# WYKŁAD 1

## Spis treści

**PODSTAWY TESTOWANIA ..... 2**
1.1 CO TO JEST TESTOWANIE? ..... 2
1.1.1. Typowe cele testowania ..... 2
1.1.2. Testowanie a debugowanie ..... 3
1.2 DLACZEGO TESTOWANIE JEST NIEZBĘDNE? ..... 3
1.2.1. Znaczenie testowania dla powodzenia projektu ..... 4
1.2.2. Zapewnienie jakości a testowanie ..... 4
1.2.3. Pomyłki, defekty i awarie ..... 5
1.2.4. Defekty, podstawowe przyczyny oraz skutki ..... 6
1.3 SIEDEM ZASAD TESTOWANIA ..... 6
1.4 PROCES TESTOWY ..... 8
1.4.1. Proces testowy w kontekście ..... 8
1.4.2. Czynności i zadania testowe ..... 9
1.4.3. Produkty pracy związane z testowaniem ..... 14
1.4.4. Śledzenie powiązań między podstawą testów a produktami pracy związanymi z testowaniem ..... 17
1.5. PSYCHOLOGIA TESTOWANIA ..... 18
1.5.1. Psychologia człowieka a testowanie ..... 18
1.5.2. Różnice w sposobie myślenia testerów i programistów ..... 19

# PODSTAWY TESTOWANIA

## 1.1 Co to jest testowanie?

Systemy oprogramowania są nieodłączną częścią naszego życia we wszystkich jego obszarach — od aplikacji biznesowych (np. w bankowości) po produkty użytkowe (np. samochody). Jednocześnie większość z nas miała zapewne do czynienia z oprogramowaniem, które nie zadziałało tak, jak powinno. Nieprawidłowe funkcjonowanie oprogramowania może powodować wiele problemów, w tym straty finansowe, stratę czasu, utratę reputacji firmy, a nawet utratę zdrowia lub życia. Testowanie oprogramowania pozwala ocenić jego **jakość** i **zmniejszyć ryzyko** wystąpienia awarii podczas eksploatacji.

Powszechnie uważa się, że testowanie polega wyłącznie na wykonywaniu testów, czyli uruchamianiu oprogramowania i sprawdzaniu uzyskanych rezultatów. Jak jednak opisano w podrozdziale 1.4., testowanie oprogramowania to proces obejmujący czynności, które wykraczają poza samo wykonywanie testów. W skład procesu testowego wchodzą również takie czynności jak: *planowanie, analiza, projektowanie i implementacja testów, raportowanie o postępie i wynikach testów oraz dokonywanie oceny jakości przedmiotu testów*. Testowanie może wymagać uruchomienia testowanego modułu lub systemu – mamy wtedy do czynienia z tzw. testowaniem **dynamicznym**. Można również wykonywać testy bez uruchamiania testowanego obiektu – takie testowanie nazywa się testowaniem **statycznym**. A zatem testowanie obejmuje również przegląd produktów pracy takich jak: wymagania, historyjki użytkownika i kod źródłowy.

Inne nieporozumienie polega na postrzeganiu testowania jako czynności skupionej wyłącznie na weryfikacji wymagań, historyjek użytkownika lub innych form specyfikacji. Chociaż w ramach testowania rzeczywiście sprawdza się, czy system spełnia wyspecyfikowane wymagania, to jednak przeprowadza się również walidację, której zadaniem jest sprawdzenie, czy system spełnia wymagania użytkowników oraz inne potrzeby interesariuszy w swoim środowisku operacyjnym.

### 1.1.1. Typowe cele testowania

W przypadku każdego projektu **cele testowania** mogą być realizowane między innymi poprzez następujące czynności:

- dokonywanie oceny produktów pracy, takich jak: wymagania, historyjki użytkownika, projekt i kod;
- sprawdzanie, czy zostały spełnione wszystkie wyspecyfikowane wymagania;
- sprawdzanie, czy przedmiot testów jest kompletny i działa zgodnie z oczekiwaniami użytkowników i innych interesariuszy;
- budowanie zaufania do poziomu jakości przedmiotu testów;
- zapobieganie defektom;

• wykrywanie defektów i awarii;

• dostarczanie interesariuszom informacji niezbędnych do podejmowania świadomych decyzji (dotyczących zwłaszcza poziomu jakości przedmiotu testów);

• obniżanie poziomu ryzyka związanego z jakością oprogramowania (np. ryzyka wystąpienia niewykrytych wcześniej awarii podczas eksploatacji);

• przestrzeganie wymagań wynikających z umów, przepisów prawa i norm/standardów i/lub sprawdzanie, czy obiekt testów jest zgodny z tymi wymaganiami lub standardami.

Cele testowania mogą różnić się w zależności od kontekstu testowanego modułu lub systemu, poziomu testów oraz modelu cyklu życia oprogramowania. Poniżej podano kilka przykładowych różnic.

- W przypadku **testowania modułowego** jednym z celów może być wykrycie jak największej liczby awarii, a w rezultacie wczesne zidentyfikowanie i usunięcie powodujących je defektów. Innym celem może być zwiększenie pokrycia kodu przez testy modułowe.

- W przypadku **testowania akceptacyjnego** jednym z celów może być potwierdzenie, że system działa zgodnie z oczekiwaniami i spełnia stawiane mu wymagania. Innym celem tego rodzaju testowania może być dostarczenie interesariuszom informacji na temat ryzyka, jakie wiąże się z przekazaniem systemu do eksploatacji w danym momencie.

### 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 czynność związana 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. W niektórych przypadkach testerzy są odpowiedzialni za przeprowadzenie testu początkowego oraz końcowego testu potwierdzającego, a programiści są odpowiedzialni za przeprowadzenie debugowania oraz związanego z nim testowania modułowego. Jednakże w przypadku zwinnego wytwarzania oprogramowania i niektórych innych cykli życia testerzy mogą być również zaangażowani w debugowanie i testowanie modułowe.

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

## 1.2 Dlaczego testowanie jest niezbędne?

Rygorystyczne testowanie modułów i systemów oraz związanej z nimi dokumentacji może pomóc w zmniejszeniu ryzyka wystąpienia awarii podczas eksploatacji oprogramowania. Wykrycie, a następnie usunięcie defektów, przyczynia się do podniesienia jakości modułów lub systemów. Ponadto testowanie oprogramowania może być niezbędne do spełnienia wymagań wynikających z umów, przepisów prawa

bądź norm/standardów branżowych.

### 1.2.1. Znaczenie testowania dla powodzenia projektu

W historii informatyki znanych jest wiele przykładów oprogramowania i systemów, które zostały przekazane do eksploatacji, ale na skutek defektów uległy awarii lub z innych powodów nie zaspokoiły potrzeb interesariuszy. Dzięki odpowiednim technikom testowania — stosowanym w sposób fachowy na odpowiednich poziomach testów i w odpowiednich fazach cyklu życia oprogramowania — częstotliwość występowania tego rodzaju problemów można jednak ograniczyć. Poniżej przedstawiono kilka przykładów takich zastosowań.

- Zaangażowanie testerów w przeglądy wymagań lub w doprecyzowywanie historyjek użytkownika pomaga wykryć defekty wpowyższych produktach pracy. Zidentyfikowanie i usunięcie tych defektów zmniejsza ryzyko wytworzenia niepoprawnej lub nietestowalnej funkcjonalności.

- Ścisła współpraca między testerami a projektantami systemu na etapie prac projektowych pozwala obu stronom lepiej zrozumieć projekt i sposób jego testowania. Lepsza znajomość tematu przekłada się na mniejsze ryzyko wystąpienia zasadniczych problemów w projekcie, a także pozwala zidentyfikować przypadki testowe w początkowym etapie projektu.

- Ścisła współpraca testerów z programistami na etapie tworzenia kodu pozwala obu stronom lepiej zrozumieć kod i sposób jego testowania. Większa wiedza w tym zakresie pozwala zmniejszyć ryzyko wystąpienia defektów w kodzie i spowodowanych przez nie awarii w testach.

- Weryfikacja i walidacja oprogramowania przez testerów przed przekazaniem go do eksploatacji pozwala wykryć defekty mogące prowadzić do awarii, które w przeciwnym razie mogłyby zostać przeoczone. Ułatwia także usuwanie związanych z nimi defektów (debugowanie). W rezultacie rośnie prawdopodobieństwo, że oprogramowanie zaspokoi potrzeby interesariuszy i spełni stawiane mu wymagania.

### 1.2.2. Zapewnienie jakości a testowanie

Testowanie jest często utożsamiane z zapewnieniem jakości (ang. *quality assurance*), ale w rzeczywistości są to dwa oddzielne (chociaż powiązane ze sobą) procesy, które zawierają się w szerszym pojęciu „zarządzania jakością” (ang. *quality management*). Zarządzanie jakością obejmuje wszystkie czynności mające na celu kierowanie i nadzorowanie działań organizacji w dziedzinie jakości. Elementami zarządzania jakością są między innymi zapewnienie jakości i kontrola jakości. Zapewnienie jakości skupia się zazwyczaj na prawidłowym przestrzeganiu właściwych procesów w celu uzyskania pewności, że zostaną osiągnięte odpowiednie poziomy jakości.

Jeśli procesy są wykonywane prawidłowo, powstające w ich wyniku produkty pracy mają zwykle wyższą jakość, co przyczynia się do zapobiegania defektom. Duże znaczenie dla skutecznego zapewnienia jakości mają również wykorzystanie procesu analizy przyczyny podstawowej do wykrywania i usuwania przyczyn defektów oraz prawidłowe wdrażanie wniosków ze spotkań

retrospektywnych w celu doskonalenia procesów.

Kontrola jakości obejmuje cały szereg czynności, włączając w to czynności testowe, które wspierają osiągnięcie odpowiednich poziomów jakości. Czynności testowe są ważnym elementem procesu wytwarzania lub pielęgnacji oprogramowania. Prawidłowy przebieg tego procesu (w tym testowania) jest istotny z punktu widzenia zapewnienia jakości, w związku z czym zapewnienie jakości wspiera właściwe testowanie. Testowanie przyczynia się do osiągnięcia wymaganej jakości produktów pracy na kilka różnych sposobów.

### 1.2.3. Pomyłki, defekty i awarie

Na skutek pomyłki (błędu) człowieka w kodzie oprogramowania lub w innym związanym z nim produkcie pracy może powstać defekt (inaczej zwany usterką lub pluskwą). Pomyłka skutkująca wprowadzeniem defektu w jednym produkcie pracy może spowodować błąd skutkujący wprowadzeniem defektu w innym, powiązanym produkcie pracy. Przykładem takiej sytuacji jest pomyłka popełniona podczas pozyskiwania wymagań, która może prowadzić do defektu w wymaganiach, co spowoduje pomyłkę programisty skutkującą wprowadzeniem defektu w kodzie.

Wykonanie kodu zawierającego defekt może spowodować awarię, ale nie musi dziać się tak w przypadku każdego defektu. Niektóre defekty powodują awarię na przykład tylko po wprowadzeniu ściśle określonych danych wejściowych bądź na skutek wystąpienia określonych warunków wstępnych, które mogą zaistnieć bardzo rzadko lub nigdy.

Pomyłki mogą pojawiać się z wielu powodów, takich jak:

- presja czasu;
- omylność człowieka;
- brak doświadczenia lub niedostateczne umiejętności uczestników projektu;
- problemy z wymianą informacji między uczestnikami projektu (w tym nieporozumienia dotyczące rozumienia wymagań i dokumentacji projektowej);
- złożoność kodu, projektu, architektury, rozwiązywanego problemu i/lub wykorzystywanej technologii;
- nieporozumienia dotyczące interfejsów wewnątrz systemu i między systemami, zwłaszcza w przypadku dużej liczby tych systemów;
- stosowanie nowych, nieznanych technologii.

Awarie – poza tymi wynikającymi z defektów w kodzie – mogą być również spowodowane warunkami środowiskowymi. Przykładem takich sytuacji i warunków są: promieniowanie, pole elektromagnetyczne lub zanieczyszczenie środowiska, które mogą spowodować wystąpienie defektów w oprogramowaniu wewnętrznym (ang. firmware) lub wpłynąć na działanie oprogramowania poprzez zmianę parametrów pracy sprzętu.

Nie wszystkie nieoczekiwane wyniki testów oznaczają awarie. Wynik fałszywie pozytywny może być, między innymi, skutkiem błędów związanych z wykonaniem testów, defektów w danych testowych, środowisku testowym, w innych testaliach itp. Podobne problemy mogą być przyczyną sytuacji odwrotnej – wyniku fałszywie negatywnego, czyli sytuacji, w której testy nie wykrywają defektu, który powinny wykryć. Wyniki fałszywie pozytywne są raportowane jako defekty, których w rzeczywistości nie ma.

### 1.2.4. Defekty, podstawowe przyczyny oraz skutki

Podstawowa przyczyna defektu to pierwotny powód, w wyniku którego defekt ten powstał. Przeanalizowanie defektu wcelu zidentyfikowania podstawowej przyczyny pozwala zredukować wystąpienia podobnych defektów wprzyszłości. Ponadto analiza przyczyny podstawowej — skupiająca się na najważniejszych, pierwotnych przyczynach defektów — może prowadzić do udoskonalenia procesów, co może z kolei przełożyć się na dalsze zmniejszenie liczby defektów w przyszłości.

Załóżmy na przykład, że defekt w jednym z wierszy kodu powoduje nieprawidłowe wypłacanie odsetek, co z kolei wiąże się z reklamacjami klientów. Wadliwy kod został napisany na podstawie historyjki użytkownika, która była niejednoznaczna, ponieważ właściciel produktu źle zrozumiał zasady naliczania odsetek. W związku z tym, jeśli przy naliczaniu odsetek występuje duży procent defektów, których podstawową przyczyną są podobne nieporozumienia, rozwiązaniem może być przeszkolenie właścicieli produktów w zakresie tego rodzaju obliczeń, co pozwoli zmniejszyć w przyszłości liczbę podobnych defektów.

W powyższym przykładzie reklamacje klientów są skutkami, nieprawidłowa wypłata odsetek jest awarią, a nieprawidłowe obliczenia wykonywane przez kod są defektem wynikającym z pierwotnego defektu, jakim była niejednoznaczność historyjki użytkownika. Podstawową przyczyną pierwotnego defektu były braki wiedzy właściciela produktu, na skutek których popełnił on pomyłkę przy pisaniu historyjki użytkownika.

## 1.3 Siedem zasad testowania

Przez ostatnie kilkadziesiąt lat zaproponowano cały szereg zasad testowania, które dostarczają ogólnych wskazówek mających zastosowanie do wszystkich rodzajów testowania.

**1. Testowanie ujawnia usterki, ale nie może dowieść ich braku**

Testowanie może wykazać obecność defektów, ale nie może dowieść, że oprogramowanie jest od nich wolne. Tym samym testowanie zmniejsza prawdopodobieństwo, że w oprogramowaniu pozostaną niewykryte defekty, ale sam fakt niewykrycia defektów nie stanowi dowodu poprawności oprogramowania.

**2. Testowanie gruntowne jest niemożliwe**

Przetestowanie wszystkiego (tj. wszystkich kombinacji danych wejściowych i warunków wstępnych) jest

możliwe tylko w najprostszych przypadkach. W związku z tym, zamiast podejmować próbę testowania gruntownego, należy odpowiednio ukierunkować wysiłki związane z testowaniem na zastosowanie analizy ryzyka, technik testowania i priorytetyzacji.

**3. Wczesne testowanie oszczędza czas i pieniądze**

Aby wcześnie wykryć defekty, należy rozpocząć testowanie statyczne i dynamiczne na jak najwcześniejszym etapie cyklu życia oprogramowania. Wczesne testowanie jest niekiedy nazywane „przesunięciem wlewo" (ang. *shift left*). Wykonywanie testów na wczesnym etapie cyklu życia oprogramowania pozwala ograniczyć lub wyeliminować kosztowne zmiany.

**4. Kumulowanie się defektów**

Zwykle większość defektów wykrytych podczas testowania przed przekazaniem oprogramowania do eksploatacji lub większość awarii występujących w fazie eksploatacji występuje lub ma swoje źródło w niewielkiej liczbie modułów. W rezultacie przewidywane skupiska defektów i skupiska defektów faktycznie zaobserwowane na etapie testowania lub eksploatacji są ważnym elementem analizy ryzyka, którą przeprowadza się wcelu odpowiedniego ukierunkowania wysiłków związanych z testowaniem.

**5. Paradoks pestycydów**

Ciągłe powtarzanie tych samych testów prowadzi do sytuacji, w której przestają one w pewnym momencie wykrywać nowe defekty. Aby móc wykrywać nowe defekty, może być konieczne zmodyfikowanie dotychczasowych testów i danych testowych, a także napisanie nowych testów. Niezmieniane testy tracą z czasem zdolność do wykrywania defektów, podobnie jak pestycydy po pewnym czasie nie są zdolne do eliminowania szkodników. W niektórych przypadkach — takich jak automatyczne testowanie regresji — paradoks pestycydów może być korzystny, ponieważ pozwala potwierdzić, że liczba defektów związanych z regresją jest niewielka.

**6. Testowanie zależy od kontekstu**

Testowanie wykonuje się w różny sposób w różnych kontekstach. Na przykład oprogramowanie sterujące systemami przemysłowymi, które jest krytyczne ze względów bezpieczeństwa, testuje się inaczej niż aplikację mobilną sklepu internetowego. Innym przykładem może być odmienny sposób przeprowadzania testów w projektach zwinnych i w tych prowadzonych zgodnie z modelem sekwencyjnym cyklu życia.

**7. Przekonanie o braku błędów jest błędem**

Niektóre organizacje oczekują, że testerzy będą w stanie uruchomić wszystkie możliwe testy i wykryć wszystkie możliwe defekty, ale powyższe zasady (odpowiednio nr 2 i 1) pokazują, że jest to niemożliwe. Co więcej, błędnym jest przekonanie, że samo znalezienie i naprawienie dużej liczby defektów zapewni pomyślne wdrożenie systemu. Na przykład bardzo dokładne testowanie wszystkich wyspecyfikowanych wymagań i naprawienie wszystkich znalezionych defektów wciąż może nas nie uchronić od zbudowania

systemu trudnego w obsłudze, który nie spełni wymagań i oczekiwań użytkowników lub będzie miał gorsze parametry od konkurencyjnych rozwiązań.

## 1.4 Proces testowy

Nie ma jednego uniwersalnego procesu testowania oprogramowania, ale istnieją typowe czynności testowe, bez stosowania których nie zostaną zrealizowane ustalone dla testowania cele. Czynności te składają się na proces testowy. Dobór procesu testowego do konkretnego oprogramowania w konkretnej sytuacji zależy od wielu czynników. Organizacja może w swojej strategii testowej zdefiniować, które czynności testowe wchodzą w skład tego procesu, jak czynności testowe mają być zaimplementowane oraz w którym momencie cyklu życia oprogramowania powinny mieć miejsce.

### 1.4.1. Proces testowy w kontekście

Przykładowe czynniki kontekstowe wpływające na proces testowy w organizacji to:

- wykorzystywany model cyklu życia oprogramowania i metodyki projektowe;
- rozważane poziomy testów i typy testów;
- czynniki ryzyka produktowego i projektowego;
- dziedzina biznesowa;
- ograniczenia operacyjne, w tym w szczególności:
    - budżety i zasoby;
    - harmonogramy;
    - złożoność procesu lub projektu;
    - wymagania wynikające z umów i przepisów;
- polityka testów i praktyki obowiązujące w organizacji;
- wymagane normy/standardy wewnętrzne i zewnętrzne.

W kolejnych punktach opisano ogólne aspekty organizacyjnego procesu testowego w relacji do:

- czynności i zadań testowych;
- produktów pracy związanych z testowaniem;
- możliwości śledzenia powiązań pomiędzy podstawą testów a produktami pracy związanymi z testowaniem.

Dobrą praktyką jest zdefiniowanie mierzalnych kryteriów pokrycia dotyczących podstawy testów (w odniesieniu do każdego rozważanego poziomu lub typu testów). Kryteria pokrycia mogą w praktyce pełnić funkcję kluczowych wskaźników wydajności (ang. *Key Performance Indicators* — KPI)

sprzyjających wykonywaniu określonych czynności i pozwalających wykazać osiągnięcie celów testowania oprogramowania.

Przykładem ilustrującym tę praktykę jest podstawa testów dla aplikacji mobilnej, która może zawierać listę wymagań oraz listę wspieranych urządzeń mobilnych. Każde wymaganie i każde wspierane urządzenie jest elementem podstawy testów. Kryteria pokrycia mogą wymagać co najmniej jednego testu dla każdego elementu podstawy testów. Po wykonaniu testów ich wyniki są źródłem informacji dla interesariuszy, czy określone wymagania zostały spełnione oraz czy na wspieranych urządzeniach zaobserwowano awarie.

### 1.4.2. Czynności i zadania testowe

W procesie testowym wyróżnia się następujące, główne grupy czynności:

- planowanie testów;
- monitorowanie testów i nadzór nad testami;
- analiza testów;
- projektowanie testów;
- implementacja testów;
- wykonywanie testów;
- ukończenie testów.

Każda grupa składa się z czynności, które opisano w kolejnych podpunktach. Ponadto poszczególne czynności w każdej z grup mogą składać się z kilku pojedynczych zadań różniących się w zależności od projektu lub wersji oprogramowania.

Należy również zaznaczyć, że chociaż wiele grup czynności może sprawiać wrażenie logicznie uszeregowanych, często są one realizowane metodą iteracyjną. Przykładem takiej praktyki jest model zwinnego wytwarzania oprogramowania, który opiera się na krótkich iteracjach obejmujących projektowanie, tworzenie i testowanie produktu. Iteracje odbywają się w sposób ciągły i są objęte ciągłym planowaniem. W rezultacie czynności testowe są również wykonywane w ramach tego podejścia do wytwarzania oprogramowania wsposób ciągły i iteracyjny. Nawet w przypadku stosowania metod sekwencyjnych, w których czynności są wykonywane krokowo w logicznej kolejności, pewne elementy nakładają się na siebie, są wykonywane łącznie lub równocześnie bądź są pomijane. W związku z powyższym zwykle konieczne jest dostosowanie głównych czynności do kontekstu danego systemu lub projektu.

#### Planowanie testów

Planowanie testów obejmuje czynności, których celem jest zdefiniowanie celów testowania oraz określenie podejścia do osiągania celów testowania w granicach wyznaczonych przez kontekst (np. określenie odpowiednich technik testowania i zadań testowych oraz sformułowanie harmonogramu testów, który umożliwi dotrzymanie wyznaczonego terminu). Plany testów mogą być następnie korygowane na podstawie informacji zwrotnych z monitorowania i nadzoru.

## Monitorowanie testów i nadzór nad testami

Monitorowanie testów polega na ciągłym porównywaniu rzeczywistego z zaplanowanym postępem testowania przy użyciu miar specjalnie w tym celu zdefiniowanych w planie testów. Nadzór nad testami polega na podejmowaniu działań, które są niezbędne do osiągnięcia celów wyznaczonych w planie testów (z uwzględnieniem jego ewentualnych aktualizacji). Elementem wspomagającym monitorowanie testów i nadzór nad testami jest ocena kryteriów wyjścia, które w przypadku niektórych cykli życia są również nazywane „definicją ukończenia". Ocena kryteriów wyjścia dla wykonania testów na określonym poziomie testów może obejmować:

- sprawdzenie rezultatów testów i dziennika testów pod kątem określonych kryteriów pokrycia;
- oszacowanie poziomu jakości modułu lub systemu na podstawie rezultatów testów i dziennika testów;
- ustalenie, czy są konieczne dalsze testy (np. w przypadku nieosiągnięcia przez dotychczas wykonane testy pierwotnie założonego poziomu pokrycia ryzyka produktowego, co wiąże się z koniecznością napisania i wykonania dodatkowych testów).

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

## Analiza testów

Celem grupy czynności w analizie testów jest przeanalizowanie podstawy testów w celu zidentyfikowania testowalnych cech i zdefiniowania związanych z nimi warunków testowych. Innymi słowy, analiza testów służy do ustalenia tego, „co" należy przetestować (w kategoriach mierzalnych kryteriów pokrycia).

Główne czynności wykonywane w ramach analizy testów to:

- dokonywanie analizy podstawy testów właściwej dla rozważanego poziomu testów, np.:
    - specyfikacji wymagań, takich jak: wymagania biznesowe, wymagania funkcjonalne, wymagania systemowe, historyjki użytkownika, opowieści (ang. epic), przypadki użycia lub podobne produkty pracy, które określają pożądane zachowanie funkcjonalne i niefunkcjonalne modułu lub systemu;
    - informacji dotyczących projektu i implementacji, takich jak: diagramy lub dokumenty opisujące architekturę systemu lub oprogramowania, specyfikacje projektowe, przepływy wywołań, modele oprogramowania (np. diagramy UML lub diagramy związków encji), specyfikacje interfejsów lub podobne produkty pracy, które określają strukturę modułu lub systemu;

- implementacji samego modułu lub systemu, w tym kodu, metadanych, zapytań do bazy danych oraz interfejsów;
- raportów z analizy ryzyka, które mogą dotyczyć aspektów funkcjonalnych, niefunkcjonalnych i strukturalnych modułu lub systemu;

- dokonywanie oceny testowalności podstawy testów i elementów testowych wcelu zidentyfikowania często występujących typów defektów, które mogą powodować problemy z testowalnością, takich jak:

- niejednoznaczności;
- pominięcia;
- niespójności;
- nieścisłości;
- sprzeczności;
- nadmiarowe (zbędne) instrukcje;

- identyfikowanie cech i zbiorów cech, które mają zostać przetestowane;

- definiowanie warunków testowych w odniesieniu do poszczególnych cech oraz określenie ich priorytetów na podstawie analizy podstawy testów — z uwzględnieniem parametrów funkcjonalnych, niefunkcjonalnych i strukturalnych, innych czynników biznesowych i technicznych oraz poziomów ryzyka;

- stworzenie możliwości dwukierunkowego śledzenia powiązań między elementami podstawy testów a związanymi z nimi warunkami testowymi.

Zastosowanie technik czarnoskrzynkowych, białoskrzynkowych oraz technik opartych na doświadczeniu w ramach analizy testów pomaga zmniejszyć prawdopodobieństwo pominięcia ważnych warunków testowych oraz zdefiniować bardziej precyzyjne i dokładne warunki testowe.

W niektórych przypadkach w wyniku analizy testów definiowane są warunki testowe, które mają być wykorzystywane jako cele testów wkartach opisów testów. Karty opisu testów są typowymi produktami pracy w niektórych odmianach testowania opartego na doświadczeniu. Jeśli istnieje śledzenie powiązań między wspomnianymi powyżej celami testów a podstawą testów, możliwy jest pomiar pokrycia uzyskanego w trakcie testowania opartego na doświadczeniu.

Identyfikowanie defektów na etapie analizy testów jest istotną potencjalną korzyścią, zwłaszcza gdy nie stosuje się żadnego innego procesu przeglądu i/lub gdy proces testowy jest ściśle powiązany z procesem przeglądu. Czynności wykonywane w ramach analizy testów pozwalają zweryfikować, czy wymagania są spójne, prawidłowo wyrażone i kompletne, a także sprawdzić, czy właściwie odzwierciedlają one potrzeby klienta, użytkowników i innych interesariuszy. Warto w tym miejscu podać przykład takich technik jak: wytwarzanie sterowane zachowaniem (ang. *Behavior Driven Development* — BDD) i wytwarzanie sterowane testami akceptacyjnymi (ang. *Acceptance Test Driven Development* — ATDD), które obejmują generowanie warunków testowych i przypadków testowych na podstawie historyjek

użytkownika ikryteriów akceptacji przed rozpoczęciem tworzenia kodu. Przewidują one również weryfikację i walidację historyjek użytkownika i kryteriów akceptacji oraz wykrywanie występujących w nich defektów

### Projektowanie testów

Podczas projektowania testów warunki testowe są przekształcane w przypadki testowe wysokiego poziomu, zbiory takich przypadków testowych oraz w inne testalia. W związku z tym o ile analiza testów odpowiada na pytanie: „co należy przetestować", o tyle projektowanie testów odpowiada na pytanie: „jak należy testować".

Główne czynności wykonywane w ramach projektowania testów to:

- projektowanie przypadków testowych i zbiorów przypadków testowych oraz określenie ich priorytetów;
- identyfikowanie danych testowych niezbędnych do obsługi warunków testowych i przypadków testowych;
- projektowanie środowiska testowego oraz zidentyfikowanie wszelkich niezbędnych narzędzi i elementów infrastruktury;
- tworzenie możliwości dwukierunkowego śledzenia powiązań między podstawą testów, warunkami testowymi, przypadkami testowymi i procedurami testowymi.

Rozwinięcie warunków testowych w przypadki testowe i zbiory przypadków testowych na etapie projektowania testów wymaga często użycia technik testowania.

Tak jak w przypadku analizy testów, projektowanie testów może również skutkować identyfikacją podobnych typów defektów w podstawie testów, co jest ważną potencjalną korzyścią.

### Implementacja testów

Podczas implementacji testów tworzone i/lub dokańczane są testalia niezbędne do wykonania testów, w tym szeregowanie przypadków testowych w ramach procedur testowych. W związku z tym, o ile projektowanie testów odpowiada na pytanie: „jak testować?", o tyle implementacja testów odpowiada na pytanie: „czy mamy wszystko, co jest potrzebne do uruchomienia testów?".

Główne czynności wykonywane w ramach implementacji testów to:

- opracowanie procedur testowych i określenie ich priorytetów oraz, potencjalnie, utworzenie skryptów testów automatycznych;
- utworzenie zestawów testowych (na podstawie procedur testowych) oraz skryptów testów

automatycznych (jeśli istnieją testy automatyczne);

- uporządkowanie zestawów testowych w harmonogram wykonywania testów w sposób zapewniający efektywny przebieg całego procesu.

- zbudowanie środowiska testowego (włączając w to - jeśli to konieczne - jarzma testowe, wirtualizację usług, symulatory i inne elementy infrastrukturalne) oraz sprawdzenie, czy zostało ono poprawnie skonfigurowane;

- przygotowanie danych testowych i sprawdzenie, czy zostały poprawnie załadowane do środowiska testowego;

- zweryfikowanie i zaktualizowanie możliwości dwukierunkowego śledzenia powiązań między podstawą testów, warunkami testowymi, przypadkami testowymi, procedurami testowymi i zestawami testowymi.

Zadania związane z projektowaniem i implementacją testów są często łączone.

Etapy projektowania i implementacji testów mogą być realizowane i dokumentowane w ramach wykonywania testów — zwłaszcza w przypadku testowania eksploracyjnego oraz innych typów testowania opartego na doświadczeniu. Testowanie eksploracyjne może odbywać się na podstawie kart opisu testów (sporządzanych w ramach analizy testów), przy czym testy eksploracyjne wykonywane są natychmiast po ich zaprojektowaniu i zaimplementowaniu.

**Wykonywanie testów**

Podczas wykonywania testów uruchamiane są zestawy testowe, zgodnie z harmonogramem wykonania testów. Główne czynności przeprowadzane w ramach wykonywania testów to:

- zarejestrowanie danych identyfikacyjnych i wersji elementów testowych bądź przedmiotu testów, narzędzi testowych i testaliów;

- wykonywanie testów ręcznie lub przy użyciu narzędzi do wykonywania testów;

- porównanie rzeczywistych wyników testów z oczekiwanymi;

- przeanalizowanie anomalii w celu ustalenia ich prawdopodobnych przyczyn (np. awarie mogą być wynikiem defektów w kodzie, ale mogą się pojawić również wyniki fałszywie pozytywne.

- raportowanie defektów oparte na obserwowanych awariach.

- zarejestrowanie (zalogowanie) wyniku wykonania testów (np. zaliczenie, niezaliczenie, test blokujący);

- powtórzenie czynności testowych w wyniku działań podjętych w związku z wystąpieniem anomalii

albo w ramach zaplanowanego testowania (np. testowania potwierdzającego, wykonywania poprawionego testu i/lub testowania regresji);

- zweryfikowanie i zaktualizowanie możliwości dwukierunkowego śledzenia powiązań między podstawą testów, warunkami testowymi, przypadkami testowymi, procedurami testowymi i wynikami testów.

**Ukończenie testów**

Ukończenie testów polega na zebraniu danych pochodzących z wykonanych czynności testowych wcelu skonsolidowania zdobytych doświadczeń, testaliów oraz innych stosownych informacji. Czynności związane z ukończeniem testów są wykonywane w momencie osiągnięcia kamieni milowych projektu, takich jak: przekazanie systemu lub oprogramowania do eksploatacji, zakończenie realizacji (lub anulowanie) projektu, zakończenie iteracji projektu zwinnego (np. w ramach spotkania retrospektywnego), ukończenie poziomu testów bądź zakończenie prac nad wydaniem pielęgnacyjnym (ang. maintenance release).

Główne czynności wykonywane w ramach ukończenia testów to:

- sprawdzenie, czy wszystkie raporty o defektach są zamknięte oraz wprowadzenie żądań zmian lub pozycji do rejestru produktu w odniesieniu do wszelkich defektów, które nie zostały rozwiązane do momentu zakończenia wykonywania testów;
- utworzenie sumarycznego raportu z testów, który zostanie przekazany interesariuszom;
- dla wersji końcowych: zarchiwizowanie środowiska testowego, danych testowych, infrastruktury testowej oraz innych testaliów, do ponownego wykorzystania w przyszłości;
- przekazanie testaliów zespołom odpowiedzialnym za pielęgnację, innym zespołom projektowym i/lub innym interesariuszom, którzy mogą odnieść korzyść z ich użycia;
- przeanalizowanie zdobytych doświadczeń omawianych na wszystkich etapach projektu w celu ustalenia, jakie zmiany będą konieczne w przypadku przyszłych iteracji, wydań i projektów;
- wykorzystanie zebranych informacji do zwiększenia dojrzałości procesu testowego.

### 1.4.3. Produkty pracy związane z testowaniem

Produkty pracy związane z testowaniem powstają w ramach procesu testowego. Podobnie jak sposób implementacji procesu testowego, również typy produktów pracy powstających w wyniku tego procesu oraz nadawane im nazwy różnią się znacznie w poszczególnych organizacjach. W niniejszym sylabusie przyjęto opisaną powyżej definicję procesu testowego oraz produktów pracy. Punktem odniesienia dla produktów pracy związanych ztestowaniem może być również standard międzynarodowy ISO/IEC/IEEE 29119-3.

Wiele z produktów pracy związanych ztestowaniem, które opisano w tym punkcie, może być tworzonych i zarządzanych przy użyciu narzędzi do zarządzania testami oraz narzędzi do zarządzania defektami.

**Produkty pracy planowania testów**

Wśród produktów pracy powstających na etapie planowania testów można zwykle wyróżnić jeden lub kilka planów testów. Plan testów zawiera informacje na temat podstawy testów, z którą zostaną powiązane za pośrednictwem informacji śledzenia inne produkty pracy związane z testowanie. Ponadto określa się w nim kryteria wyjścia (definicję ukończenia), które będą stosowane w ramach monitorowania testów i nadzoru nad testami.

**Produkty pracy monitorowania testów i nadzoru nad testami**

Typowymi produktami pracy związanymi z monitorowaniem testów i nadzorem nad testami są różnego rodzaju raporty z testów, w tym raporty o postępie testów (tworzone na bieżąco i/lub w regularnych odstępach czasu) oraz sumaryczne raporty końcowe z testów (tworzone w momencie osiągnięcia poszczególnych kamieni milowych projektu). Raporty z testów powinny zawierać szczegółowe informacje na temat dotychczasowego przebiegu procesu testowego (w tym podsumowanie rezultatów wykonywania testów, gdy zostaną one udostępnione) i powinny być przedstawiane w formie dostosowanej do potrzeb konkretnych odbiorców.

Produkty pracy monitorowania testów i nadzoru nad testami powinny również odnosić się do kwestii dotyczących zarządzania projektem, takich jak: ukończenie zadań, alokacja i zużycie zasobów czy też pracochłonność.

**Produkty pracy analizy testów**

Produkty pracy związane z analizą testów obejmują zdefiniowane i uszeregowane według priorytetów warunki testowe, przy czym pożądana jest możliwość dwukierunkowego śledzenia powiązań między tymi warunkami a pokrywanymi przez nie elementami podstawy testów. W przypadku testowania eksploracyjnego, w wyniku analizy testów mogą również powstawać karty opisu testów. Ponadto analiza testów może prowadzić do wykrycia i zgłoszenia defektów w podstawie testów.

**Produkty pracy projektowania testów**

W wyniku projektowania testów powstają przypadki testowe i zestawy testowe zdefiniowane na etapie analizy testów.

Dobrą praktyką jest projektowanie przypadków testowych wysokiego poziomu, które nie zawierają konkretnych wartości danych wejściowych i oczekiwanych rezultatów. Wysokopoziomowe przypadki testowe można wykorzystywać wielokrotnie w różnych cyklach testowych z użyciem różnych danych szczegółowych, należycie dokumentując przy tym zakres przypadku testowego. W idealnej sytuacji dla każdego przypadku testowego istnieje możliwość dwukierunkowego śledzenia powiązań między tym

przypadkiem a pokrywanym przez niego warunkiem testowym (lub warunkami testowymi).

Wśród rezultatów etapu projektowania testów można również wymienić zaprojektowanie i/lub zidentyfikowanie niezbędnych danych testowych, zaprojektowanie środowiska testowego oraz zidentyfikowanie infrastruktury inarzędzi, przy czym stopień udokumentowania powyższych rezultatów bywa bardzo zróżnicowany.

Warunki testowe zdefiniowane w fazie analizy testów mogą być doprecyzowane podczas projektowania testów.

**Produkty pracy implementacji testów**

Produkty pracy związane z implementacją testów to między innymi:

- procedury testowe oraz kolejność ich wykonywania;
- zestawy testowe;
- harmonogram wykonania testów.

W idealnym przypadku, po zakończeniu implementacji testów można wykazać spełnienie kryteriów pokrycia określonych w planie testów dzięki możliwości dwukierunkowego śledzenia powiązań między procedurami testowymi a elementami podstawy testów (za pośrednictwem przypadków testowych i warunków testowych).

W pewnych sytuacjach implementacja testów wiąże się również z utworzeniem produktów pracy, których użycie wymaga korzystania z narzędzi lub narzędzia są przez nie wykorzystywane. Przykłady takich produktów pracy to wirtualizacja usług czy skrypty testów automatycznych.

Implementacja testów może także skutkować utworzeniem i zweryfikowaniem danych testowych iśrodowiska testowego, przy czym poziom kompletności dokumentacji dotyczącej rezultatów weryfikacji danych i/lub środowiska może być bardzo różny.

Dane testowe służą do przypisywania konkretnych wartości do danych wejściowych i oczekiwanych rezultatów przypadków testowych. Te konkretne wartości, wraz z konkretnymi wskazówkami ich użycia, przekształcają przypadki testowe wysokiego poziomu w wykonywalne przypadki testowe niskiego poziomu. Ten sam przypadek testowy wysokiego poziomu można wykonywać z użyciem różnych danych testowych w odniesieniu do różnych wersji przedmiotu testów. Konkretne, oczekiwane rezultaty związane z określonymi danymi testowymi, identyfikuje się przy użyciu wyroczni testowej.

W przypadku testowania eksploracyjnego niektóre produkty pracy związane z projektowaniem i implementacją testów mogą powstawać już w trakcie wykonywania testów, przy czym stopień udokumentowania testów eksploracyjnych imożliwości śledzenia powiązań do określonych elementów podstawy testów może być bardzo różny.

Na etapie implementacji testów można również doprecyzować warunki testowe zdefiniowane podczas analizy testów.

**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;
- dokumentacja wskazująca, które elementy testowe, przedmioty testów, narzędzia testowe i testalia zostały wykorzystane w ramach testowania.

W idealnym przypadku, po zakończeniu wykonywania testów można ustalić status poszczególnych elementów podstawy testów i wygenerować raporty na ten temat dzięki możliwości dwukierunkowego śledzenia powiązań do odpowiednich procedur testowych. Można na przykład wskazać, które wymagania zostały całkowicie przetestowane i pomyślnie przeszły testy, które wymagania nie przeszły testów i/lub wiążą się z defektami oraz które wymagania nie zostały jeszcze w pełni przetestowane. Umożliwia to sprawdzenie, czy zostały spełnione kryteria pokrycia testowego oraz przedstawienie rezultatów testów w raportach w sposób zrozumiały dla interesariuszy.

**Produkty pracy ukończenia testów**

Produkty pracy związane z ukończeniem testów to między innymi: sumaryczne raporty z ukończenia testów, czynności do wykonania mające na celu wprowadzenie udoskonaleń w kolejnych projektach lub iteracjach (np. po retrospektywie projektu zwinnego), żądania zmian lub pozycje listy zaległości produktowych oraz ostateczna wersja testaliów.

### 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 różnic, w celu zapewnienia skutecznego monitorowania i nadzoru, istotne jest stworzenie i utrzymywanie mechanizmu śledzenia powiązań między każdym elementem podstawy testów a odpowiadającymi mu produktami pracy związanymi z testowaniem przez cały czas trwania procesu testowego. Sprawne śledzenie powiązań umożliwia nie tylko ocenę pokrycia testowego, ale również:

- analizowanie wpływu zmian;
- przeprowadzanie audytu testów;
- spełnianie kryteriów związanych z zarządzaniem w obszarze IT;
- tworzenie bardziej zrozumiałych raportów o statusie testów i sumarycznych raportów z testów

dzięki uwzględnieniu statusu elementów podstawy testów (np. wymagań, dla których testy zostały zaliczone, niezaliczone lub czekają na wykonanie);

- przekazywanie interesariuszom informacji o aspektach technicznych testowania w zrozumiałej dla nich formie;

- udzielanie informacji potrzebnych do oceny jakości produktów, możliwości procesów i postępu w realizacji projektu z punktu widzenia możliwości osiągnięcia celów biznesowych.

Niektóre narzędzia do zarządzania testami mają wbudowane modele produktów pracy związanych z testowaniem, które odpowiadają pewnej części lub wszystkim produktom omówionym w tym punkcie. Ponadto istnieją organizacje, które budują własne systemy zarządzania wcelu uporządkowania produktów pracy i dostarczania niezbędnych informacji związanych ze śledzeniem.

## 1.5. Psychologia testowania

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

### 1.5.1. Psychologia człowieka a testowanie

Identyfikowanie defektów podczas testowania statycznego (np. w ramach przeglądów wymagań lub sesji doprecyzowywania historyjek użytkowników) bądź identyfikowanie awarii w trakcie testowania dynamicznego może być odbierane jako krytyka produktu lub jego autora. Jednocześnie zjawisko psychologiczne zwane efektem potwierdzenia (ang. *confirmation bias*) może utrudniać zaakceptowanie informacji sprzecznych zdotychczasowymi przekonaniami. Przykładem takiego zjawiska są oczekiwania programistów, że ich kod będzie poprawny, co prowadzi do wystąpienia efektu potwierdzenia, który skutkuje tym, że trudno jest im zaakceptować fakt, że ich kod jest błędny. Obok efektu potwierdzenia mogą też występować inne błędy poznawcze, które utrudniają ludziom zrozumienie lub zaakceptowanie informacji uzyskanych w wyniku testowania. Dodatkowy czynnik utrudniający przyjęcie informacji zwrotnej z testów stanowi skłonność wielu osób do obwiniania osoby przynoszącej złe wiadomości, a takie sytuacje często są rezultatem testowania.

Na skutek działania powyższych czynników psychologicznych niektóre osoby mogą postrzegać testowanie jako czynność destrukcyjną, nawet jeśli przyczynia się ono wydatnie do postępu w realizacji projektu i podnoszenia jakości produktów. Aby ograniczyć podobne reakcje, należy przekazywać informacje o defektach i awariach w sposób jak najbardziej konstruktywny, co pozwoli zmniejszyć napięcia między testerami a analitykami, właścicielami produktów, projektantami i programistami. Zasada ta ma zastosowanie zarówno do testowania statycznego, jak i do testowania dynamicznego.

Testerzy i kierownicy testów muszą posiadać duże umiejętności interpersonalne, aby sprawnie przekazywać informacje na temat defektów, awarii, rezultatów testów, postępu testowania i ryzyk oraz budować pozytywne relacje ze współpracownikami. Poniżej przedstawiono kilka zaleceń dotyczących zasad dobrej wymiany informacji:

- Należy zacząć od współpracy, a nie od konfliktu. Wszystkich powinna łączyć świadomość wspólnego celu, jakim jest podnoszenie jakości systemów.

- Należy podkreślić korzyści wynikające z testowania. Przykładem takiego zachowania jest wykorzystanie przez autorów informacji o defektach do doskonalenia produktów pracy i rozwijania umiejętności. Dla organizacji wykrycie i usunięcie defektów podczas testowania oznacza oszczędność czasu ipieniędzy oraz zmniejszenie ogólnego ryzyka związanego z jakością produktów.

- Informacje na temat rezultatów testów i inne wnioski należy przekazywać w sposób neutralny, koncentrując się na faktach inie krytykując osoby, która stworzyła wadliwy element. W związku z powyższym należy zadbać o to, by raporty o defektach i wnioski z przeglądu były obiektywne oraz znajdowały oparcie w faktach.

- Należy wczuć się w sytuację drugiej osoby i zrozumieć, dlaczego negatywnie reaguje ona na podane informacje.

- Należy upewnić się, że rozmówca rozumie przekazywane informacje i vice versa.

W poprzedniej części dokumentu omówiono typowe cele testowania. Jednoznaczne zdefiniowanie właściwego zbioru celów testowania ma istotne implikacje psychologiczne, ponieważ większość osób ma skłonność do dopasowywania swoich planów i zachowań do celów określonych przez zespół, kierownictwo i innych interesariuszy. W związku z tym należy zadbać o to, by testerzy przestrzegali przyjętych założeń, a ich osobiste nastawienie w jak najmniejszym stopniu wpływało na wykonywaną pracę.

### 1.5.2. Różnice w sposobie myślenia testerów i programistów

Programiści itesterzy często prezentują odmienny sposób myślenia. Podstawowym celem prac programistycznych jest zaprojektowanie iwykonanie produktu. Jak wspomniano powyżej, cele testowania obejmują weryfikację iwalidację produktu, wykrycie defektów przed przekazaniem produktu do eksploatacji itd. Tym samym mamy do czynienia z różnymi zbiorami celów wymagającymi różnego nastawienia, których pogodzenie jest kluczem do uzyskania produktu wyższej jakości.

Sposób myślenia danej osoby wyznaczają przyjmowane przez nią założenia oraz preferowane przez nią sposoby podejmowania decyzji i rozwiązywania problemów. Wśród pożądanych cech osobowościowych testera należy wymienić: ciekawość, „zawodowy pesymizm", umiejętność krytycznego spojrzenia na wykonywane czynności, dbałość o szczegóły oraz motywację do utrzymywania opartych na zaufaniu i wspólnym dążeniu do celu relacji zawodowych. Warto też zaznaczyć, że sposób myślenia testera często ewoluuje i dojrzewa wraz ze zdobywaniem przez niego doświadczenia zawodowego.

Wśród typowych cech osobowościowych programisty mogą występować niektóre cechy charakterystyczne dla testera, ale programiści odnoszący sukcesy w swoim zawodzie są często bardziej zainteresowani projektowaniem i budowaniem rozwiązań niż analizowaniem ewentualnych problemów z przyjętymi rozwiązaniami. Ważnym czynnikiem jest również efekt potwierdzenia, który może utrudnić

znajdowanie pomyłek w rezultatach własnej pracy.

Programiści mający odpowiednie nastawienie mogą testować własny kod. Różne cykle życia oprogramowania oferują różne sposoby organizowania pracy testerów i wykonywania czynności testowych. Zlecając wykonanie niektórych czynności testowych niezależnym testerom, można zwiększyć skuteczność wykrywania defektów, co jest szczególnie ważne w przypadku systemów dużych, złożonych lub krytycznych ze względów bezpieczeństwa. Niezależni testerzy mają inny punkt widzenia, różny od punktów widzenia autorów produktów pracy (tj. analityków biznesowych, właścicieli produktów, projektantów i programistów), ponieważ mają inne uprzedzenia poznawcze (ang. cognitive biases).