaniu
+ sprawdzenie zależności i nieprawidłowości w modelach oprogramowania, np. w linkach
+ poprawienie łatwości serwisowania kodu i projektu
+ zapobieganie występowania defektów, jeśli twórcy oprogramowania uczą się na własnych błędach.

Typowe błędy wykryte przez narzędzia analizy statycznej:
+ referencje do niezdefiniowanej zmiennej
+ nieprawidłowe funkcjonowanie interfejsów pomiędzy modułami i komponentami
+ nigdy nieużywane zmienne
+ martwy kod (nigdy nie wykonywany)
+ naruszenie standardów kodowania
+ niedostateczne zabezpieczenie danych
+ naruszenie poprawności gramatycznej kodu i modelów oprogramowania.

Narzędzia analizy statystycznej są zazwyczaj używane przez programistów (sprawdzenie zdefiniowanych zasad lub standardów programistycznych) przed i podczas testów komponentów i integracji, oraz przez architektów podczas modelowania oprogramowania. Narzędzia analizy statycznej mogą generować olbrzymią liczbę ostrzeżeń, które musza być poprawnie zarządzane by umożliwić efektywne użycie narzędzi.

Kompilatory mogą oferować wsparcie analizy statystycznej, wliczając w to kalkulacje metryk.

testerzy.pl

Wyłączne prawa autorskie do tego dokumentu posiadają „testerzy.pl". Rozpowszechnianie dla celów komercyjnych jak i niekomercyjnych jest dozwolone tylko pod warunkiem podania źródła.