# WYKŁAD 5

## Spis treści

# 5. Zarządzanie testami

## 5.1 Organizacja testów

### 5.1.1. Niezależne testowanie

Zadania związane z testowaniem mogą wykonywać zarówno osoby pełniące konkretne role w procesie testowym, jak i osoby pełniące inne role (np. klienci). Z jednej strony pewien stopień niezależności często zwiększa skuteczność wykrywania defektów, ponieważ działania autora i testera mogą być obarczone różnymi błędami poznawczymi. Z drugiej strony niezależność nie zastępuje znajomości produktu, a deweloperzy znający doskonale produkt mogą efektywnie wykrywać wiele defektów w tworzonym przez siebie kodzie.

Poniżej przedstawiono niektóre poziomy niezależności testowania (w kolejności od najniższego do najwyższego):

- brak niezależnych testerów (jest to jedyna forma testowania dostępna w sytuacji, w której programiści testują własny kod);
- niezależni deweloperzy lub testerzy pracujący w zespole deweloperskim lub projektowym (w tym przypadku deweloperzy mogą np. testować produkty tworzone przez współpracowników);
- niezależny zespół testowy działający w ramach organizacji, podlegający kierownictwu projektu lub dyrekcji;
- niezależni testerzy będący przedstawicielami działów biznesowych lub społeczności użytkowników oraz testerzy specjalizujący się w określonych typach testów, takich jak: testowanie użyteczności, testowanie zabezpieczeń, testowanie wydajnościowe, testowanie zgodności czy testowanie przenaszalności;
- niezależni testerzy spoza organizacji pracujący u klienta (ang. in-house) lub poza siedzibą firmy (ang. outsourcing).

W większości typów projektów zazwyczaj najlepiej sprawdza się uwzględnienie wielu poziomów testowania i powierzenie wykonywania części testów niezależnym testerom. W testowaniu (zwłaszcza na niższych poziomach testów, np. testowaniu modułowym) powinni również uczestniczyć programiści, ponieważ w ten sposób mogą kontrolować jakość swojej pracy.

Sposób zapewnienia niezależności testowania zależy od wybranego modelu cyklu życia oprogramowania. W przypadku zwinnego wytwarzania oprogramowania testerzy mogą zostać przydzieleni do pracy w ramach zespołu deweloperów, a w niektórych organizacjach stosujących metody zwinne mogą być uznawani za członków szerszego niezależnego zespołu testowego.

Ponadto w takich organizacjach właściciele produktu mogą wykonywać na zakończenie każdej iteracji testowanie akceptacyjne mające na celu walidację historyjek użytkownika.

**Wśród potencjalnych korzyści wynikających z niezależności testowania należy wymienić:**

- prawdopodobieństwo wykrycia przez niezależnych testerów innego rodzaju awarii niż te wykryte przez deweloperów ze względu na różne doświadczenia, techniczne punkty widzenia i błędy poznawcze;
- możliwość zweryfikowania, zakwestionowania lub obalenia przez niezależnych testerów założeń przyjętych przez interesariuszy na etapie specyfikowania i implementowania systemu;
- niezależni testerzy po stronie dostawcy mogą w rzetelny i obiektywny sposób informować o testowanym systemie bez (politycznego) nacisku ze strony firmy, która ich zatrudniła.

**Potencjalne wady niezależności testowania to między innymi:**

- izolacja testerów od zespołu deweloperów może prowadzić do braku komunikacji, opóźnień wdostarczaniu informacji zwrotnych zespołowi wytwórczemu i złych relacji z zespołem wytwórczym;
- niebezpieczeństwo utraty przez programistów poczucia odpowiedzialności za jakość;
- niebezpieczeństwo potraktowania niezależnych testerów jako wąskiego gardła;
- ryzyko, że niezależni testerzy nie będą dysponowali ważnymi informacjami (np. na temat przedmiotu testów).

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

### 5.1.2. Zadania kierownika testów i testera

W niniejszym sylabusie omówiono dwie role związane z testowaniem — kierownika testów i testera. Czynności i zadania wykonywane przez osoby pełniące obie role zależą od kontekstu projektu i produktu, umiejętności konkretnych osób oraz specyfiki organizacji.

Kierownik testów ponosi ogólną odpowiedzialność za proces testowy i sprawne kierowanie czynnościami związanymi z testowaniem. Rolę tę może pełnić zarówno osoba zajmująca się zawodowo kierowaniem testami, jak i kierownik projektu, kierownik zespołu deweloperskiego lub kierownik ds. zapewnienia jakości. Ponadto w przypadku większych projektów lub organizacji kierownik lub koordynator testów może mieć pod sobą kilka zespołów testowych, którymi kierują liderzy testów lub testerzy prowadzący. W zakres obowiązków kierownika testów wchodzą zwykle następujące zadania:

- opracowywanie lub dokonywanie przeglądu strategii testów i polityki testów danej organizacji;
- planowanie testów, w tym uwzględnienie kontekstu oraz zapoznanie się z celami testów i czynnikami ryzyka w zakresie testowania (zadanie to może obejmować dokonywanie wyboru podejść do testowania, szacowanie czasu, pracochłonności i kosztów testowania, pozyskiwanie zasobów, definiowanie poziomów testów i cykli testowych oraz planowanie zarządzania defektami);
- sporządzanie i aktualizowanie planu testów;
- koordynowanie realizacji strategii testów i planu testów wraz z kierownikami projektu i innymi interesariuszami;
- prezentowanie punktu widzenia testerów w ramach innych czynności projektowych (takich jak planowanie integracji);
- inicjowanie procesów analizy, projektowania, implementacji i wykonywania testów, monitorowanie rezultatów testów oraz sprawdzanie statusu kryteriów wyjścia (definicji ukończenia) oraz nadzór nad czynnościami związanymi z zakończeniem testów;
- przygotowanie i dostarczenie raportu z postępu testów i raportu sumarycznego z testów na podstawie zgromadzonych informacji;
- dostosowywanie planu testów do rezultatów i postępu testów (dokumentowanych niekiedy w raportach o postępie testów i sumarycznych raportach z ukończenia testów) oraz podejmowanie niezbędnych działań w zakresie nadzoru nad testami;
- udzielanie pomocy w tworzeniu efektywnego mechanizmu zarządzania konfiguracją testaliów i systemu zarządzania defektami;
- wprowadzanie odpowiednich miar służących do mierzenia postępu testów oraz oceniania jakości testowania i produktu;
- udzielanie pomocy przy wyborze i implementacji narzędzi służących do realizacji procesu testowego, w tym przy określaniu budżetu przeznaczonego na wybór narzędzi (i potencjalnie na ich zakup lub pomoc techniczną) oraz wyznaczanie czasu na realizację projektów pilotażowych, a także udzielanie dalszej pomocy w zakresie korzystania z narzędzi;
- podejmowanie decyzji o implementacji środowisk testowych;
- monitorowanie procesu testowania oraz sporządzanie i przedstawianie raportów z testów na podstawie zgromadzonych informacji;
- promowanie testerów izespołu testowego oraz reprezentowanie ich punktu widzenia w organizacji;
- stwarzanie możliwości rozwijania umiejętności i kariery testerów (np. poprzez plan szkoleń, ewaluacje osiągnięć, coaching).

Sposób zwinnego wytwarzania oprogramowania niektóre z wyżej wymienionych zadań, a zwłaszcza zadania związane z codziennym testowaniem w obrębie zespołu zwinnego, wykonuje tester należący do tego zespołu. Niektóre zadania, które obejmują swoim zasięgiem kilka zespołów

lub całą organizację bądź są związane z zarządzaniem kadrami, mogą wykonywać kierownicy testów spoza zespołu tworzącego oprogramowaie (zwani niekiedy trenerami testowymi).

Typowe zadania testera to między innymi:

- dokonywanie przeglądu planów testów i uczestniczenie w ich opracowywaniu;
- analizowanie, dokonywanie przeglądu i ocenianie wymagań, historyjek użytkownika i kryteriów akceptacji, specyfikacji oraz modeli (tj. podstawy testów) pod kątem ich testowalności;
- identyfikowanie i dokumentowanie warunków testowych oraz rejestrowanie powiązań między przypadkami testowymi, warunkami testowymi a podstawą testów;
- projektowanie, konfigurowanie i weryfikowanie środowisk testowych (często w porozumieniu z administratorami systemu i sieci);
- projektowanie i implementowanie przypadków testowych i skryptów testowych;
- przygotowywanie i pozyskiwanie danych testowych;
- tworzenie szczegółowego harmonogramu wykonywania testów;
- wykonywanie testów, ocenianie rezultatów i dokumentowanie odchyleń od oczekiwanych rezultatów;
- korzystanie z odpowiednich narzędzi usprawniających proces testowy;
- automatyzowanie testowania w zależności od potrzeb (zadanie to może być wykonywane z pomocą programisty lub eksperta ds. automatyzacji testowania);
- ewaluacja charakterystyk niefunkcjonalnych, takich jak: wydajność, niezawodność, użyteczność, zabezpieczenia, zgodność, przenaszalność;
- dokonywanie przeglądu testów opracowanych przez inne osoby.

Osoby zajmujące się analizą testów, projektowaniem testów, tworzeniem określonych typów testów czy automatyzacją testowania mogą być specjalistami w tych dziedzinach. Ponadto, w zależności od czynników ryzyka związanych z produktem i projektem oraz od wybranego modelu cyklu życia oprogramowania, na różnych poziomach testowania rolę testerów mogą przyjmować różne osoby. Na poziomie testowania modułowego i integracji modułów testerami są często programiści, na poziomie testowania akceptacyjnego — analitycy biznesowi, eksperci merytoryczni i użytkownicy, na poziomie testowania systemowego i integracji systemów — członkowie niezależnego zespołu testowego, a na poziomie produkcyjnych testów akceptacyjnych — operatorzy i/lub administratorzy systemu.

## 5.2 Planowanie i szacowanie testów

### 5.2.1 Cel i treść planu testów

Plan testów przedstawia w sposób skrótowy czynności testowe wykonywane w ramach projektów wytwarzania i pielęgnacji oprogramowania. Na proces planowania wpływają: polityka i strategia

testów obowiązujące w danej organizacji, stosowane cykle życia i metody wytwarzania oprogramowania (patrz podrozdział 2.1.), zakres testowania, cele, czynniki ryzyka, ograniczenia, krytyczność, testowalność oraz dostępność zasobów. Ponadto, w miarę realizacji projektu i planu testów, udostępniane są dodatkowe informacje, które pozwalają uwzględnić w planie testów więcej szczegółów.

Planowanie testów to czynność o charakterze ciągłym, która jest wykonywana przez cały cykl życia oprogramowania (przy czym cykl ten może wykraczać poza zakres projektu i obejmować również fazę pielęgnacji). Ważnym jej elementem jest wykorzystanie informacji zwrotnych z czynności testowych do rozpoznawania zmieniających się czynników ryzyka i odpowiedniego korygowania planów. Rezultaty procesu planowania mogą zostać udokumentowane w głównym planie testów oraz w oddzielnych planach dotyczących poszczególnych poziomów testów (takich jak testowanie systemowe i akceptacyjne) bądź typów testów (takich jak testowanie użyteczności czy testowanie wydajnościowe). Planowanie testów może obejmować następujące działania (przy czym niektóre z nich mogą zostać udokumentowane w planie testów):

- określanie zakresu i celów testowania oraz związanego z nim ryzyka;
- określanie ogólnego podejścia do testowania;
- integrowanie i koordynowanie czynności testowych w ramach czynności związanych z cyklem życia oprogramowania;
- podejmowanie decyzji dotyczących tego, co należy przetestować, jakie kadrowe i inne zasoby będą niezbędne do wykonania poszczególnych czynności testowych oraz w jaki sposób należy wykonać te czynności; planowanie czynności związanych z analizą, projektowaniem, implementacją, wykonywaniem i oceną testów poprzez określenie konkretnych terminów (np. w ramach sekwencyjnego wytwarzania oprogramowania) lub umieszczanie tych czynności w kontekście poszczególnych iteracji (np. w przypadku iteracyjnego wytwarzania oprogramowania);
- dokonywanie wyboru miar służących do monitorowania i nadzorowania testów;
- określanie budżetu potrzebnego na czynności testowe;
- ustalanie poziomu szczegółowości i struktury dokumentacji testów (np. poprzez udostępnianie szablonów lub przykładowych dokumentów).

Zawartość planu testów może się zmieniać, może też być szersza od zakresu opisanego powyżej. Przykładowy wzorzec planu testów i przykładowy plan testów zostały opisane w standardzie ISO [ISO/IEC/IEEE 29119-3].

### 5.2.2. Strategia testów i podejście do testowania

Strategia testów jest dokumentem wysokopoziomowym i zapewnia ogólny opis realizowanego procesu testowego, zazwyczaj na poziomie produktu bądź organizacji. Do typowych strategii testów zalicza się:

- Strategia analityczna. Strategia ta opiera się na analizie określonego czynnika (np. wymagań lub ryzyka). Przykładem podejścia analitycznego jest testowanie oparte na ryzyku, w którym punktem wyjścia do projektowania testów i ustalania ich priorytetów jest poziom ryzyka.
- Strategia oparta na modelu. W przypadku tej strategii testy projektuje się na podstawie modelu konkretnego wymaganego aspektu produktu, np.: funkcji, procesu biznesowego, struktury wewnętrznej bądź charakterystyki niefunkcjonalnej (takiej jak niezawodność). Przykłady takich modeli to: modele procesów biznesowych, modele stanów czy modele wzrostu niezawodności.
- Strategia metodyczna. Podstawą tej strategii jest systematyczne stosowanie zgóry określonego zbioru testów lub warunków testowych, takich jak listy kontrolne zawierające typowe lub prawdopodobne typy awarii, listy ważnych charakterystyk jakościowych czy obowiązujące w firmie standardy wyglądu i działania (ang. look-and-feel) aplikacji mobilnych i stron internetowych.
- Strategia zgodna z procesem (lub standardem). Strategia ta polega na analizie, projektowaniu i implementacji przypadków testowych na podstawie zewnętrznych reguł istandardów (wynikających na przykład z norm branżowych), dokumentacji procesów bądź rygorystycznego identyfikowania i wykorzystania podstawy testów, a także na podstawie wszelkich procesów lub standardów narzuconych organizacji lub przez organizację.
- Strategia kierowana (lub konsultatywna). Podstawą tej strategii są przede wszystkim porady, wskazówki i instrukcje interesariuszy, ekspertów merytorycznych lub ekspertów technicznych — również spoza zespołu testowego lub organizacji.
- Strategia testowa minimalizująca regresję. Ta strategia – motywowana chęcią uniknięcia regresji już istniejących funkcjonalności – przewiduje ponowne wykorzystanie dotychczasowych testaliów (zwłaszcza przypadków testowych), szeroką automatyzację testów regresji oraz stosowanie standardowych zestawów testowych.
- Strategia reaktywna. W przypadku tej strategii testowanie jest ukierunkowane bardziej na reagowanie na zdarzenia niż na realizację ustalonego z góry planu (jak w przypadku wyżej opisanych strategii), a testy są projektowane i mogą być natychmiast wykonywane w oparciu o wiedzę uzyskaną dzięki rezultatom wcześniejszych testów.

Właściwa strategia testowa zazwyczaj jest kombinacją powyższych strategii. Przykładem takiej sytuacji jest testowanie oparte na ryzyku (strategia analityczna), które może być połączone z testowaniem eksploracyjnym (strategia reaktywna), ponieważ te strategie uzupełniają się nawzajem i wpływają na efektywność testowania, gdy są stosowane jednocześnie.

O ile strategia testów oferuje ogólny opis procesu testowego, o tyle podejście do testowania dostosowuje ją do potrzeb konkretnego projektu lub konkretnej wersji oprogramowania/systemu. Podejście do testowania jest punktem wyjścia do dokonania wyboru technik testowania, poziomów testów i typów testów oraz zdefiniowania kryteriów wejścia i wyjścia (lub definicji gotowości i definicji ukończenia). Dostosowując podejście do konkretnej sytuacji, należy podjąć określone decyzje wynikające ze złożoności icelów projektu, typu wytwarzanego produktu oraz analizy ryzyka produktowego. Wybrane podejście zależy od kontekstu i może uwzględniać takie czynniki jak: ryzyko, bezpieczeństwo, dostępność zasobów i umiejętności, technologie, charakter systemu (np. system wytworzony na zamówienie albo oprogramowanie do powszechnej sprzedaży), cele testów oraz obowiązujące przepisy.

### 5.2.3. Kryteria wejścia i wyjścia (definicja gotowości i definicja ukończenia)

W celu zapewnienia skutecznego nadzoru nad jakością oprogramowania i testowania zaleca się określenie kryteriów wskazujących, kiedy należy rozpocząć daną czynność testową oraz kiedy można ją uznać za zakończoną. Kryteria wejścia (w przypadku zwinnego wytwarzania oprogramowania zwane zwykle definicją gotowości) określają warunki wstępne, które muszą zostać spełnione przed rozpoczęciem danej czynności testowej. W przypadku niespełnienia kryteriów wejścia wykonanie tej czynności może być trudniejsze oraz bardziej czasochłonne, kosztowne i ryzykowne. Kryteria wyjścia (w przypadku zwinnego wytwarzania oprogramowania zwane zwykle definicją ukończenia) określają warunki, które muszą zostać spełnione, aby można było uznać wykonywanie testów danego poziomu lub zbioru testów za zakończone. Kryteria wejścia i wyjścia należy zdefiniować w odniesieniu do każdego poziomu i typu testów (kryteria te różnią się w zależności od celów testów).

Typowe kryteria wejścia to między innymi:

- dostępność testowalnych wymagań, historyjek użytkownika i/lub modeli (np. w przypadku stosowania strategii testowej opartej na modelu);
- dostępność elementów testowych, które spełniły kryteria wyjścia obowiązujące na wcześniejszych poziomach testów;
- dostępność środowiska testowego;
- dostępność niezbędnych narzędzi testowych;
- dostępność danych testowych i innych niezbędnych zasobów.

Typowe kryteria wyjścia to między innymi:

- zakończenie wykonywania zaplanowanych testów;
- osiągnięcie odpowiedniego poziomu pokrycia (tj. pokrycia wymagań, historyjek użytkownika i kryteriów akceptacji oraz kodu);
- nieprzekroczenie uzgodnionego limitu nieusuniętych defektów;

- uzyskanie wystarczająco niskiej, szacowanej gęstości defektów,
- uzyskanie wystarczająco wysokich wskaźników niezawodności, wydajności, użyteczności,
- zabezpieczeń i innych istotnych charakterystyk.

Nawet w przypadku niespełnienia kryteriów wyjścia, czynności testowe mogą zostać skrócone z powodu wykorzystania całego budżetu, upływu zaplanowanego czasu i/lub presji związanej z wprowadzeniem produktu na rynek. Zakończenie testowania w takich okolicznościach może być dopuszczalne, o ile interesariusze i właściciele biznesowi projektu znają i akceptują ryzyko związane z uruchomieniem systemu bez dalszego testowania.

### 5.2.4. Harmonogram wykonywania testów

Po opracowaniu poszczególnych przypadków i procedur testowych (oraz ewentualnym zautomatyzowaniu niektórych z nich) oraz zgrupowaniu tych że w zestawy testowe, można utworzyć na ich podstawie harmonogram wykonywania testów, który określa kolejność ich uruchamiania. Harmonogram ten powinien uwzględniać takie czynniki jak: priorytety, zależności, konieczność wykonania testów potwierdzających i regresji oraz najbardziej efektywną kolejność wykonywania testów.

O ile jest to możliwe, testy powinny być wykonywane w kolejności odpowiadającej poziomom priorytetów (zwykle najpierw wykonuje się testy o najwyższym priorytecie). Praktyka ta może jednak nie mieć zastosowania, jeśli przypadki testowe lub testowane przez nie cechy są uzależnione od innych elementów. Przykładem może być sytuacja, gdy przypadek testowy owyższym priorytecie jest uzależniony od przypadku o niższym priorytecie — należy wtedy w pierwszej kolejności wykonać przypadek o niższym priorytecie. Analogicznie: jeśli występują wzajemne zależności między kilkoma przypadkami testowymi, kolejność ich wykonywania należy ustalić niezależnie od priorytetów. Ponadto może zajść konieczność priorytetowego wykonania testów potwierdzających i regresji w zależności od tego, jak ważne jest szybkie uzyskanie informacji zwrotnych na temat zmian, przy czym w takim przypadku może być również konieczne uwzględnienie ewentualnych współzależności.

W niektórych przypadkach możliwe są różne sekwencje testów różniące się efektywnością. W takiej sytuacji należy odpowiednio wyważyć kwestie efektywności wykonywania testów oraz przestrzegania ustalonych priorytetów.

### 5.2.5. Czynniki wpływające na pracochłonność testowania

Jednym z elementów zarządzania testami jest szacowanie ich pracochłonności, czyli przewidywanie nakładów pracy związanych z testowaniem, które są niezbędne do osiągnięcia celów testowania wprzypadku danego projektu bądź danej wersji lub iteracji. Do czynników

wpływających na pracochłonność testowania należą między innymi: charakterystyka produktu, charakterystyka procesu wytwarzania oprogramowania, czynniki ludzkie oraz rezultaty testów.

**Charakterystyka produktu**

- czynniki ryzyka związane z produktem;
- jakość specyfikacji (tj. podstawy testów);
- wielkość produktu;
- złożoność dziedziny produktu;
- wymagania dotyczące charakterystyk jakościowych (np. bezpieczeństwa i niezawodności);
- wymagany poziom szczegółowości dokumentacji testów;
- wymagania dotyczące zgodności z przepisami prawa.

**Charakterystyka procesu wytwarzania oprogramowania**

- stabilność i dojrzałość organizacji;
- stosowany model wytwarzania oprogramowania (np. model kaskadowy, model zwinny);
- podejście do testowania;
- używane narzędzia;
- proces testowy;
- presja czasu.

**Czynniki ludzkie**

- umiejętności i doświadczenie zaangażowanych osób, zwłaszcza doświadczenie z podobnymi projektami/produktami (np. doświadczenie branżowe);
- umiejętność współpracy zespołu i umiejętności kierownika.

**Rezultaty testów**

- liczba i ważność wykrytych defektów;
- liczba wymaganych poprawek.

### 5.2.6. Techniki szacowania testów

Do oszacowania nakładów pracy niezbędnych do prawidłowego wykonania testów w ramach projektu można użyć różnych technik. Dwie najczęściej stosowane techniki to:

- technika oparta na miarach — szacowanie pracochłonności testowania na podstawie miar z wcześniejszych podobnych projektów lub na podstawie wartości typowych;

- technika ekspercka — szacowanie pracochłonności testowania na podstawie doświadczenia osób odpowiedzialnych za zadania związane z testowaniem lub przez ekspertów.

Przykładem zastosowania podejścia opartego na miarach w kontekście zwinnego wytwarzania oprogramowania są wykresy spalania (ang. *burndown charts*). Rejestrowane i raportowane przy ich użyciu nakłady pracy są następnie wykorzystywane do określania prędkości zespołu (ang. *team velocity*), czyli ilości pracy, jaką zespół ten może wykonać wnastępnej iteracji. Z kolei poker planistyczny to przykład podejścia eksperckiego, ponieważ członkowie zespołu szacują nakłady pracy niezbędne do dostarczenia danej funkcjonalności na podstawie własnego doświadczenia.

Przykładem zastosowania podejścia opartego na miarach w projektach sekwencyjnych są modele usuwania defektów, które przewidują rejestrowanie i raportowanie liczby defektów oraz czasu niezbędnego do ich usunięcia, a następnie szacowanie na tej podstawie pracochłonności przyszłych projektów o podobnym charakterze. Z kolei szerokopasmowa technika delficka jest przykładem podejścia eksperckiego, zgodnie z którym grupy ekspertów przedstawiają wartości szacunkowe na podstawie dotychczasowego doświadczenia

## 5.3 Białoskrzynkowe techniki testowania

Celem monitorowania testów jest gromadzenie i udostępnianie informacji pozwalających uzyskać wgląd w przebieg czynności testowych. Informacje objęte monitorowaniem, które mogą być zbierane ręcznie lub automatycznie, powinny być wykorzystywane do oceny postępu testów oraz pomiaru spełnienia kryteriów wyjścia dotyczących poszczególnych poziomów testów lub wykonania zadań testowych związanych z definicją ukończenia w projektach zwinnych (takich jak osiągnięcie zakładanego pokrycia czynników ryzyka produktowego, wymagań lub kryteriów akceptacji).

Nadzór nad testami obejmuje wszelkie działania zarządcze i korygujące podejmowane na podstawie zgromadzonych, wykazanych w raportach informacji i miar. Powyższe działania mogą dotyczyć dowolnych czynności testowych i wpływać na pozostałe czynności związane z cyklem życia oprogramowania.

Działania wykonywane w ramach nadzoru nad testami to między innymi:

- ponowne ustalanie priorytetów testów w przypadku zmaterializowania się zidentyfikowanego czynnika ryzyka (np. nieterminowego dostarczenia oprogramowania);
- wprowadzanie zmian w harmonogramie testów zależnie od dostępności lub niedostępności środowiska testowego lub innych zasobów;
- dokonywanie ponownej oceny kryteriów wejścia lub wyjścia w związku z wprowadzeniem poprawek.

### 5.3.1. Miary stosowane w testowaniu

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

- postęp realizacji harmonogramu i budżetu;
- bieżącą jakość przedmiotu testów;
- adekwatność wybranego podejścia do testowania;
- skuteczność czynności testowych z punktu widzenia realizacji celów.

Powszechnie stosowane są następujące miary dotyczące testów:

- procent wykonania zaplanowanych prac związanych z przygotowaniem przypadków testowych (lub procent przygotowania zaplanowanych przypadków testowych);
- procent wykonania zaplanowanych prac związanych z przygotowaniem środowiska testowego;
- poziom wykonania przypadków testowych (np. liczba wykonanych/niewykonanych przypadków testowych, zaliczonych/niezaliczonych przypadków testowych i/lub zaliczonych/niezaliczonych warunków testowych);
- informacje o defektach (np. gęstość defektów, liczba wykrytych i usuniętych defektów, współczynnik awarii oraz rezultaty testów potwierdzających);
- pokrycie testowe wymagań, historyjek użytkownika i kryteriów akceptacji, czynników ryzyka lub kodu;
- koszty testowania, w tym porównanie tych kosztów z korzyściami wynikającymi z wykrycia następnego defektu lub wykonania następnego testu.

### 5.3.2. Cel, treść i odbiorcy raportów z testów

Raporty z testów służą do podsumowywania i przekazywania informacji na temat czynności testowych (np. testów danego poziomu) zarówno w trakcie ich wykonywania, jak i po ich zakończeniu. Raport z testów sporządzany podczas wykonywania czynności testowej może nosić nazwę raportu o postępie testów, a raport sporządzany w momencie zakończenia takiej czynności — sumarycznego raportu z testów.

Na etapie monitorowania inadzoru nad testami kierownik testów regularnie sporządza raporty opostępie testów dla interesariuszy. Oprócz części wspólnych, dla raportu z postępu testów, sumarycznego raportu z testów, typowy raport z postępu testów może zawierać informacje dotyczące:

- statusu czynności testowych i postępu realizacji planu testów;
- czynników zakłócających wykonywanie prac;

- testów zaplanowanych w następnym okresie raportowania;
- jakości przedmiotu testów.

Po spełnieniu kryteriów wyjścia kierownik testów sporządza sumaryczny raport z testów. Dokument ten zawiera podsumowanie wykonanych testów zgodnie z najnowszym raportem z postępu testów oraz innych istotnych informacji.

**Typowy sumaryczny raport z testów zawiera między innymi:**

- podsumowanie wykonanych testów;
- informacje na temat zdarzeń, które miały miejsce w okresie testowania;
- informacje o odstępstwach od planu, włączając w to odstępstwa od harmonogramu, czasu trwania lub wysiłku związanego z testowaniem;
- informacje o statusie testowania i jakości produktu z punktu widzenia kryteriów wyjścia (lub definicji ukończenia);
- informacje o czynnikach, które blokowały lub nadal blokują przebieg testów;
- miary związane z defektami, przypadkami testowymi, pokryciem testowym, postępem prac oraz wykorzystaniem zasobów;
- informacje na temat ryzyka (resztkowego);
- informacje o wytworzonych produktach pracy związanych z testowaniem, które mogą zostać ponownie wykorzystane.

Treść raportu z testów zależy od projektu, wymagań organizacyjnych i cyklu życia oprogramowania. I tak przykładem może być złożony projekt z wieloma interesariuszami lub projekt podlegający określonym regulacjom prawnym, co może wymagać bardziej szczegółowego i rygorystycznego raportowania niż szybka aktualizacja oprogramowania. Ponadto w przypadku zwinnego wytwarzania oprogramowania raportowanie o postępie testów można zintegrować z tablicami zadań, przeglądami defektów i wykresami spalania, które są omawiane podczas codziennych spotkań roboczych (ang. daily stand-up meetings).

Raporty z testów muszą być dostosowane zarówno do kontekstu projektu, jak i do potrzeb docelowych odbiorców. Typ i ilość informacji umieszczonych w raporcie przeznaczonym dla odbiorców technicznych lub zespołu testowego mogą być inne od tych zamieszczonych w raporcie sumarycznym dla kierownictwa. W pierwszym przypadku ważne mogą być szczegółowe informacje na temat typów defektów i związanych z nimi trendów, w drugim zaś bardziej odpowiedni może być raport wysokiego poziomu (zawierający np. podsumowanie statusu defektów według priorytetów, budżetu i harmonogramu oraz zaliczonych, niezaliczonych i niewykonanych przypadków testowych).

Standard ISO [ISO/IEC/IEEE 29119-3] odwołuje się do dwóch typów raportów, tj. raportu o postępie testów i raportu z ukończenia testów (nazywanego w tym sylabusie sumarycznym raportem z testów), a także zawiera struktury i przykłady dotyczące każdego z tych typów.

## 5.4 Zarządzanie konfiguracją

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

W kontekście testowania zarządzanie konfiguracją może polegać na zagwarantowaniu tego, aby:

- wszystkie przedmioty testów zostały zidentyfikowane, objęte kontrolą wersji i śledzeniem zmian oraz powiązane ze sobą wzajemnie;
- wszystkie testalia zostały zidentyfikowane, objęte kontrolą wersji i śledzeniem zmian oraz powiązane ze sobą wzajemnie i z wersjami przedmiotu testów w sposób pozwalający utrzymać możliwość śledzenia na wszystkich etapach procesu testowego;
- wszystkie zidentyfikowane dokumenty i elementy oprogramowania były przywoływane w sposób jednoznaczny w dokumentacji testów.

Procedury zarządzania konfiguracją wraz zniezbędną infrastrukturą (narzędziami) należy zidentyfikować i zaimplementować na etapie planowania testów.

## 5.5 Czynniki ryzyka a testowanie

### 5.5.1. Definicja ryzyka

Ryzyko jest możliwością wystąpienia w przyszłości zdarzenia o niepożądanych konsekwencjach. O poziomie ryzyka decyduje prawdopodobieństwo wystąpienia niekorzystnego zdarzenia oraz jego wpływ (tj. wynikające z niego szkody).

### 5.5.2. Czynniki ryzyka produktowego i projektowego

Ryzyko produktowe występuje wszędzie tam, gdzie produkt pracy (np. specyfikacja, moduł, system lub test) może nie zaspokoić uzasadnionych potrzeb użytkowników i/lub interesariuszy. Czynniki ryzyka produktowego związane z konkretnymi charakterystykami jakościowymi produktu (np. przydatność funkcjonalna, niezawodność, wydajność, użyteczność, utrzymywalność, przenaszalność) są również nazywane czynnikami ryzyka jakościowego.

Przykładami czynników **ryzyka produktowego** mogą być następujące problemy:

- niewykonywanie przez oprogramowanie zakładanych funkcji zgodnie ze specyfikacją;

- niewykonywanie zakładanych funkcji oprogramowania zgodnie z potrzebami użytkowników, klientów i/lub interesariuszy;
- niedostateczne spełnienie przez architekturę systemu określonych wymagań niefunkcjonalnych;
- niepoprawne wykonywanie konkretnych obliczeń w niektórych okolicznościach;
- błędy w kodzie struktury sterowania pętlą;
- zbyt długi czas odpowiedzi w systemie transakcyjnym wysokiej wydajności;
- niezgodność informacji zwrotnych na temat doświadczenia użytkownika z oczekiwaniami dotyczącymi produktu.

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

• problemy związane z projektem, takie jak:
- potencjalne opóźnienia w dostawach, ukończeniu zadań bądź spełnieniu kryteriów wyjścia (definicji ukończenia);
- niedokładne oszacowanie, realokacja środków do projektów o wyższym priorytecie lub ogólne cięcia kosztów w całej organizacji, które mogą skutkować niewystarczającym finansowaniem projektu;
- wprowadzenie w ostatniej chwili zmian wymagających dokonania licznych przeróbek;
• problemy organizacyjne, takie jak:
- niewystarczające kwalifikacje lub przeszkolenie pracowników bądź braki kadrowe;
- konflikty i problemy wynikające z doboru personelu;
- brak dostępności użytkowników, pracowników struktur biznesowych lub ekspertów merytorycznych z powodu sprzecznych priorytetów biznesowych;
• problemy natury politycznej, takie jak:
- niewłaściwe przekazywanie przez testerów informacji na temat ich potrzeb i/lub rezultatów testów;
- niepodjęcie przez programistów/testerów dalszych działań na podstawie informacji uzyskanych wwyniku testowania i przeglądów (np. niewprowadzenie udoskonaleń w procedurach wytwarzania i testowania oprogramowania);
- niewłaściwe podejście do testowania lub oczekiwania związane z testowaniem (np. niedocenianie korzyści wynikających z wykrycia defektów podczas testowania);
• problemy techniczne, takie jak:
- niedostateczne doprecyzowanie wymagań;
- brak możliwości spełnienia wymagań z uwagi na ograniczenia czasowe;
- nieudostępnienie na czas środowiska testowego;
- zbyt późne przeprowadzenie konwersji danych, zaplanowanie migracji lub udostępnienie potrzebnych do tego narzędzi;
- wady w procesie wytwarzania oprogramowania mogące wpływać na spójność lub jakość produktów prac projektowych, kodu, konfiguracji, danych testowych i przypadków testowych;

- kumulacja defektów lub innego rodzaju dług techniczny powstały na skutek problemów z zarządzaniem defektami lub innych podobnych problemów;
• problemy związane z dostawcami, takie jak:
- niedostarczenie niezbędnego produktu lub usługi przez zewnętrznego dostawcę bądź ogłoszenie przez niego upadłości;
- utrudnienia w realizacji projektu wynikające z problemów związanych z umowami.

Ryzyko projektowe może dotyczyć zarówno czynności związanych z wytwarzaniem oprogramowania, jak i jego testowaniem. W niektórych przypadkach za zarządzanie ryzykiem projektowym odpowiadają kierownicy projektów, ale zdarza się również, że za czynniki ryzyka projektowego związane z testowaniem odpowiedzialność ponoszą kierownicy testów.

### 5.5.3. Testowanie oparte na ryzyku a jakość produktu

Wiedzę na temat ryzyka można wykorzystać do odpowiedniego ukierunkowania działań wykonywanych podczas testowania. Na podstawie ryzyka można podejmować decyzje co do tego, gdzie i kiedy należy rozpocząć testowanie, a także identyfikować obszary wymagające większej uwagi. Celem testowania jest zmniejszenie prawdopodobieństwa wystąpienia niekorzystnego zdarzenia bądź ograniczenie jego wpływu. W związku z tym testowanie przyczynia się do łagodzenia ryzyka, a także dostarcza informacje na temat zidentyfikowanych czynników ryzyka — w tym ryzyka resztkowego (rezydualnego, nie rozwiązanego).

Podejście do testowania oparte na ryzyku umożliwia prewencyjne obniżanie poziomu ryzyka produktowego. Jednym z elementów tego podejścia jest analiza ryzyka produktowego, która obejmuje identyfikowanie czynników tego typu ryzyka oraz szacowanie prawdopodobieństwa wystąpienia i wpływu każdego z nich. Uzyskane w ten sposób informacje na temat ryzyka produktowego można wykorzystać do planowania testów, specyfikowania, przygotowywania i wykonywania przypadków testowych oraz monitorowania i nadzorowania testów. Wczesna analiza ryzyka produktowego przyczynia się do powodzenia całego projektu.

W przypadku podejścia opartego na ryzyku wynik przeprowadzonej analizy ryzyka produktowego umożliwia:

- wskazanie odpowiednich technik testowania;
- określenie konkretnych typów testów, które należy wykonać (np. testy zabezpieczeń, testy dostępności);
- określenie zakresu wykonywanych testów;

- ustalenie priorytetów testowania w sposób sprzyjający jak najwcześniejszemu wykryciu defektów krytycznych;

- ustalenie, czy w celu zmniejszenia ryzyka należy wykonać inne czynności niezwiązane z testowaniem (np. przeprowadzić szkolenie dla niedoświadczonych projektantów).

Testowanie oparte na ryzyku pozwala skorzystać ze wspólnej wiedzy i spostrzeżeń interesariuszy projektu do przeprowadzenia analizy ryzyka produktowego. Aby zminimalizować prawdopodobieństwo awarii produktu, w ramach zarządzania ryzykiem wykonuje się usystematyzowane czynności obejmujące:

- analizowanie potencjalnych problemów (czynników ryzyka) i regularne dokonywanie ich ponownej oceny;
- ustalanie, które czynniki ryzyka są istotne i wymagają podjęcia działań;
- podejmowanie działań mających na celu złagodzenie ryzyka;
- tworzenie planów awaryjnych na wypadek faktycznego wystąpienia określonych czynników ryzyka.

Ponadto testowanie pozwala zidentyfikować nowe czynniki ryzyka, wskazać czynniki wymagające złagodzenia oraz zmniejszyć niepewność związaną z ryzykiem.

## 5.6 Zarządzanie defektami

Jednym z celów testowania jest znajdowanie defektów, w związku z czym wszystkie znalezione defekty należy zalogować. Sposób zgłoszenia defektu zależy od kontekstu testowania danego modułu lub systemu, poziomu testów oraz wybranego modelu wytwarzania oprogramowania. Każdy zidentyfikowany defekt powinien zostać zbadany i objęty śledzeniem od momentu wykrycia i sklasyfikowania, do momentu rozwiązania problemu (tj. usunięcia defektu i pomyślnego zakończenia testów potwierdzających, odroczenia do kolejnej wersji, zaakceptowania defektu jako trwałego ograniczenia produktu itd.). Aby umożliwić zarządzanie wszystkimi defektami do czasu ich usunięcia, organizacja powinna wdrożyć proces zarządzania defektami (obejmujący również odpowiedni przepływ pracy i reguły klasyfikacji defektów). Proces ten musi zostać uzgodniony ze wszystkimi podmiotami uczestniczącymi w zarządzaniu defektami, w tym z architektami, projektantami, deweloperami, testerami oraz właścicielami produktu. W niektórych organizacjach proces zgłaszania i śledzenia defektów może być nieformalny.

W procesie zarządzania defektami może się okazać, że w niektórych raportach opisane zostały rezultaty fałszywie pozytywne, a nie rzeczywiste awarie spowodowane defektami. Przykładem takiej sytuacji może być niezaliczenie testu na skutek przerwania połączenia sieciowego lub przekroczenia limitu czasu. Zachowanie to nie jest skutkiem defektu w przedmiocie testów, ale należy je potraktować jako nieprawidłowość i zbadać. W związku z tym testerzy powinni starać się ograniczać do minimum liczbę rezultatów fałszywie pozytywnych, które są zgłaszane jako defekty.

Defekty można zgłaszać na etapie kodowania, analizy statycznej, przeglądów, podczas testowania dynamicznego lub podczas eksploatacji produktu. Raporty o defektach mogą dotyczyć problemów występujących w kodzie lub działającym systemie bądź we wszelkiego rodzaju dokumentacji, w tym wwymaganiach, historyjkach użytkownika i kryteriach akceptacji, dokumentach związanych zprogramowaniem lub testowaniem, podręcznikach użytkownika bądź podręcznikach insta

Aby stworzyć skuteczny i efektywny proces zarządzania defektami, organizacje mogą określać własne standardy dotyczące atrybutów i klasyfikacji defektów oraz związanego z ich obsługą przepływu pracy.

**Typowy raport o defekcie ma na celu:**

- dostarczenie deweloperom i innym interesariuszom informacji na temat wszelkich zaistniałych zdarzeń wzakresie niezbędnym do zidentyfikowania skutku, wyizolowania problemu i umożliwienie skorygowania ewentualnych defektów;
- umożliwienie kierownikom testów śledzenia jakości produktu pracy i wpływu na postęp testowania (np. znalezienie dużej liczby defektów będzie wymagało czasu na ich zgłoszenie, a następnie przeprowadzenie testów potwierdzających);
- przedstawienie sugestii dotyczących usprawnienia procesów wytwarzania i testowego.

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

- identyfikator;
- tytuł i krótkie podsumowanie zgłaszanego defektu;
- data raportu o defekcie, zgłaszająca jednostka organizacyjna i autor zgłoszenia;
- identyfikacja elementu testowego (testowanego elementu konfiguracji) i środowiska;
- faza cyklu życia oprogramowania, w której zaobserwowano defekt;
- opis defektu umożliwiający jego odtworzenie i usunięcie (w tym ewentualne dzienniki, zrzuty bazy danych, zrzuty ekranu lub nagrania), gdy defekt został wykryty podczas wykonywania testów;
- oczekiwane i rzeczywiste rezultaty;
- zakres lub stopień wpływu (ważność) defektu z punktu widzenia interesariuszy;
- priorytet usunięcia defektu;
- status raportu o defekcie (np. otwarty, odroczony, powielony, oczekujący na poprawkę, oczekujący na testowanie potwierdzające, ponownie otwarty, zamknięty);
- wnioski, zalecenia i zgody;
- kwestie globalne takie jak wpływ zmiany wynikającej zdefektu na inne obszary oprogramowania/systemu;

- historia zmian, w tym sekwencja działań podjętych przez członków zespołu projektowego w celu zlokalizowania, usunięcia i potwierdzenia usunięcia defektu;
- odwołania do innych elementów, w tym do przypadku testowego, dzięki któremu problem został ujawniony.

Niektóre z powyższych szczegółowych informacji mogą zostać automatycznie uwzględnione i/lub objęte zarządzaniem w przypadku korzystania z narzędzi do zarządzania defektami (przykładem może być automatyczne przypisanie identyfikatora, przypisanie i aktualizacja stanu raportu o defekcie w ramach przepływu pracy itd.). Defekty wykryte podczas testowania statycznego (a zwłaszcza w trakcie przeglądów) są zwykle dokumentowane w inny sposób, np. w notatkach ze spotkania związanego z przeglądem.