Certyfikowany Tester

Poziom Zaawansowany
Analityk Testów
(CTAL-TA)

Sylabus

wersja 4.0

wersja tłumaczenia PL 4.0.1

International Software Testing Qualifications Board

Informacja o prawach autorskich

Historia zmian

Wersja	Data	Uwagi
4.0	02.05.2025	Generalna aktualizacja z ogólnym przeglądem i aktualizacją zakresu
3.1.2	31.01.2022	Errata. Poprawiono drobne usterki formatowania, gramatyki i sformułowań
3.1.1	15.05.2021	Errata. Informacja o prawach autorskich została dostosowana do aktualnych standardów ISTQB®
3.1	03.03.2021	Drobna aktualizacja z przeredagowaniem sekcji 3.2.3 i różnymi poprawkami sformułowań
3.0 (2019)	19.10.2019	Generalna aktualizacja z ogólnym przeglądem i zmniejszeniem zakresu
2.0 (2012)	19.10.2012	Pierwsza wersja jako odrębny sylabus CTAL-TA

Historia zmian polskiej wersji dokumentu

Wersja	Data	Uwagi
4.0	14.03.2026	Opublikowanie polskiego tłumaczenia sylabusa

Spis treści

Podziękowania

Zgromadzenie Ogólne ISTQB® formalnie opublikowało ten dokument w dniu 2.05.2025. Został on opracowany przez zespół International Software Testing Qualifications Board w składzie: Armin Born, Filipe Carlos, Wim Decoutere, István Forgács, Matthias Hamburg (Product Owner), Attila Kovács, Sandy Liu, François Martin, Stuart Reid, Adam Roman, Jan Sabak, Murian Song, Tanja Tremmel, Marc-Florian Wendland, Tao Xian Feng.

Zespół dziękuje Julii Sabatine za korektę, Gary'emu Mogyorodi za przegląd techniczny, Danielowi Polanowi za jego stałe wsparcie techniczne, a także zespołowi recenzentów i radom krajowym za ich sugestie i uwagi.

W recenzowaniu, komentowaniu i głosowaniu nad niniejszym sylabusem uczestniczyły następujące osoby:

Aktualne wydanie (v4.0): Gergely Ágnecz, Laura Albert, Dani Almog, Somying Atiporntham, Andre Baumann, Lars Bjørstrup, Ralf Bongard, Earl Burba, Filipe Carlos, Renzo Cerquozzi, Kanitta Chantaramanon, Alessandro Collino, Nicola De Rosa, Aneta Derkova, Dingguofu, Zheng Dandan, Karol Frühauf, Dinu Gamage, Chen Geng, Sabine Gschwandtner, Ole Chr. Hansen, Zsolt Hargitai, Tharushi Hettiarachchi, Ágota Horváth, Arnika Hryszko, Caroline Julien, Beata Karpińska, Ramit Manohar Kaul, Mattijs Kemmink, László Kvintovics, Thomas Letzkus, Marek Majerník, Donald Marcotte, Dénes Medzihradszky, Imre Mészáros, Krisztián Miskó, Gary Mogyorodi, Ebbe Munk, Maysinee Nakmanee, Ingvar Nordström, Tal Pe'er, Kunlanan Peetijade, Sahani Pinidiya, Lukáš Piška, Nishan Portoyan, Meile Posthuma, Yasassri Ratnayake, Randall Rice, Adam Roman, Marc Rutz, Jan Sabak, Vimukthi Saranga, Sumuduni Sasikala, Gil Shekel, Radoslaw Smilgin, Péter Sótér , Helder Sousa, Richard Taylor, Benjamin Timmermans, Giancarlo Tomasig, Robert Treffny, Shun Tsunoda, Stephanie Ulrich, François Vaillancourt, Linda Vreeswijk, Carsten Weise, Marc-Florian Wendland, Paul Weymouth, Elżbieta Wiśniewska, John Young, Claude Zhang.

Edycja 2021–2022 (v3.1): Gery Ágnecz, Armin Born, Chenyifan, Klaudia Dussa-Zieger, Chen Geng (Kevin), Istvan Gercsák, Richard Green, Ole Chr. Hansen, Zsolt Hargitai, Andreas Hetz, Tobias Horn, Joan Killeen, Attila Kovacs, Rik Marselis, Marton Matyas, Blair Mo, Gary Mogyorodi, Ingvar Nordström, Tal Pe'er, Palma Polyak, Nishan Portoyan, Meile Posthuma, Stuart Reid, Murian Song, Péter Sótér, Lucjan Stapp, Benjamin Timmermans, Chris van Bael, Stephanie van Dijck, Paul Weymouth.

Następnie Tal Pe'er, Stuart Reid, Marc-Florian Wendland i Matthias Hamburg zasugerowali poprawki formalne, gramatyczne i słowne, które zostały wdrożone i opublikowane w Erratach 3.1.1 i 3.1.2.

Wydanie 2019 (v3.0): Laura Albert, Markus Beck, Henriett Braunné Bokor, Francisca Cano Ortiz, Guo Chaonian, Wim Decoutere, Milena Donato, Klaudia Dussa-Zieger, Melinda Eckrich-Brajer, Péter Földházi Jr, David Frei, Chen Geng, Matthias Hamburg, Zsolt Hargitai, Zhai Hongbao, Tobias Horn, Ágota Horváth, Beata Karpinska, Attila Kovács, József Kreisz, Dietrich Leimsner, Ren Liang, Claire Lohr, Ramit Manohar Kaul, Rik Marselis, Marton Matyas, Don Mills, Blair Mo, Gary Mogyorodi, Ingvar Nordström, Tal Peer, Pálma Polyák, Meile Posthuma, Lloyd Roden, Adam Roman, Abhishek Sharma, Péter Sótér, Lucjan Stapp, Andrea Szabó, Jan te Kock, Benjamin Timmermans, Chris Van Bael, Erik van Veenendaal, Jan Versmissen, Carsten Weise, Robert Werkhoven, Paul Weymouth.

Wydanie 2012 (v2.0): Graham Bath, Arne Becher, Rex Black, Piet de Roo, Frans Dijkman, Mats Grindal, Kobi Halperin, Bernard Homès, Maria Jönsson, Junfei Ma, Eli Margolin, Rik Marselis, Don Mills, Gary Mogyorodi, Stefan Mohacsi, Reto Mueller, Thomas Mueller, Ingvar Nordstrom, Tal Pe'er, Raluca Madalina Popescu, Stuart Reid, Jan Sabak, Hans Schaefer, Marco Sogliani, Yaron Tsubery, Hans Weiberg, Paul Weymouth, Chris van Bael, Jurian van der Laar, Stephanie van Dijk, Erik van Veenendaal, Wenqiang Zheng, Debi Zylbermann.

Cel niniejszego sylabusa

Niniejszy sylabus stanowi podstawę dla międzynarodowej kwalifikacji ISTQB® w zakresie testowania oprogramowania dla poziomu Certyfikowany Tester Poziom Zaawansowany Analityk Testów. ISTQB® udostępnia niniejszy sylabus:

1. Radom krajowym w celu przetłumaczenia go na języki lokalne i akredytacji dostawców szkoleń. Rady krajowe mogą dostosować sylabus do swoich potrzeb językowych i zmodyfikować źródła bibliograficzne, aby dostosować je do lokalnych publikacji.
2. Organom certyfikującym w celu opracowania pytań egzaminacyjnych w ich lokalnym języku, dostosowanych do celów nauczania określonych w niniejszym sylabusie.
3. Organizatorom szkoleń w celu opracowania materiałów szkoleniowych i określenia odpowiednich metod nauczania.
4. Kandydatom do certyfikacji w celu przygotowania się do egzaminu certyfikacyjnego (w ramach kursu szkoleniowego lub samodzielnie).
5. Międzynarodowej społeczności inżynierów oprogramowania i systemów w celu rozwoju zawodu testera oprogramowania i systemów oraz jako podstawa do tworzenia książek i artykułów.

Analityk testów w testowaniu oprogramowania

Certyfikat ISTQB® Certyfikowany Tester Poziom Zaawansowany Analityk Testów (CTAL-TA) świadczy o posiadaniu umiejętności potrzebnych do przeprowadzania ustrukturyzowanych i dokładnych testów oprogramowania w całym cyklu wytwarzania oprogramowania. Szczegółowo opisuje rolę i obowiązki analityka testów na każdym etapie standardowego procesu testowego i rozszerza opisane w sylabusie poziomu podstawowego ważne techniki testowania. Certyfikat ISTQB® Certyfikowany Tester Poziom Zaawansowany Analityk Testów jest przeznaczony dla osób posiadających certyfikat ISTQB® poziom podstawowy, które chcą dalej rozwijać się w zakresie analizy testów i technik testowania.

W tym sylabusie analityk testów jest rozumiany jako rola, która:

- koncentruje się bardziej na potrzebach biznesowych klienta niż na technicznych aspektach testowania,
- przeprowadza głównie testy funkcjonalne, ale uczestniczy również w testach niefunkcjonalnych skoncentrowanych na użytkowniku, takich jak testy użyteczności, adaptowalności, instalowalności lub współdziałania,
- wykorzystuje raczej czarnoskrzynkowe techniki testowania i techniki testowania opartego na doświadczeniu niż białoskrzynkowe techniki testowania,
- poprawia skuteczność testowania przy użyciu technik zapobiegania defektom.

Ścieżki kariery dla testerów

Program ISTQB® wspiera profesjonalistów testowania na wszystkich etapach ich kariery. Osoby, które uzyskają certyfikat ISTQB® Certyfikowany Tester Poziom Zaawansowany Analityk Testów mogą być również zainteresowane innymi certyfikatami poziomu zaawansowanego (techniczny

analityk testów, zarządzanie testami, inżynieria automatyzacji testów), a następnie poziomu eksperckiego (wdrażanie udoskonaleń procesu testowego, zarządzanie testami). Każdy, kto chce rozwinąć umiejętności w zakresie praktyk testowania w środowisku zwinnym (agile), może rozważyć certyfikaty *Agile Technical Tester* (techniczny tester zwinny) lub *Agile Test Leadership at Scale*. Ponadto specjalistyczne ścieżki oferują produkty certyfikacyjne koncentrujące się na konkretnych technologiach i podejściach, określonych charakterystykach jakościowych oprogramowania i poziomach testów lub testowaniu w określonych dziedzinach przemysłu. Najnowsze informacje na temat programu certyfikacji testerów ISTQB® można znaleźć na stronie www.istqb.org.

Cele biznesowe

W tej sekcji wymieniono cele biznesowe (ang. *Business Outcomes*) osiągane przez kandydata, który uzyskał certyfikat ISTQB® poziom zaawansowany – analityk testów. Certyfikowany Tester Poziom Zaawansowany Analityk Testów...

Kod	Opis celu biznesowego
TA-BO1	Wspiera i przeprowadza odpowiednie testy w ramach cyklu wytwarzania oprogramowania
TA-BO2	Stosuje podejście do testowania oparte na ryzyku
TA-BO3	Wybiera i stosuje odpowiednie techniki testowania, aby wspierać osiąganie celów testów
TA-BO4	Dostarcza dokumentację o odpowiednim poziomie szczegółowości i jakości
TA-BO5	Określa odpowiednie rodzaje testów funkcjonalnych, które należy przeprowadzić
TA-BO6	Bierze udział w testach niefunkcjonalnych
TA-BO7	Wnosi wkład w zapobieganie defektom
TA-BO8	Poprawia wydajność procesu testowego, w tym przy użyciu narzędzi
TA-BO9	Określa wymagania dotyczące środowisk testowych i danych testowych

Cele nauczania oraz poziomy wiedzy

Cele nauczania (ang. *Learning Objectives*) wspierają cele biznesowe i są podstawą do tworzenia pytań egzaminacyjnych w egzaminach ISTQB® Certyfikowany Tester Poziom Zaawansowany Analityk Testów. Co do zasady, cała zawartość rozdziałów 1–5 niniejszego sylabusa podlega egzaminowaniu. Pytania egzaminacyjne weryfikują znajomość słów kluczowych na poziomie K1 oraz cele nauczania na odpowiednim poziomie wiedzy (zob. niżej, patrz także Dodatek A). Cele nauczania i odpowiadające im poziomy wiedzy są przedstawione na początku każdego rozdziału i sklasyfikowane w następujący sposób:

- K2 – zrozumieć,
- K3 – zastosować,
- K4 – przeanalizować.

W przypadku wszystkich terminów wymienionych jako słowa kluczowe tuż pod nagłówkami rozdziałów, należy zapamiętać (K1) ich poprawne nazwy oraz definicje ze słownika ISTQB® (ISTQB Glossary), nawet jeśli konieczność ich znajomości nie została wyraźnie wymieniona w żadnym celu nauczania. Szczegółowy opis i przykłady celów nauczania opisano w Dodatku A.

Egzamin certyfikacyjny Certyfikowany Tester Poziom Zaawansowany Analityk Testów

Egzamin na certyfikat ISTQB® Certyfikowany Tester Poziom Zaawansowany Analityk Testów jest oparty na niniejszym sylabusie. Odpowiedzi na pytania egzaminacyjne mogą wymagać wykorzystania materiałów opartych na więcej niż jednej sekcji niniejszego sylabusa. Wszystkie sekcje sylabusa podlegają egzaminowaniu, z wyjątkiem wprowadzenia i dodatków. Normy i książki są uwzględnione jako odniesienia, ale ich treść nie podlega egzaminowaniu, poza tym, co zostało opisane w samym sylabusie.

Więcej informacji o egzaminie można znaleźć w dokumentach *Exam Structure and Rules* oraz *Exam Structure Tables* dostępnych na podstronie www.istqb.org dotyczącej sylabusa Certyfikowany Tester Poziom Zaawansowany Analityk Testów v4.0 (CTAL TA). Do egzaminu ISTQB® CTAL TA może przystąpić każdy, kto jest zainteresowany testowaniem oprogramowania. Zaleca się jednak, aby kandydaci również:

- posiadali przynajmniej minimalne doświadczenie w tworzeniu lub testowaniu oprogramowania, np. sześciomiesięczne doświadczenie jako tester lub jako programista,
- wzięli udział w szkoleniu akredytowanym przez jedną z uznanych rad krajowych ISTQB®, zgodnie ze standardami ISTQB®.

Wymaganiem wstępnym przed przystąpieniem do egzaminu ISTQB® Certyfikowany Tester Poziom Zaawansowany Analityk Testów jest posiadanie certyfikatu ISTQB® Certyfikowany Tester Poziom Podstawowy.

Akredytacja

Rada krajowa ISTQB® może akredytować dostawców szkoleń, których materiały szkoleniowe są zgodne z niniejszym sylabusem. Dostawcy szkoleń powinni uzyskać wytyczne dotyczące akredytacji od rady krajowej lub organu, który przeprowadza akredytację. Akredytowane szkolenie jest uznawane za zgodne z niniejszym sylabusem i umożliwia włączenie egzaminu ISTQB® do szkolenia. Wytyczne akredytacyjne dla niniejszego sylabusa są zgodne z ogólnymi wytycznymi akredytacyjnymi opublikowanymi przez ISTQB® *Processes Management and Compliance Working Group*.

Odniesienia do norm

Międzynarodowe organizacje normalizacyjne, takie jak IEEE i ISO, wydają normy związane z charakterystykami jakościowymi i testowaniem oprogramowania. Niektóre z tych norm są przywoływane w niniejszym sylabusie. Celem tych odniesień jest zapewnienie ram pojęciowych (jak w odniesieniach do normy ISO/IEC 25010 dotyczącej charakterystyk jakościowych oprogramowania) lub źródła dodatkowych informacji, jeśli jest to pożądane przez czytelnika. Należy pamiętać, że sylabusy ISTQB® wykorzystują normy jako odniesienie. Zawartość tych dokumentów nie podlega egzaminowaniu. Więcej informacji o normach znajduje się w Rozdziale 6.

Poziom szczegółowości

Poziom szczegółowości niniejszego sylabusa umożliwia prowadzenie szkoleń i egzaminów spójnych w skali międzynarodowej. Aby osiągnąć ten cel, w sylabusie uwzględniono:

- ogólne cele instruktażowe dotyczące roli analityka testów na poziomie zaawansowanym,
- listy słów kluczowych, których definicje kandydaci muszą pamiętać,
- cele nauczania dla każdego obszaru wiedzy, opisujące poznawcze efekty uczenia się, które mają zostać osiągnięte przez kandydatów,
- opis kluczowych pojęć, w tym odniesień do źródeł, takich jak uznana literatura lub normy.

Zawartość sylabusa nie opisuje całego obszaru wiedzy na temat testowania oprogramowania; odzwierciedla raczej poziom szczegółowości, który ma zostać omówiony na szkoleniach dla analityków testów na poziomie zaawansowanym.

Terminologia (tj. terminy i ich znaczenie) z zakresu testowania i zapewniania jakości oprogramowania wykorzystywana w niniejszym sylabusie jest zgodna ze słownikiem terminów testowych ISTQB® (ISTQB Glossary).

Terminologię dotyczącą dyscyplin powiązanych można znaleźć w słowniku inżynierii wymagań IREB (IREB-CPRE) oraz słowniku inżynierii oprogramowania IEEE (IEEE-Pascal).

Struktura sylabusa

Sylabus zawiera pięć rozdziałów z treściami podlegającymi egzaminowaniu. W nagłówku każdego rozdziału określony jest czas przeznaczony na omówienie tego rozdziału podczas szkolenia. Czas ten nie jest podawany poniżej poziomu rozdziału. W przypadku akredytowanych szkoleń sylabus wymaga co najmniej 20,25 godzin zajęć (1215 minut), rozłożonych na pięć rozdziałów w następujący sposób:

Rozdział 1 (225 minut): Zadania analityka testów w procesie testowym

- Kandydat dowiaduje się, jak wygląda zaangażowanie analityk testów w różnych cyklach wytwarzania oprogramowania.
- Kandydat dowiaduje się, w jaki sposób analityk testów jest zaangażowany w różne działania testowe.
- Kandydat poznaje zadania wykonywane przez analityka testów związane z testaliami (produktami pracy procesu testowego).

Rozdział 2 (90 minut): Zadania analityka testów w testowaniu opartym na ryzyku

- Kandydat dowiaduje się, w jaki sposób analityk testów przyczynia się do analizy ryzyka produktowego.
- Kandydat uczy się, jak analizować wpływ zmian w celu określenia zakresu testów regresji.

Rozdział 3 (615 minut): Analiza i projektowanie testów

- Kandydat poznaje techniki testowania oparte na danych, takie jak testowanie dziedziny, testowanie kombinatoryczne i testowanie losowe.
- Kandydat poznaje techniki testowania oparte na zachowaniu, takie jak testowanie CRUD, testowanie przejść pomiędzy stanami i testowanie oparte na scenariuszach.
- Kandydat poznaje techniki testowania oparte na regułach, takie jak testowanie w oparciu

o tablicę decyzyjną i testowanie metamorficzne.
- Kandydat poznaje techniki testowania oparte na doświadczeniu, takie jak testowanie w sesjach i testowanie w tłumie.
- Kandydat dowiaduje się, jak wybrać odpowiednie techniki testowania w celu łagodzenia ryzyk produktowych.

Rozdział 4 (60 minut): Testowanie charakterystyk jakościowych

- Kandydat uczy się, jak przeprowadzać różne rodzaje testów funkcjonalnych (testowanie kompletności funkcjonalnej, poprawności funkcjonalnej i adekwatności funkcjonalnej).
- Kandydat dowiaduje się, jak wykorzystać konkretną wiedzę na temat funkcjonalności, aby wnieść wkład do testów niefunkcjonalnych, takich jak testowanie użyteczności, testowanie przenaszalności (elastyczności) i testowanie zgodności.

Rozdział 5 (225 minut): Zapobieganie defektom

- Kandydat poznaje różne praktyki zapobiegania defektom (ang. *defect prevention*).
- Kandydat poznaje różne podejścia wspierające powstrzymanie fazowe (ang. *phase containment*).
- Kandydat dowiaduje się, jak ograniczyć ponowne występowanie defektów (ang. *defect recurrence*).

1. Zadania analityka testów w procesie testowym (225 min.)

Słowa kluczowe

analityk testów, analiza testów, cykl wytwarzania oprogramowania, dane testowe, implementacja testów, projektowanie testów, przypadek testowy, przypadek testowy niskiego poziomu, przypadek testowy wysokiego poziomu, skrypt testowy, słowo kluczowe, środowisko testowe, testalia, testowanie oparte na słowach kluczowych, warunek testowy, wykonywanie testów, wyrocznia testowa

Cele nauczania dla rozdziału 1

1.1 Testowanie w cyklu wytwarzania oprogramowania

TA-1.1.1 (K2) Podsumować zaangażowanie analityka testów w różnych modelach wytwarzania oprogramowania.

1.2 Zaangażowanie w czynności testowe

TA-1.2.1 (K2) Podsumować zadania wykonywane przez analityka testów w ramach analizy testów.

TA-1.2.2 (K2) Podsumować zadania wykonywane przez analityka testów w ramach projektowania testów.

TA-1.2.3 (K2) Podsumować zadania wykonywane przez analityka testów w ramach implementacji testów.

TA-1.2.4 (K2) Podsumować zadania wykonywane przez analityka testów w ramach wykonywania testów.

1.3 Zadania związane z testaliami

TA-1.3.1 (K2) Rozróżniać przypadki testowe wysokiego i niskiego poziomu.

TA-1.3.2 (K2) Wyjaśnić kryteria jakości dla przypadków testowych.

TA-1.3.3 (K2) Podać przykłady wymagań dla środowiska testowego.

TA-1.3.4 (K2) Wyjaśnić problem wyroczni testowej i jego możliwe rozwiązania.

TA-1.3.5 (K2) Podać przykłady wymagań dla danych testowych.

TA-1.3.6 (K3) Zastosować testowanie oparte na słowach kluczowych, aby opracować skrypty testowe.

TA-1.3.7 (K2) Podsumować typy narzędzi do zarządzania testaliami.

Wstęp

Sylabus (ISTQB CTFL) opisuje dwie główne role w testowaniu: rolę zarządzania testowaniem i rolę testera. W tym sylabusie osoba pełniąca rolę testera odpowiedzialnego za testowanie aspektów biznesowych oprogramowania nazywana jest **analitykiem testów (AT)**. Chociaż odpowiedzialność ta rzadko jest przypisywana do konkretnego stanowiska lub osoby, AT ma jasno określone zadania i wymagane kompetencje. Jeśli chodzi o poziomy testów, AT koncentruje się na testach systemowych, testach akceptacyjnych i testach integracji systemów. Jeśli chodzi o cechy jakościowe, kompetencje AT koncentrują się na funkcjonalności, a także obejmują pewne niefunkcjonalne charakterystyki jakościowe ukierunkowane na użytkownika, takie jak użyteczność, adaptowalność, instalowalność i współdziałanie.

1.1. Testowanie w cyklu wytwarzania oprogramowania

Organizacja czynności testowych może się różnić w zależności od stosowanego cyklu wytwarzania oprogramowania (SDLC). W związku z tym zaangażowanie AT w działania testowe może się różnić w zależności od przyjętego modelu SDLC.

W **sekwencyjnych modelach wytwarzania oprogramowania** czynności wytwórcze są przeprowadzane w fazach i rozpoczynają się dopiero po zakończeniu fazy poprzedniej. Zazwyczaj w niewielkim stopniu nachodzą na siebie. Dlatego też zadania AT zwykle zmieniają się w czasie. We wczesnych fazach SDLC, AT koncentruje się na wspieraniu planowania testów. AT rozpoczyna analizę testów, gdy dostępna jest podstawa testów. Projektowanie i implementacja testów odbywają się równolegle z projektowaniem i implementacją oprogramowania. AT wykonuje testy i wspiera fazę ukończenia testów w późnych fazach SDLC.

Przyrostowe modele wytwarzania oprogramowania dzielą oprogramowanie na mniejsze, zarządzalne przyrosty. Każdy przyrost jest opracowywany i testowany niezależnie. W związku z tym AT wykonuje te same czynności dla każdego przyrostu (tj. analizę, projektowanie, implementację, wykonywanie testów oraz wsparcie ukończenia testów). Praca AT może być jednak zorganizowana inaczej dla każdego przyrostu. Testowanie koncentruje się na nowych lub zmodyfikowanych funkcjach. Ponadto ze względu na zwiększone ryzyko regresji, AT musi zwrócić szczególną uwagę na refaktoryzację i zawartość zestawów testów regresji.

W **iteracyjnych modelach wytwarzania oprogramowania** proces rozwoju jest cykliczny. Projekt przechodzi powtarzające się cykle prototypowania, testowania, udoskonalania i wdrażania. Rola AT jest dynamiczna i adaptacyjna. AT ściśle współpracuje z programistami i przedstawicielami biznesu, dostosowując się do ewoluującego produktu. AT dostosowuje i modyfikuje warunki testowe i przypadki testowe w miarę ewolucji oprogramowania oraz przekazuje informacje zwrotne w celu doskonalenia procesu testowego w każdej iteracji. Im częstsze są iteracje, tym bardziej krytyczna jest bieżąca pielęgnacja i rozwój testów regresji przez AT.

SDLC może łączyć elementy różnych modeli wytwarzania oprogramowania oraz konkretnych technik i podejść (np. zwinne wytwarzanie oprogramowania łączy w sobie aspekty zarówno modeli iteracyjnych, jak i przyrostowych). W takich przypadkach zaangażowanie AT będzie zależeć od konkretnych cech SDLC i sposobu ich połączenia. Dobrą praktyką wspólną dla wszystkich modeli SDLC jest to, że AT powinien być zaangażowany już od początkowych faz SDLC.

1.2. Zaangażowanie w czynności testowe

Sylabus Certyfikowany Tester Poziom Podstawowy (ISTQB CTFL) opisuje siedem grup czynności w ramach procesu testowego. AT koncentruje się głównie na czterech z nich: analizie testów, projektowaniu testów, implementacji testów i wykonywaniu testów. Zadania AT w ramach tych czterech grup czynności zostały szczegółowo opisane w poniższych sekcjach.

1.2.1. Analiza testów

Podczas analizy testów, AT sprawdza kompletność podstawy testów i zbiera wszelkie dodatkowe informacje istotne dla testowania. Obejmuje to nie tylko dokumentację, ale także informacje werbalne, np. rozmowy podczas wspólnego pisania historyjek użytkownika (patrz ISTQB CTFL, sekcja 4.5.1). Zmiany w podstawie testów mogą prowadzić do dostosowania zakresu testów w porozumieniu z kierownikiem testów.

Aby skutecznie przeprowadzić analizę testów, AT sprawdza następujące kryteria wejścia:

- Przeprowadzono planowanie testów, a zakres testów, cele testów i podejście do testów są jasne.
- Podstawa testów (zawierająca informacje takie jak wymagania lub historyjki użytkownika) jest zdefiniowana.
- Zidentyfikowane już ryzyka produktowe zostały ocenione i w razie potrzeby udokumentowane.

AT ocenia podstawę testów, aby zidentyfikować wszelkie defekty, które może zawierać i ocenić jej testowalność, zapewniając w ten sposób wczesne informacje zwrotne właścicielom produktu. Może to obejmować modelowanie zachowania systemu zgodnie z technikami testowania, które mają zostać zastosowane (patrz rozdział 3 oraz sekcja 5.2.1). Jako część procesu stosowane są również techniki przeglądu (patrz sekcja 5.2.2). Defekty w podstawie testów, jeśli nie zostały bezpośrednio usunięte, muszą zostać udokumentowane. Ponadto AT określa potrzebne wyrocznie testowe (patrz sekcja 1.3.4).

AT definiuje i priorytetyzuje warunki testowe dla każdego elementu testowego w zakresie. Warunki testowe odnoszą się do celów testów (patrz ISTQB CTFL, sekcja 1.1.1) i muszą być śledzone do elementów podstawy testów. Zakres i cel warunków testowych uwzględniają ryzyka produktowe. W przyrostowych lub iteracyjnych modelach wytwarzania obejmuje to określenie zakresu testów regresji na podstawie analizy wpływu. W zwinnym wytwarzaniu oprogramowania warunki testowe można wyrazić jako kryteria akceptacji, które odzwierciedlają ryzyka związane z historyjkami użytkownika.

AT może działać etapowo, zaczynając od definiowania warunków testowych wysokiego poziomu, takich jak „funkcjonalność ekranu x”. Następnie AT definiuje bardziej szczegółowe (niskopoziomowe) warunki testowe, takie jak „ekran x odrzuca numery kont, które są o jedną cyfrę za krótkie”. Takie podejście zapewnia wystarczające pokrycie i umożliwia wczesne rozpoczęcie projektowania testów, np. dla historyjek użytkownika, które nadal wymagają dopracowania.

AT angażuje interesariuszy w przegląd warunków testowych, aby upewnić się, że podstawa testów jest zrozumiała, a testowanie jest zgodne z celami testów.

1.2.2. Projektowanie testów

Projektowanie testów opisuje sposób przeprowadzania testów w celu osiągnięcia określonych celów testów. Zazwyczaj odbywa się to za pomocą przypadków testowych. Sposób projektowania testów zależy od wielu czynników, w tym wymaganego pokrycia, podstawy testów, SDLC, ograniczeń projektowych oraz wiedzy i doświadczenia testerów.

Podczas projektowania testów AT określa, w których obszarach odpowiednie będą przypadki testowe niskiego lub wysokiego poziomu (patrz sekcja 1.3.1). W obu przypadkach musi on określić jasne kryteria zaliczenia/niezaliczenia. AT projektuje przypadki testowe dla nowych lub zmienionych warunków testowych zgodnie z kryteriami jakości (patrz sekcja 1.3.2). W przypadku testów regresji zazwyczaj wystarczający jest wybór istniejących przypadków testowych wysokiego poziomu lub dostosowanie istniejących przypadków testowych niskiego poziomu w oparciu o ich priorytetyzację.

AT rejestruje śledzenie powiązań między podstawą testów, warunkami testowymi a przypadkami testowymi. W testowaniu opartym na doświadczeniu przypadki testowe nie zawsze są udokumentowane; zamiast tego wykonywaniem testów mogą kierować (między innymi) warunki testowe. AT może zaprojektować niektóre przypadki testowe w oparciu o wysokopoziomowe cele testów.

Oprócz powyższych zadań, AT definiuje wymagania dotyczące środowiska testowego (patrz sekcja 1.3.3) oraz identyfikuje, tworzy i określa wymagania dotyczące danych testowych (patrz sekcja 1.3.5).

AT wykorzystuje kryteria wyjścia zdefiniowane podczas planowania testów w celu określenia, kiedy zaprojektowano wystarczającą liczbę przypadków testowych. Jednak inne kryteria wyjścia, takie jak poziom ryzyka rezydualnego lub ograniczenia projektowe (np. budżet lub czas), również wskazują, kiedy projektowanie testów może się zakończyć.

Projektowanie testów może być wspierane przez narzędzia, ale – w kwestiach merytorycznych – powinno być od nich niezależne. Projektowanie testów stosuje systematyczne podejście z wykorzystaniem technik testowania lub jest doraźne (ang. *ad hoc*).

Przypadki testowe pełnią rolę komunikacyjną i powinny być zrozumiałe dla odpowiednich interesariuszy. Ponieważ przypadek testowy nie zawsze musi być wykonywany przez jego autora, inni testerzy muszą rozumieć sposób jego wykonania, cele testów (tj. podstawowe warunki testowe) oraz jego znaczenie. Przypadki testowe muszą być również zrozumiałe dla programistów, którzy mogą je wdrażać lub powtarzać w przypadku niepowodzenia, a także dla audytorów, którzy mogą być odpowiedzialni za ich zatwierdzenie.

1.2.3. Implementacja testów

Podczas implementacji testów, AT dostarcza testalia niezbędne do wykonywania testów. AT może organizować procedury testowe i skrypty testowe w zestawy testowe lub sugerować przypadki testowe do automatyzacji. Definiowanie procedur testowych wymaga starannego określenia ograniczeń i zależności, które mogą mieć wpływ na kolejność wykonywania testów. Oprócz kroków zawartych w przypadkach testowych, procedury testowe obejmują kroki konfiguracji warunków wstępnych (np. ładowanie danych testowych z repozytorium), weryfikację

oczekiwanych wyników i warunków końcowych oraz przywracanie ustalonego stanu systemu i środowiska po wykonaniu (np. resetowanie bazy danych, środowiska i systemu).

AT nadaje priorytet procedurom testowym i skryptom testowym do wykonania w oparciu o kryteria priorytetyzacji zidentyfikowane podczas analizy ryzyka i planowania testów oraz identyfikuje procedury testowe lub skrypty testowe, które powinny zostać wykonane dla bieżącej wersji przedmiotu testów. Umożliwia to wykonanie powiązanych testów (np. dla nowych funkcji lub testów regresji) razem w określonym przebiegu testowym. AT aktualizuje śledzenie powiązań między podstawą testów a innymi testaliami takimi jak procedury testowe, skrypty testowe i zestawy testowe.

AT może pomóc kierownikowi testów w zdefiniowaniu harmonogramu wykonywania testów, w tym alokacji zasobów, aby umożliwić wydajne wykonywanie testów poprzez zdefiniowanie kolejności ich wykonywania (patrz ISTQB CTFL, sekcja 5.1.5).

AT tworzy wejściowe i środowiskowe dane testowe do załadowania do baz danych i innych repozytoriów (patrz sekcja 1.3.5). Dane te muszą być „odpowiednie do celu" (ang. *fit for purpose*), aby wspierać określone cele testów.

AT powinien również zweryfikować, czy środowisko testowe jest w pełni skonfigurowane i gotowe do wykonywania testów (patrz sekcja 1.3.3). W tym celu najlepiej jest zaprojektować i uruchomić test dymny. Środowisko testowe powinno ujawniać defekty przedmiotu testów poprzez wykonanie testu, działać normalnie, gdy nie wystąpią awarie i odpowiednio odzwierciedlać, w razie potrzeby, środowisko produkcyjne lub środowisko użytkownika końcowego.

Poziom szczegółowości i związana z nim złożoność prac wykonywanych podczas implementacji testów może zależeć od poziomu szczegółowości warunków testowych i przypadków testowych. W niektórych przypadkach mogą mieć zastosowanie zasady regulacyjne, a testalia powinny dostarczać dowodów zgodności z obowiązującymi normami (np. RTCA DO-178C, 2011).

1.2.4. Wykonywanie testów

Wykonywanie testów odbywa się zgodnie z harmonogramem wykonywania testów. Typowe zadania AT to uruchamianie testów, porównywanie rzeczywistych wyników z oczekiwanymi, analizowanie anomalii, zgłaszanie defektów i rejestrowanie wyników testów.

AT wykonuje testy manualnie. Obejmuje to testowanie eksploracyjne, wykonywanie procedur testowych, testy regresji i testy potwierdzające. W przypadku testowania eksploracyjnego, AT może korzystać z testowania w sesjach z kartami opisu testów (patrz sekcja 3.4.1). AT może również uruchamiać zautomatyzowane skrypty testowe, ale zarówno ta czynność, jak i analizowanie skryptów w przypadku awarii, mogą być zadaniem programistów, inżynierów automatyzacji testów lub technicznych analityków testów.

AT analizuje anomalie występujące podczas manualnego lub automatycznego wykonywania testów w celu ustalenia ich prawdopodobnych przyczyn. Anomalia może być konsekwencją defektu w przedmiocie testów. Mogą jednak istnieć również inne przyczyny, w tym brak warunków wstępnych, nieprawidłowe dane testowe, defekty skryptów testowych lub środowiska testowego, czy też niezrozumienie specyfikacji. AT rejestruje rzeczywiste wyniki wykonania testów i komunikuje (lub raportuje, jeśli to konieczne) defekty w oparciu o zaobserwowane awarie.

AT aktualizuje śledzenie powiązań między podstawą testów a innymi testaliami, biorąc pod uwagę wyniki testów. Informacje te umożliwiają przekształcenie wyników testów w wysokopoziomowe informacje o ryzyku lub pokryciu, co pozwala interesariuszom na podejmowanie świadomych decyzji. Na przykład, interesariusze mogą być poinformowani o tym, ile przypadków testowych związanych z konkretnym warunkiem testowym zostało zaliczonych lub niezaliczonych.

Oprócz typowych zadań opisanych powyżej, AT ocenia wyniki testów, w tym:

- Rozpoznaje skupiska defektów, które mogą wskazywać na potrzebę przeprowadzenia większej liczby testów określonej części przedmiotu testów (patrz sekcja 5.3.1).
- Wykonuje manualnie zautomatyzowane testy, które nie zostały zaliczone, aby upewnić się, że błędy w automatyzacji testów nie dały wyniku fałszywie pozytywnego.
- Sugeruje dodatkowe testy w oparciu o doświadczenie zdobyte podczas poprzednich testów.
- Identyfikuje nowe ryzyka na podstawie informacji uzyskanych podczas wykonywania testów.
- Sugeruje ulepszenia w projektowaniu lub implementacji testów (np. ulepszenia procedur testowych), a nawet w systemie podlegającym testowaniu.
- Sugeruje ulepszenia w zestawach testów regresji, w tym refaktoryzację, dostosowanie zakresu i automatyzację testów (patrz sekcja 2.2).

1.3. Zadania związane z testaliami

AT musi zapewnić jakość testaliów, za które jest odpowiedzialny. Obejmuje to przypadki testowe, środowiska testowe, dane testowe, wyrocznie testowe i skrypty testowe. W tym podrozdziale sylabusa omówiono zadania AT związane z testaliami, a także rodzaje narzędzi do zarządzania testaliami.

1.3.1. Przypadki testowe wysokiego i niskiego poziomu

Przypadek testowy wysokiego poziomu (zwany również *abstrakcyjnym* lub *logicznym przypadkiem testowym*) opisuje okoliczności, w których badany jest przedmiot testów, wskazując, które warunki testowe są nim objęte. Przypadki testowe wysokiego poziomu są zatem odpowiednie do zapewnienia, że testy obejmują wszystkie istotne warunki testowe. Przypadki testowe wysokiego poziomu nie zawierają konkretnych informacji na temat warunków wstępnych, danych wejściowych, oczekiwanych wyników lub warunków wyjściowych. Są one wyrażone na poziomie abstrakcyjnym (np. „zamów więcej niż jedną książkę o łącznej cenie zamówienia skutkującej rabatem; oczekiwany wynik: rabat jest przyznany”).

Przypadek testowy niskiego poziomu (zwany również *konkretnym* lub *fizycznym przypadkiem testowym*) jest uszczegółowieniem przypadku testowego wysokiego poziomu. Przypadki testowe niskiego poziomu opisują, jakie dane należy przygotować, jakie czynności musi wykonać tester (jeśli to konieczne) oraz jaki jest konkretny oczekiwany wynik. Przypadki testowe niskiego poziomu zawierają określone warunki wstępne, dane wejściowe, oczekiwane wyniki i warunki wyjściowe (np. „zamów książki B1 (10$) i B2 (20$), o łącznej cenie zamówienia 30$; oczekiwany wynik: 10% rabatu, łączna cena: 27$”).

Zazwyczaj AT początkowo projektuje przypadki testowe wysokiego poziomu, które stanowią podstawę do opracowania przypadków testowych niskiego poziomu. Jeden przypadek testowy

wysokiego poziomu może zostać zaimplementowany w jednym lub kilku przypadkach testowych niskiego poziomu. Czasami przypadek testowy może pozostać na wysokim poziomie, a konkretne informacje są określane przez AT podczas wykonywania testu. Przykładowo, wysokopoziomowe przypadki testowe mogą kierować działaniami AT podczas tworzenia celów testów w karcie opisu testów dla testowania w sesjach, umożliwiając AT rozwinięcie tych celów podczas wykonywania testów (patrz sekcja 3.4.1).

Przejście od przypadków testowych wysokiego do niskiego poziomu wykracza poza samo wypełnienie konkretnych wartości danych. Jest to również krok od etapu koncepcyjnego do technicznego. Czynność ta jest często odraczana do etapu implementacji testów, zwłaszcza jeśli potrzebne są określone dane testowe. AT musi zapewnić, że wszystko, co niezbędne do wykonania przypadków testowych niskiego poziomu, jest znane (Koomen *et al.*, 2006). W praktyce wiele przypadków testowych ma charakter hybrydowy, będąc konkretnymi w niektórych aspektach, a abstrakcyjnymi w innych. Jest to często spowodowane kompromisem między łatwością pielęgnacji a zrozumiałością przypadków testowych.

1.3.2. Kryteria jakościowe dla przypadków testowych

Zaniedbanie jakości przypadków testowych może prowadzić do wielu problemów, takich jak wysokie koszty utrzymania, zmniejszona zrozumiałość lub opóźnienia w ich wykonywaniu. Kryteria jakościowe dla przypadków testowych są pierwszym krokiem w kierunku łatwiejszych w utrzymaniu przypadków testowych. Obejmują one:

- **Poprawność**. Przypadek testowy musi ułatwiać dokładną weryfikację warunków testowych, na których jest oparty.
- **Osiągalność**. Wykonanie przypadku testowego musi być możliwe.
- **Niezbędność**. Każdy przypadek testowy powinien obejmować jasny cel testu, wyrażony w jego tytule lub podsumowaniu. Należy unikać duplikatów. Rzeczy, które nie powinny być testowane, nie powinny mieć zaprojektowanych przypadków testowych.
- **Zrozumiałość**. Przypadki testowe mogą być przeglądane, modyfikowane i wykonywane przez osoby inne niż ich autor. AT powinien pisać przypadki testowe w języku i formacie zrozumiałym dla wszystkich zaangażowanych interesariuszy, bez wyjaśniania rzeczy oczywistych. Złożone przypadki testowe powinny zostać uproszczone lub podzielone.
- **Śledzenie powiązań**. Przypadki testowe powinny być powiązane z warunkami testowymi, wymaganiami i ryzykami, aby umożliwić AT zapewnienie ich aktualności (patrz ISTQB CTFL, sekcja 1.4.4).
- **Spójność**. Spójność języka, formatowania i struktury ułatwia zrozumienie i utrzymanie przypadków testowych. W tym celu AT może skorzystać ze słownika.
- **Jasność**. Powinna istnieć tylko jedna interpretacja przypadku testowego, aby uniknąć fałszywie negatywnych i fałszywie pozytywnych wyników testu. Należy unikać niejednoznacznych terminów, takich jak „odpowiedni", „jeśli potrzeba" czy „kilka".
- **Kompletność**. Wszystkie niezbędne atrybuty (patrz np. ISO/IEC/IEEE 29119-3, 2021) powinny być obecne, w tym wymagane dane testowe (patrz sekcja 1.3.5) oraz jasny wynik oczekiwany, aby uniknąć wątpliwości podczas porównywania z wynikiem rzeczywistym.
- **Zwartość**. Granularność przypadków testowych (tj. jeden duży przypadek testowy z wieloma krokami testowymi vs. kilka mniejszych przypadków testowych) powinna odpowiadać podstawie testów i warunkom testowym. Preferowane są mniejsze przypadki

testowe skoncentrowane na kilku elementach pokrycia, ponieważ ułatwiają znalezienie przyczyn awarii, można je elastycznie łączyć w procedury testowe i zestawy testowe, a awaria podczas wykonywania testu nie zablokuje dalszych testów.

Format i poziom szczegółowości przypadków testowych zależy od kontekstu projektu i produktu i powinien zostać uzgodniony w ramach zespołu testowego.

1.3.3. Wymagania dotyczące środowiska testowego

Środowisko testowe jest krytycznym czynnikiem sukcesu zarówno w przypadku manualnego, jak i automatycznego wykonywania testów. Implementacja środowiska testowego wpływa na testowalność, wykrywanie defektów, ogólne koszty testowania oraz wiarygodność wyników testów.

Zaliczony lub niezaliczony przypadek testowy wykonany w środowisku testowym powinien mieć taki sam wynik jak w środowisku produkcyjnym. Idealnie byłoby, gdyby środowisko testowe było stabilne, przewidywalne i, w razie potrzeby, zintegrowane ze strukturą automatyzacji testów. AT może zdefiniować wymagania dotyczące środowiska testowego podczas projektowania testów w oparciu o analizę:

- warunków testowych, przypadków testowych i wymagań wobec danych testowych, przy czym wymagania dotyczące środowiska testowego opisują warunki niezbędne do skonfigurowania i utrzymania środowiska testowego w celu zapewnienia spełnienia warunków wstępnych wykonania testu,
- poziomów i typów testów, które wpływają na kompromis między elastycznością środowiska testowego a jego podobieństwem do środowiska produkcyjnego,
- dostępności i niezależności modułów i systemów, co może wskazywać na potrzebę użycia obiektów imitacji (np. zaślepek lub sterowników).

Wymagania dotyczące środowiska testowego opisują elementy środowiska testowego, które można podzielić na kilka typów, takich jak sprzęt, oprogramowanie pośredniczące (ang. *middleware*), oprogramowanie, usługi zwirtualizowane, sieć, interfejsy, narzędzia, zabezpieczenia, konfiguracja i fizyczna lokalizacja. Dla każdego elementu środowiska testowego wymagania powinny zawierać następujące informacje (ISO/IEC/IEEE 29119-3, 2021):

- unikalny identyfikator – używany do celów identyfikacji i śledzenia powiązań,
- opis – wystarczająco szczegółowy, aby wdrożyć element zgodnie z wymaganiami,
- odpowiedzialność – opisuje, kto jest odpowiedzialny za udostępnienie elementu,
- okres – określa, kiedy i jak długo element jest potrzebny,
- wierność – stopień, w jakim element reprezentuje lub odbiega od środowiska produkcyjnego.

Wymagania dotyczące środowiska testowego powinny również uwzględniać nadrzędne potrzeby środowiska testowego, w tym konfigurację (ang. *setup*), tworzenie kopii zapasowych i przywracanie, potrzeby w zakresie zabezpieczeń, możliwość zmiany środowiska testowego oraz role i autoryzację (Koomen *et al.*, 2006).

AT powinien udokumentować wymagania dotyczące środowiska testowego w sposób jasny, zwarty i spójny. Może w tym celu wykorzystać diagramy lub tabele. W celu uniknięcia redundancji i niepotrzebnej dokumentacji, AT może odwołać się do istniejących środowisk testowych i skupić

się na konkretnych potrzebach poziomu testów. Odpowiedni interesariusze (np. programiści, techniczni analitycy testów, inżynierowie automatyzacji testów, analitycy biznesowi, sponsorzy i właściciele produktów) powinni również przeglądać, zatwierdzać i aktualizować wymagania dotyczące środowiska testowego.

1.3.4. Określenie wyroczni testowych

Wyrocznia testowa (ang. *test oracle*) jest potrzebna do określenia oczekiwanych wyników w testach dynamicznych. W idealnym przypadku podstawa testów zapewnia wyrocznie testowe (np. w specyfikacji tekstowej lub formalnej). Doświadczenie lub wiedza na temat przedmiotu testów może również służyć jako wyrocznia testowa. Zautomatyzowana wyrocznia testowa może być wymagana ze względu na opłacalność (np. jeśli dane wejściowe są generowane automatycznie lub ludzkie wyrocznie są kosztowne).

W zależności od jakości i kompletności podstawy testów lub charakterystyk systemu, efektywna kosztowo wyrocznia testowa może nie być dostępna. Jest to tzw. *problem wyroczni testowej*. Typowe czynniki przyczyniające się do problemu wyroczni testowej obejmują złożoność związaną z danymi, niedeterminizm (np. systemy oparte na sztucznej inteligencji), zachowanie probabilistyczne oraz brakujące lub niejednoznaczne wymagania.

Niektóre znane rozwiązania problemu wyroczni testowej opisano poniżej (Barr, 2014).

- Pseudowyrocznie to niezależnie opracowane systemy, które spełniają tę samą specyfikację co przedmiot testów (np. zastane systemy [ang. *legacy systems*], uproszczone wersje przedmiotów testów). Stworzenie dedykowanej pseudowyroczni dla złożonego przedmiotu testów może być kosztowne, ale jest typowe dla systemów krytycznych.
- Testowanie oparte na modelu może sformalizować wyrocznię testową jako część modelu testów. Umożliwia to automatyczne generowanie oczekiwanych wyników i wyprowadzanie testów z modelu. Techniki testowania oparte na zachowaniu często obejmują implementację zautomatyzowanej wyroczni testowej (patrz sekcje 3.2.2 i 3.2.3).
- Testowanie oparte na właściwościach (ang. *property-based testing*) wykorzystuje określone właściwości przedmiotu testów do weryfikacji relacji między wejściem a wynikiem oczekiwanym poszczególnych przypadków testowych. Jeśli taka relacja nie jest spełniona, przypadek testowy kończy się niezaliczeniem. Rozwiązanie to dobrze nadaje się do automatyzacji testów, ale jego skuteczność zależy od relacji, które mogą być trudne do określenia.
- Testowanie metamorficzne (patrz sekcja 3.3.2).
- Ludzkie wyrocznie (ang. *human oracles*) wykorzystują zdolność ludzi do określania oczekiwanych wyników. Zasoby ludzkie mogą być kosztowne i ograniczone. Wyrocznie ludzkie są jednak preferowane w niektórych podejściach testowych (np. w testach eksploracyjnych).
- Asercje mogą być wbudowane w kod testów automatycznych lub w sam przedmiot testów w celu zaimplementowania zautomatyzowanej wyroczni testowej. Są to wykonywalne instrukcje, które weryfikują stan lub zachowanie przedmiotu testów. W przypadku wbudowania w przedmiot testów, zazwyczaj weryfikują one tylko to, co jest niezbędne do kontynuowania zadania.

1.3.5. Wymagania dotyczące danych testowych

Podczas projektowania testów, AT identyfikuje i żąda danych testowych, które mogą wymagać przygotowania lub udostępnienia w celu wykonania testów. Rozważania obejmują cel, format i kontekst użycia (patrz też ISO/IEC/IEEE 29119-3, 2021, podrozdział 8.5). Kluczowe aspekty danych testowych to:

- **Podobieństwo do danych produkcyjnych**. Dane produkcyjne odzwierciedlają rzeczywiste dane, ale mogą być pozbawione zmienności. Dane syntetyczne pozwalają na kontrolowaną zmienność. Powinny one odzwierciedlać kluczowe aspekty wzorców danych produkcyjnych, rozkładów i wartości odstających, ale mogą pomijać pewne defekty, które występują tylko w przypadku danych rzeczywistych. W przypadku systemów, w których brakuje danych produkcyjnych, dane syntetyczne muszą odzwierciedlać realistyczne scenariusze biznesowe i techniczne. Persony (ang. *personas*) pomagają zapewnić realistyczne, skoncentrowane na użytkowniku profile, które kierują tworzeniem danych odzwierciedlających różne scenariusze i zachowania użytkowników.
- **Poufność**. Wrażliwe dane testowe (np. dane osobowe) wymagają ochrony. Dane pseudonimizowane zastępują dane osobowe sztucznymi identyfikatorami, natomiast dane zanonimizowane usuwają możliwe do zidentyfikowania informacje o osobach. W razie potrzeby AT musi przestrzegać przepisów o ochronie danych, takich jak RODO w Unii Europejskiej (GDPR, 2016) lub HIPAA w USA (HIPAA, 1996).
- **Cel**. Dane testowe mają kluczowe znaczenie dla określenia warunków wstępnych i oczekiwanych wyników, które wpływają na stan systemu i jego konfigurację (np. czas i datę systemową). Obejmuje to również określanie uprawnień użytkowników i ustanawianie relacji między produktami, działami i kategoriami.
- **Kryteria pokrycia**. Dane testowe muszą być zgodne z kryteriami pokrycia dla wybranej techniki testowania. Oprócz prawidłowych danych testowych może to również wymagać nieprawidłowych danych testowych na potrzeby testów negatywnych.
- **Format danych**. Systematyczne zarządzanie danymi (np. w testach API) może wymagać ustrukturyzowanych danych (np. CSV, JSON, XML lub relacyjna baza danych).
- **Śledzenie powiązań**. Śledzenie powiązań zapewnia utrzymanie danych testowych po wprowadzeniu zmian w przypadkach testowych.
- **Pielęgnowalność**. Należy unikać zakodowanych na stałe danych testowych w niskopoziomowych przypadkach testowych, aby ułatwić wykrywanie defektów i pielęgnację. Przypadki testowe powinny oddzielać logikę testową od danych testowych (patrz sekcja 1.3.2).
- **Zależności**. Dane zależne wymagają szeregu kroków, aby je utworzyć.
- **Dostępność**. Wirtualizacja usług może rozwiązać problem brakujących danych symulując nieobecne lub niedostępne usługi w celu interakcji z zewnętrznymi systemami lub usługami.
- **Wrażliwość czasowa i starzenie się danych**. Przypadki testowe obejmujące nieaktualne lub wrażliwe na czas dane mogą wpływać na zachowanie systemu w nieoczekiwany lub niedokładny sposób.

1.3.6. Tworzenie skryptów testowych za pomocą testowania opartego na słowach kluczowych

Jeśli stosowane jest testowanie oparte na słowach kluczowych, AT tworzy skrypty testowe przy użyciu słów kluczowych. Implementacja skryptów testowych jest zadaniem technicznego analityka testów, inżyniera automatyzacji testów lub programisty. AT identyfikuje i specyfikuje słowa kluczowe, analizując podstawę testów lub współpracując z odpowiednimi interesariuszami. Słowa kluczowe można podzielić na dwie kategorie: słowa akcji i słowa weryfikacji. Słowa akcji muszą wchodzić w interakcję z przedmiotem testów (np. wykonywanie funkcji, przesyłanie danych, nawigacja w obrębie przedmiotu testów), środowiskiem testowym (np. ustawianie konfiguracji i aktywowanie symulatorów) lub innymi modułami lub systemami (np. używanie interfejsu przedmiotu testów). Słowa weryfikacji reprezentują asercje w celu oceny, czy wynik rzeczywisty, uzyskany przez przedmiot testów, jest zgodny z wynikiem oczekiwanym.

Słowa kluczowe mogą znajdować się w co najmniej dwóch warstwach abstrakcji: warstwie dziedziny i warstwie interfejsu testowego. Słowa kluczowe warstwy dziedziny odpowiadają działaniom związanym z dziedziną biznesową i odzwierciedlają terminologię dziedziny aplikacji. Abstrahują one od szczegółów technicznych interfejsu przedmiotu testów. Słowa kluczowe warstwy interfejsu testowego komunikują się z przedmiotami testów, elementami środowiska testowego lub innymi modułami lub systemami za pośrednictwem ich interfejsów testowych. Słowa te znajdują się w najniższej warstwie abstrakcji. Dodatkowe warstwy pośrednie mogą wspierać utrzymywalność słów kluczowych.

Słowa kluczowe mogą być proste (atomowe) lub złożone z innych słów kluczowych. Struktura słowa kluczowego i warstwa abstrakcji są niezależnymi atrybutami, jednak złożone słowa kluczowe często znajdują się na wyższym poziomie abstrakcji, podczas gdy w warstwie interfejsu testowego zwykle znajdują się proste słowa kluczowe.

Zadania AT w testowaniu opartym na słowach kluczowych obejmują:

- specyfikowanie słów kluczowych i ich parametrów,
- określanie przypadków testowych opartych na słowach kluczowych (ang. *keyword test cases*), tj. skryptów testowych wykorzystujących słowa kluczowe,
- określanie dodatkowych kroków w skryptach testowych za pomocą słów kluczowych, takich jak warunki wstępne, działania weryfikacyjne i czyszczenie środowiska testowego po zakończeniu testu,
- utrzymywanie przypadków testowych opartych na słowach kluczowych w celu odzwierciedlenia zmian w przedmiocie testów,
- wykonywanie manualnych lub automatycznych skryptów testowych opartych na słowach kluczowych,
- analizowanie niezaliczonych przypadków testowych opartych na słowach kluczowych w celu określenia przyczyny niezaliczenia testu.

Analizując podstawę testów, AT szuka interakcji między przedmiotem testów a jego środowiskiem (np. użytkownikami, innymi systemami i urządzeniami). Rozważmy na przykład historyjkę użytkownika: „Jako członek chcę się uwierzytelnić, aby uzyskać dostęp do obiektów”, z kryterium akceptacji „Do uwierzytelnienia można użyć ważnych kart członkowskich”. AT może określić słowo kluczowe (akcji) „Uwierzytelnij członka” z parametrem „numer karty członkowskiej” i słowo

kluczowe (weryfikacji) „Zweryfikuj udzielenie dostępu". AT sprawdza, czy zidentyfikowane słowa kluczowe znajdują się w odpowiedniej warstwie abstrakcji.

Określając słowa kluczowe, AT musi pamiętać, że słowa kluczowe muszą:

- zawierać czasownik (+ rzeczownik),
- używać formy rozkazującej czasownika,
- być unikalne w swoim znaczeniu,
- być odpowiednio udokumentowane,
- odzwierciedlać słownictwo dziedziny aplikacji,
- być wielokrotnego użytku.

Podczas tworzenia przypadków testowych opartych na słowach kluczowych, AT może rozpoznać niektóre brakujące słowa kluczowe i natychmiast je wyspecyfikować. Przypadki testowe oparte na słowach kluczowych mogą być zapisywane w różnych formatach, takich jak listy lub tabele.

Słowa kluczowe mogą się zmieniać w trakcie trwania projektu i są podatne na redundancję (Rwemalika *et al.*, 2019). Zalecenia wymienione w tej sekcji mają na celu uniknięcie redundancji i zmniejszenie nakładów na pielęgnację.

Chociaż testowanie oparte na słowach kluczowych jest podejściem silnie związanym z automatyzacją testów, testowanie manualne również może z niego korzystać. Jeśli podejście to będzie zastosowane do testowania manualnego, będzie skutecznie wspierać późniejsze przejście z testowania manualnego na zautomatyzowane.

Więcej szczegółów na temat automatyzacji wykonywania testów i testowania opartego na słowach kluczowych można znaleźć w (ISTQB TTA, 2021), (ISTQB TAE, 2024) oraz (ISO/IEC/IEEE 29119-5, 2016).

1.3.7. Narzędzia do zarządzania testaliami

Właściwe zarządzanie testaliami za pomocą narzędzi może pomóc AT wspierać ogólny proces testowania. Narzędzia mogą zapewnić informację o statusie produktów pracy oraz wspierać monitorowanie i nadzór nad testami. W przypadku awarii w systemie podlegającym testowaniu, pozwalają one AT sprawdzić wyniki testów z poprzednich przebiegów testu i przeanalizować punkt w czasie, w którym wystąpiły defekty.

Niektóre kluczowe rodzaje narzędzi do zarządzania testaliami obejmują:

- **Narzędzia do zarządzania testami** – zapewniają dostęp do repozytorium wszystkich istotnych testaliów (np. warunków testowych, przypadków testowych, skryptów testowych, zestawów testowych i przebiegów testu). Narzędzia te ułatwiają śledzenie powiązań (np. za pomocą macierzy śledzenia), wyszukiwanie przypadków testowych, planowanie przebiegów testu, rejestrowanie wyników testów oraz ogólne raportowanie postępów i jakości testów.
- **Narzędzia do zarządzania defektami** – służą do rejestrowania, ustalania priorytetów i monitorowania procesu rozwiązywania defektów.
- **Narzędzia do zarządzania danymi testowymi** – służą do tworzenia i utrzymywania danych testowych, w tym ochrony danych wrażliwych (patrz sekcja 1.3.5).

- **Narzędzia do zarządzania konfiguracją** – ułatwiają działania związane z testami w ramach procesów implementacji, wydawania i utrzymywania oprogramowania, w tym zarządzanie konfiguracją i dostępnością środowisk testowych.
- **Narzędzia do zarządzania wymaganiami** – zawierają wymagania wysokiego poziomu, zapewniając, że są one jasno zdefiniowane, wersjonowane i śledzone w całym cyklu wytwarzania oprogramowania.

AT wspiera zarządzanie testaliami poprzez:

- analizę projektu i wydania w celu wyboru właściwego podzbioru testaliów (np. specyfikacja zidentyfikowana na podstawie wersji, aby pasowała do wersji systemu podlegającego testowaniu),
- zdefiniowanie odpowiedniej struktury funkcjonalnej (np. według cech lub modułów) lub technicznej (np. według typu testu lub środowiska) dla organizacji przypadków testowych w narzędziu do zarządzania testami,
- dodawanie metadanych do przypadków testowych (np. informacji o wysiłku związanym z wykonaniem testu lub wymaganym konkretnym środowisku testowym),
- zapewnienie śledzenia powiązań między wymaganiami, warunkami testowymi, testami, przebiegami testu i defektami,
- wybór odpowiednich zestawów testowych do testowania regresji (do manualnego lub automatycznego wykonywania testów),
- zarządzanie konfiguracją przypadków testowych, w tym identyfikację nieaktualnych przypadków testowych.

Narzędzia pomagają organizować, śledzić i zapewniać jakość procesu testowania. AT wspiera ten wysiłek wybierając, strukturyzując i utrzymując przypadki testowe, zapewniając śledzenie powiązań i zarządzając wersjami przypadków testowych w celu wydajnego testowania i kontroli testów.

2. Zadania analityka testów w testowaniu opartym na ryzyku (90 min.)

Słowa kluczowe

analiza ryzyka, analiza wpływu, identyfikacja ryzyka, kontrola ryzyka, łagodzenie ryzyka, monitorowanie ryzyka, ocena ryzyka, ryzyko produktowe, testowanie oparte na ryzyku, testowanie regresji

Cele nauczania dla rozdziału 2

2.1 Analiza ryzyka

TA-2.1.1 (K2) Podsumować wkład analityka testów w analizę ryzyka produktowego.

2.2 Kontrola ryzyka

TA-2.2.1 (K4) Przeanalizować wpływ zmian w celu określenia zakresu testów regresji.

Wstęp

Testowanie oparte na ryzyku to podejście do testowania, w którym priorytetyzuje się wysiłek testowy w oparciu o poziomy ryzyka elementów testowych. Podejście to określa kierownik testów. AT odgrywa aktywną rolę we wdrażaniu tego podejścia. Testowanie oparte na ryzyku omówiono bardziej szczegółowo w sylabusie (ISTQB TM, 2024).

Testowanie oparte na ryzyku obejmuje analizę ryzyka i kontrolę ryzyka (ISTQB CTFL, podrozdział 5.2).

Ponadto, ponieważ ryzyka i ich poziomy nie są statyczne i zmieniają się w czasie, testowanie oparte na ryzyku obejmuje regularne monitorowanie ryzyka. W iteracyjnym cyklu wytwarzania jest to wykonywane z częstotliwością określoną przez zespół (zazwyczaj raz na iterację). W innych cyklach wytwarzania częstotliwość ta jest ustalana przez osobę odpowiedzialną za zarządzanie ryzykiem produktowym, często kierownika testów. AT wnosi swój wkład poprzez aktualizację rejestru ryzyk w oparciu o wprowadzone zmiany i dostosowanie wyżej wymienionych działań łagodzących ryzyko.

2.1. Analiza ryzyka

Zgodnie z (ISTQB CTFL, podrozdział 5.2) analiza ryzyka obejmuje identyfikację ryzyka i ocenę ryzyka. Niniejszy sylabus koncentruje się na tym, w jaki sposób AT powinien uczestniczyć w działaniach związanych z analizą ryzyka, aby zapewnić prawidłowe wdrożenie testowania opartego na ryzyku. AT często posiada unikalną wiedzę na temat systemu, a także doświadczenie i intuicyjną wiedzę na temat tego, co zwykle może pójść nie tak podczas tworzenia systemu, jaki ma to wpływ i jak testowanie może złagodzić ryzyko. Sprawia to, że AT jest cennym interesariuszem w analizie ryzyka produktowego.

Podczas **identyfikacji ryzyka**, AT wnosi własne doświadczenie i wiedzę do retrospektyw, warsztatów ryzyka, burzy mózgów i tworzenia list kontrolnych. AT może również przeprowadzać wywiady z interesariuszami, aby lepiej zrozumieć, co uważają za najważniejsze ryzyka z ich perspektywy.

Podczas **oceny ryzyka** AT wraz z innymi interesariuszami określa poziom ryzyka szacując czynniki ryzyka takie jak:

- częstotliwość użytkowania oraz krytyczność związanych z ryzykiem funkcjonalności,
- krytyczność celów biznesowych, których dotyczy ryzyko,
- szkody finansowe, środowiskowe i wizerunkowe,
- jakość podstawy testów,
- wymogi prawne lub potrzeby związane z bezpieczeństwem.

AT pomaga również kategoryzować ryzyka produktowe według charakterystyk jakościowych, na przykład przy użyciu modelu jakości produktu (ISO/IEC 25010, 2023). Poziom ryzyka w przedmiocie testów często nie jest rozłożony równomiernie. W takich przypadkach AT powinien podzielić przedmiot testów na elementy testowe (np. moduły, interfejsy i funkcje) i ocenić dane ryzyko dla każdego elementu testowego osobno.

Wreszcie, w ramach oceny ryzyka produktowego, AT proponuje odpowiednie działania testowe w celu złagodzenia każdego zidentyfikowanego ryzyka produktowego. Działania te mogą

obejmować testowanie statyczne i dynamiczne. W zależności od czynników, takich jak poziom ryzyka, powiązany typ elementu testowego czy charakterystyka jakościowa, której dotyczy ryzyko, AT wskazuje wymagane poziomy testów, typy testów, techniki testowania, poziomy niezależności testowania i dokładność testów. Stosując przesunięcie w lewo (ang. *shift left*), AT wskazuje, które działania testowe mogą złagodzić ryzyko najwcześniej, aby zminimalizować wysiłek testowy.

2.2. Kontrola ryzyka

Zgodnie z (ISTQB CTFL, sekcja 5.2.4) kontrola ryzyka obejmuje łagodzenie ryzyka i monitorowanie ryzyka. AT rozumie funkcjonalności systemu i potencjalne ryzyka z nimi związane. W związku z tym rola AT jest kluczowa w kilku działaniach łagodzących ryzyko wymienionych w (ISTQB CTFL, sekcja 5.2.4):

- wykonywanie przeglądów (omówiono w sekcji 5.2.2),
- zastosowanie odpowiednich technik testowania i poziomów pokrycia (omówiono w podrozdziale 3.5),
- zastosowanie odpowiednich typów testów (omówiono w rozdziale 4),
- przeprowadzanie testów regresji (omówiono poniżej).

Ponadto, ponieważ ryzyka i ich poziomy nie są statyczne, lecz zmieniają się w czasie, testowanie oparte na ryzyku obejmuje regularne monitorowanie ryzyka. W iteracyjnym modelu wytwarzania monitorowanie ryzyka jest wykonywane z częstotliwością określoną przez zespół (zazwyczaj raz na iterację). W innych modelach wytwarzania oprogramowania częstotliwość jest ustalana przez osobę odpowiedzialną za zarządzanie ryzykiem produktowym, często kierownika testów. AT wnosi swój wkład poprzez aktualizację rejestru ryzyk w oparciu o wprowadzone zmiany i dostosowanie działań ograniczających ryzyko, o których mowa powyżej.

Głównym celem testowania regresji jest zapewnienie pewności co do jakości przedmiotu testów po wprowadzeniu w nim zmiany. Przeprowadzenie wszystkich testów regresji podczas testowania regresji może być jednak niemożliwe ze względu na ograniczenia (np. czasowe, budżetowe, dotyczące środowiska testowego lub danych testowych). Dotyczy to przede wszystkim testów wykonywanych manualnie, ale także testów automatycznych, na przykład, gdy cykle testowe są krótkie i istnieje wiele długotrwałych testów automatycznych. Powoduje to konieczność wyboru odpowiednich testów regresji w oparciu o określone kryteria. Niezbędne jest dokonanie przeglądu zakresu testów regresji przy każdym cyklu testowym. W wyniku przeglądu może okazać się, że istniejący zestaw testów regresji wymaga korekty.

Najbardziej niezawodną techniką wyboru testów automatycznych jest analiza wpływu, która może być wspierana narzędziami. Narzędzia te oparte są na zautomatyzowanym systemie zarządzania konfiguracją. Rejestrują one, które elementy konfiguracji są aktywowane podczas wykonywania każdego przypadku testowego. Gdy następuje zmiana, śledzą one, które elementy konfiguracji zostały zmodyfikowane i wybierają testy regresji, które wchodzą w interakcję ze zmienionymi elementami. W ten sposób narzędzia zapewniają, że po zmianie elementu konfiguracji testy koncentrują się na tych obszarach, w których najprawdopodobniej znajdą awarie (Juergens *et al.*, 2018).

Jeśli chodzi o manualne wykonywanie testów, nie ma jednoznacznych dowodów na to, że jakaś technika wyboru testów regresji jest wyraźnie lepsza od innych, ponieważ efektywność zależy

od różnych czynników (Engström *et al.*, 2010). Dlatego AT dokonuje wyboru techniki w oparciu o konkretną sytuację. Powszechnie stosowane techniki obejmują:

- **Wybór oparty na ryzyku**, w ramach którego AT utrzymuje śledzenie powiązań zestawu testów regresji z rejestrem ryzyk. Gdy wprowadzana jest zmiana, a rejestr ryzyk jest odpowiednio aktualizowany, AT dostosowuje zestaw testów regresji, aby pokryć ryzyka o najwyższym poziomie.
- **Testowanie oparte na historii**, w którym AT ocenia wcześniejsze wykonania testów i określa, które z nich ujawniły defekty lub były wrażliwe na zmiany podobne do tych wprowadzonych od czasu ostatniego wykonania testu. Ponowne wykonanie odpowiednich testów w ramach testów regresji zwiększa prawdopodobieństwo ujawnienia podobnych defektów. AT może również włączyć do zestawu testów regresji niektóre testy, które nie były wykonywane przez długi czas, aby upewnić się, że nadal są one zaliczane.
- **Testowanie oparte na pokryciu**, w którym AT wybiera pewną liczbę testów, które łącznie osiągają jak największe pokrycie w oparciu o wybraną technikę (techniki) testowania. Liczba testów musi być starannie zrównoważona wraz ze wzrostem pokrycia, jaki daje każdy kolejny test.
- **Macierz śledzenia wymagań**, która służy do oceny wpływu zmian wymagań na powiązane testy. Może być szczególnie cenna, gdy nowe lub zmienione wymagania pośrednio wpływają na istniejące funkcje. AT wybiera testy regresji dla bezpośrednio dotkniętych i powiązanych funkcji, aby pokryć możliwe niezamierzone skutki uboczne. W zwinnym tworzeniu oprogramowania można to zrobić podobnie, wybierając testy, które obejmują kryteria akceptacji, na które mają wpływ nowe lub zmienione historyjki użytkownika.
- **Testowanie oparte na profilu operacyjnym**, gdzie AT wybiera przypadki testowe do wykonania testów regresji na podstawie wzorców użycia przedmiotu testów. Na przykład, podczas testowania sklepu internetowego, jeden z takich testów może obejmować logowanie użytkownika, wyszukiwanie produktów, dodawanie ich do koszyka i składanie zamówienia. Gdy aplikacja zostanie znacząco zmieniona, technika ta zapewnia szybki przegląd ogólnej funkcjonalności systemu. Jeśli zmiana skutkuje zbyt dużą liczbą testów, AT nadaje priorytet krytycznym wzorcom użycia (np. tym, które występują często lub obejmują krytyczne funkcje i procesy biznesowe).
- **Analiza wpływu**, która może być również wykorzystana do wyboru przypadków testowych regresji do wykonania manualnego, jeśli AT wie, które z nich wchodzą w interakcję ze zmienionymi elementami konfiguracji.

Często konieczne jest zastosowanie kombinacji technik wyboru testów w celu uzyskania bardziej kompleksowego i skutecznego zestawu testów regresji. AT musi jednak starannie zrównoważyć potrzebę pokrycia z możliwym do zarządzania rozmiarem zestawu testowego Po każdym cyklu testowym AT analizuje wyniki testów w celu określenia skuteczności zastosowanych technik (patrz sekcja 5.3.1). Przy każdym kolejnym wyborze testów regresji AT zachowuje skuteczne techniki i zastępuje nieskuteczne. Pozwala to na ciągłe ulepszanie wyboru testów regresji. Jest to niezbędne w iteracyjnych i przyrostowych modelach wytwarzania, w których zmiany są częste.

3. Analiza i projektowanie testów (615 min.)

Słowa kluczowe

karta opisu testu, klasa równoważności, relacja metamorficzna, testowanie CRUD, testowanie dziedziny, testowanie kombinatoryczne, testowanie losowe, testowanie metamorficzne, testowanie przejść pomiędzy stanami, testowanie w oparciu o doświadczenie, testowanie w oparciu o listę kontrolną, testowanie w oparciu o scenariusze, testowanie w oparciu o tablicę decyzyjną, testowanie w sesjach, testowanie w tłumie

Cele nauczania dla rozdziału 3

3.1 Techniki testowania oparte na danych

TA-3.1.1 (K3) Zastosować testowanie dziedziny.

TA-3.1.2 (K3) Zastosować testowanie kombinatoryczne.

TA-3.1.3 (K2) Podsumować korzyści i ograniczenia testowania losowego.

3.2 Techniki testowania oparte na zachowaniu

TA-3.2.1 (K2) Wyjaśnić testowanie CRUD.

TA-3.2.2 (K3) Zastosować testowanie przejść pomiędzy stanami.

TA-3.2.3 (K3) Zastosować testowanie oparte na scenariuszach.

3.3 Techniki testowania oparte na regułach

TA-3.3.1 (K3) Zastosować testowanie w oparciu o tablice decyzyjne.

TA-3.3.2 (K3) Zastosować testowanie metamorficzne.

3.4 Techniki testowania oparte na doświadczeniu

TA-3.4.1 (K3) Przygotować karty opisu testów dla testowania w sesjach.

TA-3.4.2 (K3) Przygotować listy kontrolne wspierające testowanie oparte na doświadczeniu.

TA-3.4.3 (K2) Podać przykłady korzyści i ograniczeń testowania w tłumie.

3.5 Zastosowanie najlepszych technik testowania

TA-3.5.1 (K4) Wybrać właściwe techniki testowania w celu łagodzenia ryzyk produktowych w danej sytuacji.

TA-3.5.2 (K2) Wyjaśnić korzyści i ryzyka związane z automatyzacją projektowania testów.

Wstęp

Techniki testowania są wykorzystywane głównie w analizie i projektowaniu testów. Techniki omówione w tym rozdziale obejmują czarnoskrzynkowe techniki testowania i techniki testowania oparte na doświadczeniu. Białoskrzynkowe techniki testowania są omówione w sylabusie (ISTQB TTA, 2021).

Niniejszy sylabus dzieli czarnoskrzynkowe techniki testowania na trzy kategorie w oparciu o warunki testowe, które modelują:

- elementy danych (techniki oparte na danych),
- elementy dynamicznego zachowania (techniki oparte na zachowaniu),
- elementy statycznych reguł zachowania (techniki oparte na regułach).

3.1. Techniki testowania oparte na danych

W tym podrozdziale termin „dziedzina" jest używany do reprezentowania zestawu danych wejściowych elementu testowego. Dane wejściowe z różnych obszarów dziedziny prowadzą do różnych zachowań elementu testowego. Techniki testowania oparte na danych mają na celu sprawdzenie, czy implementacja poprawnie obsługuje określone obszary dziedziny. Sylabus Certyfikowany Tester Poziom Podstawowy (ISTQB CTFL, podrozdział 4.2) opisuje dwie techniki testowania, które można sklasyfikować jako oparte na danych: podział na klasy równoważności i analizę wartości brzegowych. Niniejszy sylabus wprowadza trzy kolejne techniki testowania oparte na danych:

- testowanie dziedziny – uogólnia podział na klasy równoważności i analizę wartości brzegowych na dziedziny opisane wieloma zmiennymi i dziedziny wielowymiarowe,
- testowanie kombinatoryczne – koncentruje się na interakcjach wielu parametrów w wielowymiarowej dziedzinie,
- testowanie losowe – wybiera losowe dane wejściowe z dziedziny na podstawie określonego rozkładu prawdopodobieństwa.

Testowanie losowe może odnosić się zarówno do losowych danych wejściowych, jak i losowych zdarzeń. Niniejszy sylabus dotyczy wyłącznie testowania z losowymi danymi wejściowymi.

3.1.1. Testowanie dziedziny

Testowanie dziedziny weryfikuje, czy element testowy zachowuje się poprawnie w klasach równoważności dziedziny i na ich brzegach. Klasy równoważności są definiowane przy użyciu wyrażeń, które łączą warunki atomowe za pomocą operatorów logicznych (np. AND, OR i NOT) i obejmują jedną lub więcej zmiennych. Każdy warunek atomowy definiuje brzeg klasy równoważności. Brzegi domknięte są tworzone przy użyciu wyrażeń relacyjnych z operatorami ≤, ≥ lub =, a brzegi otwarte – przy użyciu wyrażeń relacyjnych z operatorami <, > lub ≠.

Na przykład wyrażenie „wzrost > 1,29 AND waga / wzrost^2 ≥ 30" definiuje klasę równoważności dwuwymiarowej dziedziny dwóch zmiennych o dwóch brzegach: jednym otwartym, a drugim domkniętym.

Testowanie dziedziny ma na celu zidentyfikowanie defektów w implementacji klas równoważności (np. użycie niewłaściwego operatora lub stałej) poprzez wybranie odpowiednich

elementów pokrycia, które mogą ujawnić te defekty. Kryteria pokrycia w testowaniu dziedziny odnoszą się do tzw. punktów ON, OFF, IN i OUT:

- W przypadku brzegu domkniętego punkt ON leży na tym brzegu. W przypadku brzegu otwartego punkt ON leży wewnątrz klasy równoważności, najbliżej brzegu, zgodnie z podaną dokładnością.
- W przypadku brzegu domkniętego punkt OFF leży poza klasą równoważności, najbliżej brzegu, zgodnie z podaną dokładnością. W przypadku brzegu otwartego punkt OFF leży na tym brzegu.
- Punkt IN należy do klasy równoważności i nie jest punktem ON.
- Punkt OUT leży poza klasą równoważności i nie jest punktem OFF.

Każdy z tych typów punktów jest powiązany z brzegiem określonej klasy równoważności. Ten sam punkt może być różnego typu dla różnych brzegów.

Kryteria pokrycia obejmują:

Pokrycie uproszczone (ang. *simplified coverage*), które wymaga następujących elementów pokrycia (Jeng *et al.*, 1994):

- Jeden punkt ON i jeden punkt OFF dla każdego brzegu zdefiniowanego operatorem <, ≤, > lub ≥.
- Jeden punkt ON i dwa punkty OFF leżące po różnych stronach brzegu zdefiniowanego przez operator =.
- Jeden punkt OFF i dwa punkty ON leżące po różnych stronach brzegu zdefiniowanego przez operator ≠.

Każdy punkt OFF powinien znajdować się jak najbliżej odpowiadającego mu punktu ON zgodnie z podaną dokładnością.

Pokrycie niezawodne (ang. *reliable coverage*), które wymaga następujących elementów pokrycia (Forgács *et al.*, 2024):

- Po jednym punkcie ON, OFF, IN i OUT dla każdego brzegu zdefiniowanego przez operator <, ≤, > lub ≥.
- Jeden punkt ON i dwa punkty OFF leżące po różnych stronach brzegu zdefiniowanego przez operator =.
- Jeden punkt OFF i dwa punkty ON leżące po różnych stronach brzegu zdefiniowanego przez operator ≠.

W przypadku obu kryteriów pokrycia liczba elementów pokrycia zależy liniowo od liczby brzegów. Liczbę tę można zoptymalizować dla obu kryteriów pokrycia, używając tego samego elementu pokrycia dla różnych brzegów. Na przykład wszystkie brzegi klasy równoważności mogą używać wspólnego punktu IN, a para punktów ON i OFF dla pewnego brzegu może być użyta jako punkty OFF i ON dla brzegu sąsiedniej klasy równoważności. Dla dziedzin z wieloma brzegami istnieje zaawansowany algorytm optymalizacji liczby punktów (Forgács *et al.*, 2024).

Pokrycie niezawodne wymaga nieco większej liczby elementów pokrycia niż pokrycie uproszczone, ale może wykryć znacznie więcej defektów dziedziny (SSTD, 2020).

Testowanie dziedziny może być stosowane na dowolnym poziomie testów. Uogólnia ono podział na klasy równoważności i analizę wartości brzegowych (ISTQB CTFL, podrozdział 4.2) na bardziej złożone dziedziny. Różne podejścia do testowania dziedziny można znaleźć w podręcznikach takich jak (Beizer, 1990, rozdz. 6), (Binder, 2000, sekcja 10.2.4), (Kaner *et al.*, 2013), (Jorgensen, 2014, rozdz. 5).

3.1.2. Testowanie kombinatoryczne

Niektóre awarie oprogramowania znane jako awarie interakcji, wynikają z określonych kombinacji wartości parametrów. Testowanie kombinatoryczne ma na celu ich ujawnienie poprzez zbadanie kombinacji wartości parametrów. W testowaniu kombinatorycznym warunki testowe zazwyczaj łączą parametry konfiguracji lub wartości wejściowe. W związku z tym istnieją dwa główne podejścia do testowania kombinatorycznego. Pierwsze wykorzystuje kombinacje parametrów konfiguracji, a każda kombinacja może być testowana przy użyciu tego samego przypadku testowego. Drugie podejście wykorzystuje kombinacje wartości danych wejściowych, które następnie stają się częścią kompletnych przypadków testowych, tworząc zestaw testowy dla testowanego systemu (Kuhn *et al.*, 2013).

Konkretna para parametr-wartość składa się z parametru i jego wartości (np. „(kolor, czerwony)”).

Kombinatoryczne kryteria pokrycia obejmują (Ammann *et al.*, 2008; Forgács *et al.*, 2019):

- **Pokrycie wyboru bazowego,** które zakłada, że niektóre pary parametr-wartość są ważniejsze od innych. Dla każdego parametru wybierana jest bazowa para parametr-wartość, a bazowy element pokrycia jest kombinacją bazowych par parametr-wartość. Kolejne elementy pokrycia są tworzone z bazowego elementu pokrycia poprzez zastąpienie jednej wartości wartością niebazową na wszystkie możliwe sposoby (tzn. dla każdej wartości każdego z parametrów).
- **Pokrycie sposobem par**, w którym elementy pokrycia są wszystkimi możliwymi parami par parametr-wartość dla dowolnych dwóch różnych parametrów. Dostępne są narzędzia do generowania elementów pokrycia. Jednak znalezienie *minimalnego* zestawu przypadków testowych zapewniających pokrycie sposobem par jest w ogólności problemem trudnym obliczeniowo.

W przypadku parametrów o wielu wartościach można najpierw zastosować podział na klasy równoważności, aby zmniejszyć liczbę możliwych wartości i zestaw wynikowych kombinacji. Reprezentowanie parametrów i ich wartości w drzewie klasyfikacji lub modelu cech (patrz IREB-CPRE) wspiera tę czynność. Na ostateczną liczbę przypadków testowych mogą mieć wpływ ograniczenia między parami parametr-wartość, ręcznie dodane kombinacje, o których wiadomo, że są problematyczne, a także nieprawidłowe lub niewykonalne (nietestowalne) kombinacje.

Kluczowe dla testowania kombinatorycznego jest to, że nie każdy parametr przyczynia się do awarii i że większość awarii jest wywoływana przez pojedynczą wartość parametru lub interakcje między stosunkowo niewielką liczbą parametrów (Cohen *et al.*, 1994). Jest to zgodne z hipotezą „efektu sprzężenia”, wskazującą, że wykrywanie prostych defektów w programie może często ujawniać również defekty złożone (Offutt, 1992). W ograniczonym eksperymencie (Kuhn *et al.*, 2004) wyniki wykazały, że około 97% awarii jest spowodowanych tylko jednym lub dwoma współdziałającymi warunkami, co wskazuje, że testowanie sposobem par jest skuteczną techniką testowania.

Więcej informacji na temat testowania kombinatorycznego, w tym innych kryteriów pokrycia, takich jak *diff-pair-t* lub *n-wise testing*, można znaleźć w (Ammann *et al*. 2008; Forgács *et al*., 2019). Opis narzędzi wspierających testowanie kombinatoryczne można znaleźć na stronie (Czerwonka, 2004).

3.1.3. Testowanie losowe

Testowanie losowe polega na losowym wybieraniu danych testowych z dziedziny wejściowej elementu testowego w oparciu o określony rozkład prawdopodobieństwa. Do celów walidacji zalecany jest rozkład oparty na profilach operacyjnych. Do celów weryfikacji rozkład powinien być niezależny od użycia, aby uniknąć uprzedzeń. Wyniki oczekiwane mogą być dodawane do przypadków testowych przy użyciu wyroczni testowej, co zwykle wymaga, by była ona zautomatyzowana.

Testowanie losowe może być kierowane lub niekierowane. W niekierowanym testowaniu losowym rozkład prawdopodobieństwa pozostaje stały przez cały proces. W kierowanym testowaniu losowym, którego przykładem są techniki takie jak adaptacyjne testowanie losowe (Huang *et al*., 2019), rozkład prawdopodobieństwa zmienia się na podstawie wcześniej wylosowanych wartości, ewoluując w czasie. Kierowane testowanie losowe ma na celu skuteczne pokrycie dziedziny wejściowej, biorąc pod uwagę, że defekty często gromadzą się w określonych regionach dziedziny.

Testowanie losowe nie posiada uznanych powszechnie kryteriów pokrycia. Kryteria wyjścia mogą zatem opierać się wyłącznie na liczbie wykonanych testów, czasie testowania lub podobnych miarach ukończenia.

Testowanie losowe jest szczególnie przydatne, gdy wiedza na temat dziedziny jest ograniczona lub gdy istnieje potrzeba uzyskania dużej ilości danych testowych. Jest ono opłacalne i zapewnia, w ujęciu probabilistycznym, wgląd w niezawodność przedmiotu testów. Testowanie losowe pomaga uniknąć błędów, takich jak przeoczenie defektów w testach manualnych z powodu niewłaściwego zaufania do kodu lub funkcjonalności. Testowanie losowe wiąże się jednak również z wyzwaniami i ograniczeniami, w tym zaniedbywaniem semantyki danych, potencjalnym pomijaniem defektów związanych ze znaczeniem danych, przeoczaniem niektórych defektów, generowaniem redundantnych testów, zależnością od zautomatyzowanej wyroczni testowej oraz losowymi danymi wyjściowymi prowadzącymi do niespójnych wyników testów. Zrównoważenie zalet i ograniczeń testowania losowego jest niezbędne w każdym kontekście testowania.

Tradycyjnie testowanie losowe jest uważane za mniej skuteczne niż inne techniki testowania. W ostatnich latach hipoteza ta została zbadana w wielu badaniach empirycznych. Stwierdzono, że w wyżej wymienionych okolicznościach testowanie losowe może być bardziej skuteczne i wydajne niż inne techniki testowania oparte na danych (Arcuri *et al*., 2012; Wu *et al*., 2020). Testowanie losowe jest również stosowane w testowaniu *fuzz* (testowaniu przez losowe dane testowe) i inżynierii chaosu.

3.2. Techniki testowania oparte na zachowaniu

W technikach testowania opartych na zachowaniu przypadki testowe wyprowadza się na podstawie specyfikacji dynamicznego (tj. zależnego od stanu) zachowania elementu testowego. Niniejszy sylabus omawia trzy techniki testowania oparte na zachowaniu:

- testowanie CRUD,
- testowanie przejść pomiędzy stanami,
- testowanie oparte na scenariuszach.

Testowanie CRUD i testowanie oparte na scenariuszach rozszerzają katalog technik testowania czarnoskrzynkowego znanych z (ISTQB CTFL, podrozdział 4.2), a w przypadku testowania przejść pomiędzy stanami – uzupełnia opis techniki o dodatkowe kryteria pokrycia.

3.2.1. Testowanie CRUD

Testowanie CRUD weryfikuje cykl życia jednostek danych (zwanych encjami) przetwarzanych przez element testowy. Akronim CRUD oznacza *tworzenie* (ang. *Create)*, *odczyt* (ang. *Read*), *aktualizację* (ang. *Update*) i *usuwanie* (ang. *Delete*). Są to cztery podstawowe operacje, które funkcje mogą wykonywać na encjach.

Macierz CRUD opisuje cykl życia encji. Jej kolumny reprezentują encje, a wiersze reprezentują funkcje. Załóżmy, że funkcja wykonuje jedną lub więcej ww. operacji na danej encji. Jest to pokazane w macierzy za pomocą inicjałów tych operacji: C, R, U lub D. Aby utworzyć macierz CRUD, AT określa dla każdej funkcji, które z czterech operacji wykonuje na których encjach. Szczególną uwagę należy zwrócić na operację odczytu, często domyślnie powiązaną z operacjami C-U-D.

Testowanie CRUD posiada dwie odmiany (Koomen *et al.*, 2006):

- **Testowanie kompletności** to testowanie statyczne, które sprawdza, czy wszystkie możliwe operacje (tj. C, R, U i D) występują z każdą encją (tj. czy cały cykl życia został zaimplementowany dla każdej encji). Brak operacji jest anomalią, która wymaga zbadania.
- **Testowanie spójności** to testowanie dynamiczne mające na celu integrację różnych funkcji i sprawdzenie, czy encja jest używana w sposób spójny. Weryfikuje, czy funkcje współdziałają poprawnie podczas obsługi encji. Przypadki testowe powinny obejmować wszystkie operacje w macierzy CRUD. Należy również uwzględnić testy negatywne (np. próbę odczytania encji, która nie została jeszcze utworzona). Przypadki testowe są projektowane dla każdej encji poprzez łączenie funkcji tak, aby pokryć cały cykl życia tej encji.

Pokrycie CRUD jest mierzone liczbą operacji wykonanych przez zestaw testowy podzieloną przez całkowitą liczbę operacji w macierzy CRUD. Bardziej rygorystyczne pokrycie CRUD może uwzględniać określone kombinacje operacji jako elementy pokrycia (np. po każdym U należy wykonać wszystkie możliwe R).

Testowanie CRUD jest stosowane głównie na poziomie testowania systemowego. Koncentruje się na defektach związanych z zachowaniem elementu testowego podczas przetwarzania encji, takich jak naruszenia integralności danych, defekty kontroli dostępu lub niespójności danych.

3.2.2. Testowanie przejść pomiędzy stanami

Wiele złożonych systemów ma charakter stanowy, tj. reakcja systemu na zdarzenie zależy od aktualnego stanu systemu. Przykładami systemów stanowych są systemy wbudowane, systemy dialogowe, systemy kontroli lub systemy zajmujące się cyklami życia różnych encji.

Stan jest uproszczoną reprezentacją złożonych wewnętrznych szczegółów systemu, co ułatwia interesariuszom zrozumienie oczekiwanego zachowania. Podczas analizy testów, AT musi upewnić się, że model stanowy reprezentuje oczekiwane zachowanie elementu testowego na poziomie szczegółowości potrzebnym do testowania. Jeśli podstawa testu zawiera już taki model, AT musi sprawdzić, czy zawiera on warunki testowe i, jeśli to konieczne, dostosować go lub zaprojektować nowy.

W modelach stanowych wierzchołki reprezentują stany, a krawędzie – przejścia pomiędzy stanami. Przejścia są wyzwalane przez zdarzenia i mogą obejmować warunki dozoru i akcje (patrz ISTQB CTFL, sekcja 4.2.4). W użyciu jest kilka wariantów tych modeli, takich jak rozszerzone skończone maszyny stanów (Bochmann *et al.*, 1994), diagramy stanów Harela (Harel, 1987) czy diagram maszyny stanów UML (OMG UML, 2017).

Oprócz kryteriów pokrycia omówionych w (ISTQB CTFL), poniżej omówiono dwa dodatkowe kryteria, dla których empirycznie udowodniono wysoką skuteczność wykrywania defektów:

- **Pokrycie N-przełączeń** (ang. *N-switch coverage*) stosuje się do poprawnych sekwencji N+1 kolejnych przejść, zwanych N-przełączeniami (Chow, 1978). Pokrycie 0-przełączeń jest równe pokryciu poprawnych przejść (patrz ISTQB CTFL, sekcja 4.2.4). 1-przełączenie to para przejść wchodzącego do i wychodzącego z danego stanu. Rozszerzenie N-przełączeń na ich końcu o kolejne przejście poprawne daje w wyniku (N+1)-przełączenie. Pokrycie 0- i 1-przełączeń jest często stosowane w praktyce. Pełne pokrycie N-przełączeń dla N ≥ 2 jest wskazane tylko w przypadku wysokiego ryzyka awarii z powodu nieoczekiwanych sekwencji zdarzeń, ponieważ liczba N-przełączeń może rosnąć wykładniczo wraz ze wzrostem N.
- **Pokrycie pętli** (ang. *round-trip coverage*) (Ammann *et al.*, 2008). Pętla to ścieżka, która zaczyna się i kończy w tym samym stanie, a żaden inny stan na tej ścieżce nie występuje dwukrotnie. Elementami pokrycia są pętle. W (Antoniol *et al.*, 2002) uznano to kryterium za wysoce skuteczne w wykrywaniu defektów.

Testowanie przejść pomiędzy stanami dobrze nadaje się do automatyzacji przy użyciu narzędzi do testowania opartego na modelu. Podobnie jak w przypadku większości czarnoskrzynkowych technik testowania, testowanie przejść pomiędzy stanami przyczynia się do zapobiegania defektom poprzez wykrywanie problemów w specyfikacji podczas modelowania. Testowanie przejść pomiędzy stanami może być stosowane na dowolnym poziomie testów.

Więcej kryteriów pokrycia przejść pomiędzy stanami można znaleźć w (Rechtberger *et al.*, 2022), a podejście wykorzystujące tzw. model stanów akcji omówiono w (Forgács *et al.*, 2024).

3.2.3. Testowanie oparte na scenariuszach

Testowanie oparte na scenariuszach ocenia zachowanie elementu testowego w realistycznych scenariuszach. Techniki takie jak badania użytkowników (ang. *user research*), historyjki użytkownika, mapy podróży użytkowników (ang. *user journey maps*) i persony (patrz rozdział 11) mogą pomóc w identyfikacji wartościowych scenariuszy. W testowaniu opartym na scenariuszach AT tworzy model scenariusza oparty na sekwencjach działań, które stanowią przepływy pracy przez element testowy (por. ISO/IEC/IEEE 29119-4, 2021). Niniejszy sylabus omawia dwa takie modele: diagramy aktywności i przypadki użycia. Inne modele obejmują schematy blokowe (ang. *flowcharts*), notację i modele procesów biznesowych (OMG BPMN,

2013), diagramy sekwencji lub diagramy współpracy (OMG UML, 2017). Modele te nie są omawiane w niniejszym sylabusie. Więcej szczegółów można znaleźć w sylabusie (ISTQB AcT, 2019).

Diagram aktywności to graficzna reprezentacja przepływu pracy w systemie. Diagramy aktywności są szczególnie przydatne do modelowania procesów biznesowych, ale mogą również modelować przepływ sterowania. Diagramy aktywności rozszerzają notację schematów blokowych, umożliwiając modelowanie współbieżności. Głównymi elementami diagramów aktywności są: wierzchołki początkowy i końcowy, akcje, przejścia, wierzchołki decyzyjne, scalające (ang. *merge nodes*), rozwidlające (ang. *fork nodes*), łączące (ang. *join nodes*) oraz pasy aktywności (ang. *swim lanes*).

Przypadek użycia to tekstowy lub graficzny opis działań reprezentujących interakcje między użytkownikiem a systemem lub między systemami. W modelu tym wyróżnia się trzy rodzaje scenariuszy:

- **Scenariusz główny** (tzw. „szczęśliwa ścieżka", ang. *happy path*) – typowa, oczekiwana sekwencja działań prowadząca do osiągnięcia założonego celu z perspektywy użytkownika. Przypadek użycia posiada dokładnie jeden scenariusz główny.
- **Scenariusz alternatywny** (lub rozszerzenie) – sekwencja działań innych niż scenariusz główny, która ostatecznie prowadzi do osiągnięcia celu scenariusza głównego.
- **Wyjątek** – sekwencja działań, która nie pozwala na osiągnięcie celu głównego scenariusza z powodu nieoczekiwanego działania (np. nieprawidłowego użycia lub nieprawidłowych danych wejściowych).

W przypadku testowania opartego na scenariuszach AT projektuje przypadki testowe obejmujące scenariusze (tj. elementy pokrycia), często zgodnie z priorytetyzacją opartą na ryzyku. Gdy model scenariusza nie zawiera pętli, każdy scenariusz można przetestować za pomocą osobnego przypadku testowego (tj. wszystkie możliwe scenariusze w modelu można przetestować za pomocą skończonego zestawu testów). W przypadku istnienia pętli liczba potencjalnych scenariuszy (ścieżek) może być nieskończona. W takim przypadku do modelu scenariusza można zastosować tzw. proste pokrycie pętli. Wymaga ono, aby każdą pętlę wykonać zero razy (tj. pominąć ją), dokładnie raz, więcej niż raz (tj. typową liczbę iteracji) i maksymalną możliwą liczbą razy (jeśli to możliwe).

Pokrycie oparte na scenariuszach jest mierzone jako liczba wykonanych scenariuszy podzielona przez liczbę wszystkich zidentyfikowanych scenariuszy. Scenariusze można zidentyfikować poprzez zastosowanie różnych kryteriów pokrycia do modelu scenariusza (patrz Koomen *et al.*, 2006). Kryteria pokrycia mogą wymagać przetestowania każdego scenariusza więcej niż jeden raz. Na przykład scenariusz może wymagać dodatkowego pokrycia klas równoważności lub wartości brzegowych dla zmiennych występujących w scenariuszu. W takich przypadkach jeden scenariusz może wymagać przetestowania przez więcej niż jeden przypadek testowy.

Testowanie oparte na scenariuszach jest często wykonywane w ramach testowania systemowego lub akceptacyjnego. Przyjmuje ono formę kompleksowego testowania skoncentrowanego na przydatności funkcjonalnej systemu (patrz podrozdział 4.1) z perspektywy użytkownika. Testowanie oparte na scenariuszach może być jednak wykorzystywane również na innych poziomach testów (np. testowania integracyjnego modułów ze scenariuszami opartymi

na protokołach interakcji lub testowania modułowego stanowych, zorientowanych obiektowo klas ze scenariuszami wywołującymi różne metody) oraz w testowaniu niefunkcjonalnym (np. scenariusze mogą stanowić elementy profili operacyjnych wykorzystywanych w testach niezawodności, przenaszalności lub zgodności).

3.3. Techniki testowania oparte na regułach

Techniki testowania oparte na regułach weryfikują implementację bezstanowego zachowania elementu testowego. Zachowanie to określone jest przez reguły, które są ważne niezależnie od stanu elementu testowego (np. reguły biznesowe). W niniejszym sylabusie omówiono dwie techniki testowania oparte na regułach, a mianowicie:

- testowanie w oparciu o tablicę decyzyjną,
- testowanie metamorficzne.

3.3.1. Testowanie w oparciu o tablicę decyzyjną

Sylabus Certyfikowany Tester Poziom Podstawowy (ISTQB CTFL) opisuje podstawy testowania w oparciu o tablicę decyzyjną. Niniejszy sylabus omawia bardziej zaawansowane tematy w tym obszarze. Terminologia i notacja są zgodne z normą (OMG DMN, 2004).

W przypadku testowania w oparciu o tablicę decyzyjną, AT zazwyczaj rozpoczyna od utworzenia pełnej tablicy decyzyjnej lub przeanalizowania istniejącej, uzyskanej z podstawy testów. Liczba reguł pełnej tablicy decyzyjnej jest iloczynem liczby wartości jej warunków. Liczba ta rośnie wykładniczo wraz z liczbą warunków i ich wartości i motywuje do minimalizacji tablic decyzyjnych. Minimalizacja rozpoczyna się od oryginalnej tablicy decyzyjnej, która może być pełna lub nie, i prowadzi do uzyskania równoważnej tablicy decyzyjnej zawierającej mniej reguł.

Tablice decyzyjne można zminimalizować, łącząc reguły za pomocą operatora „–” (wartość nieistotna). Podczas łączenia reguł zaleca się pomijanie reguł nieosiągalnych (tj. reguł z kombinacją wartości warunków, które nigdy nie mogą wystąpić). Często reguły nieosiągalne są usuwane z tablicy decyzyjnej przed rozpoczęciem minimalizacji. Przykładowy algorytm systematycznej minimalizacji wyszukuje reguły z identycznymi akcjami, które różnią się tylko jednym warunkiem i obejmują wszystkie jego możliwe wartości. Reguły te są łączone, a różniąca się wartość warunku jest zastępowana znakiem „–”. Wynik systematycznej minimalizacji może zależeć od kolejności, w jakiej kolumny są minimalizowane, nie zawsze prowadząc do optymalnego rozwiązania. AT musi sprawdzić, czy możliwa jest dalsza minimalizacja.

Zadaniem AT jest dokonanie przeglądu tablicy decyzyjnej, w miarę możliwości z innymi interesariuszami, z uwzględnieniem następujących kryteriów:

- spójność (tj. jeśli dwie różne reguły odpowiadają tej samej kombinacji wartości warunków, to mają identyczne akcje),
- osiągalność (tj. tablica decyzyjna nie zawiera reguł nieosiągalnych),
- kompletność (tj. nie brakuje żadnej osiągalnej kombinacji wartości warunków),
- poprawność (tj. reguły modelują zamierzone zachowanie systemu).

Ponadto zaleca się, aby reguły nie nakładały się na siebie (tj. dla dowolnej kombinacji wartości warunków powinna istnieć co najwyżej jedna reguła). Nakładanie się reguł może wystąpić, gdy oryginalna tablica decyzyjna jest już zminimalizowana lub gdy reguły są nieprawidłowo połączone.

Procedura sumy kontrolnej (ang. *checksum procedure*) wykorzystuje liczbę i postać reguł w zminimalizowanej tablicy decyzyjnej do identyfikacji nakładania się reguł oraz luk. Dla każdej reguły w zminimalizowanej tablicy decyzyjnej obliczana jest liczba reguł, które reguła ta reprezentuje w oryginalnej tablicy decyzyjnej. Reguła bez „–" w warunkach (tj. z konkretnymi wartościami dla wszystkich warunków) otrzymuje wynik 1. Każda wartość „–" mnoży wynik reguły przez liczbę indywidualnych wartości dla danego warunku. Suma wyników reguł stanowi sumę kontrolną zminimalizowanej tablicy decyzyjnej. Jeśli suma kontrolna jest mniejsza niż suma kontrolna oryginalnej tablicy decyzyjnej, zminimalizowana tablica decyzyjna jest niekompletna. Jeśli suma kontrolna jest wyższa, niektóre reguły nakładają się na siebie lub istnieją dodatkowe reguły (np. nieosiągalne kombinacje warunków). Jeśli minimalizacja została przeprowadzona poprawnie, sumy kontrolne są równe. Jednak równość sum kontrolnych nie gwarantuje, że zminimalizowana tablica decyzyjna jest równoważna oryginalnej.

Pokrycie tablicy decyzyjnej jest mierzone poprzez podzielenie liczby przetestowanych kolumn przez całkowitą liczbę osiągalnych kolumn w tablicy decyzyjnej. Podczas implementacji przypadków testowych dla reguły tablicy decyzyjnej, AT musi zaimplementować warunki i akcje. W szczególności musi zdecydować, w jaki sposób zaimplementować wartości warunków oznaczone jako nieistotne („–"), ponieważ symbol „–" reprezentuje co najmniej dwie wartości. Jednakże, jeśli poziom ryzyka związany z tablicą decyzyjną jest wysoki, AT powinien powstrzymać się od minimalizacji i powinien mierzyć pokrycie osiągalnych kolumn pełnej tablicy decyzyjnej.

3.3.2. Testowanie metamorficzne

Testowanie metamorficzne (TM) to technika generowania przypadków testowych opartych na istniejącym źródłowym przypadku testowym (ang. *source test case*). Jeden lub więcej kolejnych, tzw. następczych przypadków testowych (ang. *follow-up test cases*) jest generowanych poprzez zmianę (metamorfozę) źródłowego przypadku testowego w oparciu o relację metamorficzną (RM). Relacja metamorficzna definiuje właściwość elementu testowego i opisuje, w jaki sposób zmiana danych wejściowych przypadku testowego przekłada się na zmianę wyniku oczekiwanego przypadku testowego.

AT łączy źródłowe i następujące przypadki testowe dla zadanej RM w procedurę testową ze wspólną oceną wyników. Jeśli wyniki spełniają relację metamorficzną, test jest zaliczony, w przeciwnym razie jest niezaliczony. W przypadku niezaliczenia, późniejsze debugowanie musi określić, który z poszczególnych przypadków testowych zakończył się niezaliczeniem.

Dla przykładu, rozważmy funkcję, która oblicza średnią zadanych liczb. Tworzony jest źródłowy przypadek testowy z serią liczb i oczekiwaną średnią, a następnie przypadek testowy jest uruchamiany w celu potwierdzenia, że został zaliczony. RM może stanowić, że każda permutacja serii liczb skutkuje tą samą średnią. Korzystając z tej RM, AT może utworzyć kilka następczych przypadków testowych, każdy z tym samym zestawem liczb na wejściu, ale w innej kolejności. Wynik oczekiwany pozostaje taki sam.

W przypadku tej samej funkcji, AT może również użyć innej RM, która określa, że jeśli każda liczba z serii zostanie przemnożona przez tę samą liczbę x, wówczas wynik oczekiwany zostanie również przemnożony przez x. Dzięki tej RM AT może utworzyć dowolną liczbę następczych przypadków testowych na podstawie źródłowego przypadku testowego, wybierając różne wartości x. Może to być szczególnie przydatne, gdy projektowanie i wykonywanie testów jest zautomatyzowane. AT

może również utworzyć następujące przypadki testowe łącząc dwie lub więcej RM (np. permutować i pomnożyć przez 2).

TM może być również stosowane, gdy zachodzi problem wyroczni testowej (patrz sekcja 1.3.4). W takiej sytuacji oczekiwane wyniki źródłowego przypadku testowego i następczych przypadków testowych nie są dostępne, więc ich wyniki nie mogą być indywidualnie oceniane. Przykładem może być program aktuarialny oparty na sztucznej inteligencji, który przewiduje wiek w chwili śmierci na podstawie dużego zbioru danych. RM może określić, że jeśli liczba wypalanych papierosów wzrośnie, to przewidywany wiek zgonu powinien się obniżyć.

Obecnie nie ma powszechnie uznanych kryteriów pokrycia dla TM, które zapewniałyby użyteczne kryteria wyjścia. Jednokrotne pokrycie każdej RM jest niewystarczające, ponieważ można uzyskać jedynie częściową weryfikację oczekiwanych wyników. AT może połączyć TM z testowaniem losowym, aby wygenerować wiele niskopoziomowych źródłowych przypadków testowych i następczych przypadków testowych dla tej samej RM. Na przykład we wspomnianym powyżej obliczaniu średniej można użyć generatora liczb losowych do wygenerowania różnych danych wejściowych dla źródłowego przypadku testowego, a także losowych permutacji i mnożników dla następczych przypadków testowych.

TM może być stosowane dla większości elementów testowych i do testowania funkcjonalnego oraz niefunkcjonalnego (np. testowanie obciążenia, generowanie obciążenia przez następujące przypadki testowe przy użyciu RM lub testowanie instalowalności z różnymi parametrami instalacji, które można wybierać w wielu sekwencjach). Jest to preferowana technika testowania systemów opartych na sztucznej inteligencji (ISTQB AI, 2021).

Więcej informacji o TM można znaleźć w (ISO/IEC/IEEE 29119-4, 2021) i w artykułach przeglądowych (Segura *et al.*, 2016; Segura *et al.*, 2020).

3.4. Testowanie oparte na doświadczeniu

AT stosuje podejścia i techniki testowania opartego na doświadczeniu, wykorzystując swoją wiedzę i wcześniejsze doświadczenie w testowaniu. W tym podrozdziale szczegółowo opisano wykorzystanie dokumentacji przez AT w testowaniu w sesjach i w testowaniu w oparciu o listę kontrolną (patrz ISTQB CFTL, sekcje 4.4.2 i 4.4.3). Ponadto omówiono testowanie w tłumie. Wszystkie techniki testowania oparte na doświadczeniu mogą być stosowane w testowaniu opartym na współpracy, takim jak testowanie oparte na testach akceptacyjnych (szczegóły można znaleźć w (ISTQB CTFL, podrozdział 4.5).

3.4.1. Karty opisu testu wspierające testowanie w sesjach

W przypadku testów eksploracyjnych, karta opisu testu definiuje misję sesji testowej, która określa jej zakres i cele oraz informacje takie jak ograniczenia, ramy czasowe i ryzyka. Służy ona jako mapa drogowa dla testów, zapewniając strukturę sesji testowych. Karty opisu testu pomagają AT skupić się na konkretnych obszarach lub funkcjach, które mają zostać przetestowane, jednocześnie zapewniając im elastyczność w eksploracji systemu, gdy jest to konieczne. Karta opisu testu nie określa konkretnych zestawów testowych, które będą wykonywane w każdej sesji testowej.

Przygotowując kartę opisu testu do testowania w sesjach, AT musi wziąć pod uwagę pewne czynniki, które mają wpływ na projekt karty, w szczególności:

- czynniki związane z klientami i wymaganiami (np. wymagania uzyskane od klientów, biznesowe przypadki użycia systemu, wymagania jakościowe) oraz mapy podróży użytkownika (tj. opis interakcji użytkownika z systemem w czasie),
- czynniki związane z produktem (np. przepływy funkcjonalne, główne cele produktu, cechy produktu, projekt oprogramowania i interfejsy),
- czynniki związane z zarządzaniem projektem (np. ograniczenia czasowe, cel projektu, szacowany nakład pracy i wartość biznesową).

Karta opisu testu składa się z misji opisującej cel testu oraz z różnych dodatkowych informacji.

Popularnym, lekkim formatem misji jest „Zbadaj [cel] za pomocą [zasobów], aby odkryć [informacje]" (Hendrickson, 2013), gdzie [cel] opisuje, co ma zostać zbadane (np. obszar, cecha, ryzyko, moduł, wymaganie), [zasoby] opisują, czego użyje AT (np. dane testowe, konfiguracje, narzędzia, ograniczenia, heurystyki i zależności), a [informacje] wyjaśniają, jaki rodzaj informacji AT chce znaleźć (np. oceny charakterystyk jakościowych, oczekiwany rodzaj defektów, naruszenia norm).

Karty opisu testu mogą zawierać między innymi następujące informacje dodatkowe (Ghazi *et al.*, 2017):

- informacje organizacyjne (np. czas trwania sesji testowej, data i godzina rozpoczęcia oraz imię i nazwisko testera),
- cele testu (np. motywacja testu i misja karty opisu testu),
- zakres testów (np. konkretne obszary zainteresowania w ramach systemu podlegającego testowaniu, poziom testów, techniki testowania, które mają zostać wykorzystane, pomysły na testy, kryteria wyjścia, priorytety, co ma być pokryte, a czego nie pokrywać testami),
- kryteria wejścia (tj. warunki wstępne, które muszą zostać spełnione, aby można było rozpocząć sesję testową),
- informacje związane z produktem (np. definicja, dane i przepływ pracy między modułami oraz architektura systemu),
- ograniczenia (tj. czego produkt nigdy nie może zrobić),
- opis środowiska testowego,
- istniejące źródła danych, informacje o produktach i narzędzia testowe, które pomogłyby w testowaniu,
- informacje historyczne (np. wcześniej wykryte defekty przykładowo związane z kompatybilnością, bieżące pytania otwarte, które odnoszą się do istniejących anomalii, oraz wzorce awarii związane z testami w przeszłości),
- ograniczenia i ryzyka (np. stosowane regulacje, zasady i normy).

Podczas sesji testowej AT tworzy dziennik testów, który zawiera pytania, obserwacje lub pomysły na przyszłe testy wraz z wynikami testów. Dane te są dokumentowane w arkuszu sesji testowej.

Zakres i szczegółowość informacji zawartych w poszczególnych kartach opisu testu mogą się różnić i wpływać na stopień elastyczności AT. Na przykład, zdefiniowanie jedynie ogólnych celów testów zapewnia szerokie pole do eksploracji, podczas gdy dodanie informacji na temat technik testowania, które mają zostać wykorzystane, może ograniczyć AT. Z drugiej strony, dodanie

informacji (np. czego system zdecydowanie nie może zrobić) może pomóc zmniejszyć prawdopodobieństwo zgłaszania wyników fałszywie pozytywnych.

3.4.2. Listy kontrolne wspierające techniki testowania oparte na doświadczeniu

Testowanie w oparciu o listę kontrolną jest powszechnie stosowaną techniką testowania ze względu na jej adaptowalność, prostotę i skuteczność w zapewnianiu jakości oprogramowania. Korzystając z list kontrolnych, AT upewnia się, że obejmują one wszystkie znane istotne aspekty elementu testowego, co zapobiega pominięciu krytycznych obszarów. Wprowadzają one również spójność między cyklami testowymi i między różnymi analitykami testów. Korzystając z listy kontrolnej, AT koncentruje się na ważnych aspektach. Listy kontrolne są formą rejestrowania wcześniejszych doświadczeń AT z awariami i defektami. Służą one jako przypomnienie lub źródło inspiracji w przypadku wyczerpania się pomysłów (np. podczas testowania eksploracyjnego). AT oszczędza czas poprzez ponowne wykorzystanie standardowej listy kontrolnej lub listy kontrolnej stworzonej przez jego współpracowników, ale tylko wtedy, gdy te listy kontrolne są istotne dla elementu testowego. Listy kontrolne pomagają zredukować ilość pracy potrzebnej do udokumentowania przypadków testowych, co może być dużym atutem, gdy wymagania i oprogramowanie stale się zmieniają.

Przygotowanie list kontrolnych jest często obowiązkiem AT. Listy kontrolne wspierają testowanie oparte na doświadczeniu, ponieważ pomagają organizować, strukturyzować i kierować testowaniem. Stworzenie listy kontrolnej wielokrotnego użytku, łatwej w utrzymaniu, przejrzystej i wydajnej wymaga wysiłku.

Poniższe kroki mogą być wskazówką do stworzenia odpowiedniej listy kontrolnej na potrzeby testów opartych na doświadczeniu.

Po pierwsze, AT określa zakres, cele i format listy kontrolnej, ponieważ wpływają one na dokładność testowania i wymagany poziom szczegółowości. Lista kontrolna „czytaj-zrób” (ang. *read-do checklist*) zawiera główne elementy, które należy wziąć pod uwagę w przypadku określonego procesu (np. elementy listy kontrolnej zawierają konkretne nieprawidłowe dane wejściowe dla pola wejściowego, które należy sprawdzić). Lista kontrolna „zrób-potwierdź” (ang. *do-confirm checklist*) służy jako pomoc w prowadzeniu procesu myślowego, dostarczając opartych na doświadczeniu pomysłów na dalsze badanie aplikacji (np. element listy kontrolnej może wymagać sprawdzenia, czy wyniki wyszukiwania są trafne)[1].

Następnie AT zbiera informacje niezbędne do zdefiniowania elementów listy kontrolnej. Może to obejmować zbieranie informacji od doświadczonych specjalistów, przeglądanie bibliotek defektów i taksonomii defektów (Beizer 1990; Kaner *et al*, 1999), przeglądanie odpowiedniej dokumentacji oraz analizę ryzyk, przypadków testowych i potencjalnych scenariuszy. Elementy listy kontrolnej powinny być jasne, konkretne, jednoznaczne, spójne, istotne, możliwe do utrzymania, wykonalne i mierzalne. Powinny być sformułowane jako pytania, na które można

[1] Lista „czytaj–zrób” jest zazwyczaj zestawem działań do wykonania i weryfikowaniu poprawności tuż po wykonaniu danego elementu listy. Nadaje się do weryfikowania zadań wykonywanych w specyficznej kolejności i jest powszechnie stosowana w wysoce sformalizowanych, ustrukturyzowanych dziedzinach biznesowych (np. chirurgia, lotnictwo). Lista „zrób–potwierdź” zazwyczaj jest używana po wykonaniu zadań z pamięci lub na podstawie doświadczenia i jest typowym narzędziem doświadczonych ekspertów, dając im większą swobodę niż listy „czytaj–zrób” (przyp. tłum.).

odpowiedzieć „tak”, „nie” lub „nie dotyczy”. Wymagają one przypisania priorytetów w oparciu o ich znaczenie, potencjalny wpływ i poziom ryzyka. Listy kontrolne nie są kompleksowymi przewodnikami, lecz raczej szybkimi i prostymi narzędziami dla specjalistów.

Wreszcie, AT strukturyzuje listę kontrolną, kategoryzując jej elementy w logiczne grupy w oparciu o obszary funkcjonalne, role użytkowników, poziomy testów lub inne istotne kryteria. Tworzenie oddzielnych kategorii może być przydatne w przypadku długich list kontrolnych.

Tam, gdzie to możliwe, należy korzystać z szablonów i norm. AT może zaoszczędzić wysiłek, wykorzystując istniejące szablony lub predefiniowane listy kontrolne dostosowane do standardów branżowych i najlepszych praktyk.

Lista kontrolna nigdy nie jest ostateczna. AT regularnie ją przegląda i udoskonala, dostosowując ją do nowych ustaleń, zmienionych priorytetów, informacji zwrotnych od innych testerów lub wniosków wyciągniętych z poprzednich cykli testowych. Dzieląc się listą kontrolną z innymi testerami, TA promuje spójność i współpracę oraz pomaga im lepiej zrozumieć elementy testowe i krytyczne obszary, na których należy się skupić podczas testowania.

3.4.3. Testowanie w tłumie

Testowanie w tłumie polega na dystrybucji zadań testowych wśród grupy wewnętrznych lub zewnętrznych testerów z różnych środowisk i lokalizacji. Może to być opłacalny sposób walidacji użyteczności i obejmuje zarówno funkcjonalne, jak i niefunkcjonalne charakterystyki jakościowe (Alyahya, 2020; Leicht *et al*., 2017).

Korzyści płynące z testowania w tłumie to m.in.:

- **Zróżnicowane środowiska testowe**. Testerzy mogą znajdować się w różnych lokalizacjach geograficznych i korzystać z różnych konfiguracji środowisk z szeroką gamą urządzeń, przeglądarek i warunków sieciowych.
- **Większa elastyczność**. Łatwa skalowalność do obsługi wielu testów w krótkim czasie.
- **Opłacalność**. Testowanie w tłumie jest zazwyczaj tańsze niż utrzymywanie dużego i zróżnicowanego wewnętrznego zespołu testowego lub uzupełnianie go zewnętrznymi usługami testowymi.
- **Szybka informacja zwrotna**. Testerzy mogą szybko przekazywać informacje zwrotne, pomagając wcześnie identyfikować i usuwać awarie.
- **Perspektywa rzeczywistego użytkownika**. Testerzy mogą być rzeczywistymi użytkownikami aplikacji i mogą lepiej ocenić doświadczenie użytkownika (ang. *user experience*) i użyteczność. Może to być szczególnie cenne w testach akceptacyjnych.
- **Zmienność**. Ponieważ testy wykonywane są za każdym razem przez wielu różnych testerów, nie są one powtarzalne. Choć może to stanowić ograniczenie, skutkuje również szerszym pokryciem, co zwiększa szanse na znalezienie większej liczby defektów.

Testowanie w tłumie posiada również swoje ograniczenia, m.in.:

- **Problem z jakością testów**. Jakość testowania może się znacznie różnić w zależności od umiejętności poszczególnych testerów, choć może to nie być istotne, np. gdy celem jest uzyskanie informacji zwrotnej na temat doświadczenia użytkownika.
- **Wyzwania komunikacyjne**. Koordynacja z wieloma testerami z różnych lokalizacji, w różnych strefach czasowych, z różnicami kulturowymi i barierami językowymi może stanowić wyzwanie.
- **Bezpieczeństwo**. Udostępnianie oprogramowania zewnętrznym testerom wiąże się z ryzykiem w zakresie bezpieczeństwa i poufności danych. Odpowiednie środki mogą złagodzić to ryzyko, umożliwiając odpowiedzialne korzystanie z testowania w tłumie bez ujawniania wrażliwych szczegółów lub ułatwiania plagiatu.
- **Dokumentacja i raportowanie**. Zapewnienie kompleksowej dokumentacji testowej i zarządzanie dużą liczbą wyników, w tym duplikatów i wyników fałszywie pozytywnych, może stanowić wyzwanie w przypadku dużej i zróżnicowanej grupy testerów.

Testowanie w tłumie to podejście, które nie zastępuje analityków testów stosujących techniki testowe, ale może zwiększyć pokrycie różnorodnych środowisk testowych.

3.5. Zastosowanie najwłaściwszych technik testowania

Testowanie powinno być jak najbardziej efektywne i wydajne w danym kontekście. W tym celu AT wspiera kierownika testów w wyborze najbardziej odpowiednich technik testowania. Ponadto AT może wykorzystywać automatyzację do wspierania czynności testowych. Obejmuje to automatyzację projektowania testów, opisaną w tym podrozdziale, oraz wspieranie automatyzacji wykonywania testów, opisane w sekcji 1.3.6.

3.5.1. Wybór technik testowania w celu łagodzenia ryzyk produktowych

Wybór najbardziej odpowiednich technik testowania ma kluczowe znaczenie dla skutecznego i wydajnego łagodzenia ryzyka produktowego i zależy od wielu czynników, w tym następujących:

Cele testów (patrz ISTQB CTFL, sekcja 1.1.1) określają, które aspekty przedmiotu testów należy ocenić. Kierują one wyborem technik testowania i zależą od typu systemu (np. testowanie dziedziny dla obliczeń numerycznych w inżynierii vs. testowanie w oparciu o tablicę decyzyjną aplikacji do zarządzania ryzykiem kredytowym).

Ryzyka produktowe wiążą się z potencjalnymi defektami, które można najlepiej wykryć za pomocą określonych technik testowania, ponieważ większość tych technik koncentruje się na wykrywaniu określonych rodzajów defektów (patrz podrozdziały 3.1, 3.2, 3.3, 3.4). Na przykład:

- Techniki testowania oparte na danych mogą wykrywać defekty w obsłudze danych, implementacji dziedziny, interfejsach użytkownika, obliczeniach i kombinacjach parametrów.
- Techniki testowania oparte na zachowaniu mogą wykrywać defekty wymagań użytkownika, takie jak brakujące funkcje, problemy komunikacji i defekty przetwarzania.
- Techniki testowania oparte na regułach mogą wykrywać defekty w logice i przepływie sterowania.

Analiza ryzyka pomaga również określić odpowiednie podejście do testów, np.:

- Kryteria wyjścia oparte na pokryciu. Im wyższy poziom ryzyka, tym bardziej rygorystyczne pokrycie może być wymagane (np. testowanie wszystkich kombinacji zamiast testowania sposobem par w pokryciu kombinatorycznym). AT musi jednak zawsze brać pod uwagę kompromis między siłą pokrycia a wymaganym wysiłkiem testowym.
- Testowanie oparte na doświadczeniu może być stosowane, gdy zdefiniowanie pokrycia jest trudne, gdy poziom ryzyka jest niski lub gdy harmonogram projektu jest napięty.

Podstawa testu. Jeśli specyfikacja przedmiotu testów wykorzystuje modele, AT może wykorzystać techniki testowania oparte na tych modelach. Jeśli wyprowadzenie wyroczni testowej z podstawy testów jest trudne lub niemożliwe, można zastosować techniki takie jak testowanie metamorficzne lub techniki testowania oparte na doświadczeniu.

Znajomość powtarzających się typów defektów może wskazywać na wybór technik testowania, które koncentrują się na wykrywaniu takich defektów (np. testowanie oparte na listach kontrolnych). Jeśli oczekuje się wykrycia podobnych defektów jak w poprzednich iteracjach lub projektach, rozsądne może być zastosowanie tych samych skutecznych technik testowania.

Wiedza i doświadczenie testera. Jeśli AT nie jest zaznajomiony z daną techniką testowania, nie zaleca się stosowania jej w krytycznych zadaniach testowych. Znajomość dziedziny może również wpływać na wybór technik testowania. Przykładowo, niewielka wiedza biznesowa lub jej brak może powodować, że techniki testowania takie jak testowanie eksploracyjne mogą być nieskuteczne.

Zastosowany cykl życia oprogramowania. Sekwencyjny model wytwarzania oprogramowania sprzyja stosowaniu bardziej formalnych technik. Z kolei model iteracyjny może być bardziej odpowiedni do przyjęcia lekkich technik testowania (np. technik testowania opartych na doświadczeniu) lub gdy projektowanie testów można zautomatyzować.

Wymagania klientów i umowy. Umowy mogą wprost wymagać przeprowadzenia określonych testów (np. pod względem określonych poziomów testów lub typów testów), co wpływa na wybór technik testowania (np. kryteria akceptacji z zestawem scenariuszy dostarczonych przez klienta sugerują użycie techniki testowania opartej na scenariuszach).

Wymagania regulacyjne. Gdy projekt musi być zgodny z normą, może wymagać użycia określonych technik testowania. Na przykład norma (ISO 26262-6, 2018) wymaga stosowania technik testowania takich jak podział na klasy równoważności, analiza wartości brzegowych lub zgadywanie błędów, w zależności od tzw. poziomu nienaruszalności bezpieczeństwa ASIL (ang. *Automotive Safety Integrity Level*) przypisanego do przedmiotu testów.

Ograniczenia projektu, takie jak czas i budżet, mogą wpływać na wykorzystanie technik, które są czasochłonne lub wymagają użycia kosztownych zasobów.

Techniki testowania są często łączone w celu zwiększenia wydajności i skuteczności wykrywania defektów. Na przykład:

- Analiza wartości brzegowych może być wykorzystywana do testowania warunków dozoru w testowaniu przejść pomiędzy stanami.

- Testowanie dziedziny może być wykorzystane w testowaniu opartym na scenariuszach w celu określenia wartości warunku z tablicy decyzyjnej lub zmiennej występującej w systemie podlegającym testowaniu.
- Testowanie oparte na scenariuszach można połączyć z testowaniem decyzji, białoskrzynkową techniką testowania omówioną w (ISTQB TTA, 2021), w celu rygorystycznego pokrycia decyzji w procesie biznesowym, patrz np. tzw. testowanie cyklu procesowego (Koomen *et al.*, 2006).
- Testowanie oparte na scenariuszach może być łączone z pokryciem pętli (ang. *round-trip coverage*) w celu uwzględnienia specyficznych ryzyk związanych z cyklicznymi czynnościami w procesach biznesowych.

3.5.2. Korzyści i ryzyka automatyzacji projektowania testów

AT może używać narzędzi do stosowania technik testowania, w szczególności technik czarnoskrzynkowych. Automatyzując projektowanie testów, AT tworzy model przedmiotu testów i na jego podstawie automatycznie generuje testalia. Na przykład, AT projektuje model przejść pomiędzy stanami i pozwala narzędziu opartemu na modelu generować przypadki testowe tak, by spełniły one pokrycie pętli.

Automatyzacja projektowania testów często poprawia wydajność i skuteczność testowania. Jej zalety obejmują m.in.:

- **Zapobieganie defektom**. Modelowanie na potrzeby testowania jest skutecznym sposobem oceny jakości podstawy testów (patrz sekcja 5.2.1).
- **Rozszerzone możliwości**. Automatyzacja pozwala na zastosowanie bardziej złożonych technik testowania i kryteriów pokrycia, takich jak testowanie kombinatoryczne, testowanie losowe lub pokrycie N-przełączeń, zmniejszając ryzyko pozostawienia nieprzetestowanego kodu.
- **Większa zrozumiałość**. Kryteria wyboru testów określone w narzędziu w bardziej przejrzysty i widoczny sposób odnoszą się do warunków testowych i w bardziej zrozumiały sposób uzasadniają wygenerowane pokrycie.
- **Mniej powtarzalnej pracy**. Testalia mogą być automatycznie generowane z modelu, ograniczając manualną, powtarzalną pracę, taką jak specyfikowanie testów.
- **Mniejsze nakłady na pielęgnację**. Model testów jest jedynym źródłem prawdy (ang. *single source of truth*) wykorzystywanym do tworzenia testaliów. W rezultacie tylko on podlega pielęgnacji.
- **Mniej wadliwych testaliów**. Praca manualna jest podatna na błędy. Testalia generowane przez narzędzia cechują się wyższą jakością i spójnością.
- **Lepsza współpraca w zespole**. Interesariusze mogą przeglądać model w celu znalezienia defektów lub lepszego zrozumienia warunków testowych.
- **Zwiększone śledzenie powiązań**. Łatwiej jest powiązać elementy modelu testu z warunkami testowymi niż same przypadki testowe. Jeśli narzędzie obsługuje taką możliwość, wygenerowane przypadki testowe odziedziczą te powiązania, poprawiając ogólne śledzenie powiązań w testowaniu.
- **Różne formaty wyjściowe**. Testalia mogą być generowane w różnych formatach wyjściowych, zgodnie z wymaganiami innych narzędzi i dalszych działań.

AT musi również wziąć pod uwagę ryzyka związane z automatyzacją projektowania testów. Obejmują one: przeoczenie warunków testowych, które nie są pokazane w modelu, niedoszacowanie wysiłku związanego z utrzymaniem modelu, trudność w zrozumieniu modelu przez interesariuszy oraz ogólne ryzyka związane z automatyzacją testów (patrz ISTQB CTFL, podrozdział 6.2).

4. Testowanie charakterystyk jakościowych (60 min.)

Słowa kluczowe

adaptowalność, adekwatność funkcjonalna, doświadczenie użytkownika, elastyczność, funkcjonalność, instalowalność, kompletność funkcjonalna, poprawność funkcjonalna, testowanie funkcjonalne, użyteczność, współdziałanie, zdolność interakcji, zgodność

Cele nauczania dla rozdziału 4

4.1 Testowanie funkcjonalności

TA-4.1.1 (K2) Rozróżniać testowanie poprawności, adekwatności i kompletności funkcjonalnej.

4.2 Testowanie użyteczności

TA-4.2.1 (K2) Wyjaśnić wkład analityka testów w testowanie użyteczności.

4.3 Testowanie elastyczności (przenaszalności)

TA-4.3.1 (K2) Wyjaśnić wkład analityka testów w testowanie adaptowalności i instalowalności.

4.4 Testowanie zgodności (kompatybilności)

TA-4.4.1 (K2) Wyjaśnić wkład analityka testów w testowanie współdziałania.

Wstęp

Niniejszy sylabus jako podstawę przyjmuje model jakości oprogramowania przedstawiony w normie ISO/IEC 25010 (2023) i omawia te charakterystyki jakościowe, na których skupia się AT. Jednakże użyty w sylabusie termin „użyteczność" (ang. *usability*) różni się od terminu użytego w normie (zdolność interakcji, ang. *interaction capability*), aby zachować spójność z sylabusem testowanie użyteczności (ISTQB UT, 2018).

Dodatek F zawiera przegląd modelu jakości wg (ISO/IEC 25010, 2023), zmiany w porównaniu z poprzednią wersją tej normy oraz powiązane sylabusy ISTQB®, które koncentrują się na testowaniu określonych charakterystyk jakościowych oprogramowania.

4.1. Testowanie funkcjonalności

Testowanie funkcjonalności (ang. *functional suitability testing*) jest jednym z podstawowych zadań AT. Podczas gdy (ISTQB CTFL) krótko podsumowuje ten typ testów, niniejszy sylabus zawiera więcej szczegółów, omawiając poszczególne podcharakterystyki funkcjonalności.

Model jakości produktu ISO/IEC 25010 (2023) rozróżnia trzy podcharakterystyki funkcjonalności. Chociaż każdą z nich można oceniać za pomocą testów funkcjonalnych, nie każda czynność testowania funkcjonalnego odnosi się do nich w taki sam sposób. AT powinien być w stanie wybrać odpowiednie poziomy testów i techniki testowania, aby być w stanie łagodzić ryzyka produktowe związane z określoną podcharakterystyką funkcjonalności.

Testowanie kompletności funkcjonalnej powinno obejmować wszystkie określone zadania oprogramowania i zamierzone cele użytkowników. Kluczową kwestią jest to, czy wszystko, o co poproszono, zostało zaimplementowane.

Do kwestii kompletności funkcjonalnej należy podejść jak najwcześniej, dokonując przeglądu specyfikacji wymagań (w sekwencyjnych modelach wytwarzania oprogramowania) oraz omawiając historyjki użytkownika, w tym kryteria akceptacji, podczas wspólnego pisania historyjek użytkownika (w ramach zwinnego wytwarzania oprogramowania). Podczas testowania systemowego, testowania integracji systemów i testowania akceptacyjnego kompletność funkcjonalna może być testowana dynamicznie.

Techniki testowania oparte na zachowaniu, takie jak testowanie oparte na scenariuszach, są dobrze dopasowane do tego zadania, chociaż inne techniki czarnoskrzynkowe są również odpowiednie. Śledzenie powiązań pomiędzy podstawą testów, warunkami testowymi i przypadkami testowymi jest niezbędne przy określaniu osiągniętego poziomu kompletności funkcjonalnej.

Testowanie poprawności funkcjonalnej odpowiada na pytanie, czy rzeczywiste wyniki są poprawne (np. dokładne, precyzyjne i spójne) dla prawidłowych i nieprawidłowych danych wejściowych. Kluczowe znaczenie ma znalezienie skutecznej wyroczni testowej, która może dostarczyć w szczegółowy sposób oczekiwane wyniki.

Poprawność funkcjonalna może być testowana na dowolnym poziomie testów. Jeśli chodzi o przesunięcie w lewo (ang. *shift left*), większość testów poprawności funkcjonalnej powinna odbywać się podczas testowania modułowego i testowania integracji modułów. Nawet jeśli AT

nie jest odpowiedzialny za te poziomy testów, powinien wnieść do nich swój wkład, aby jak najlepiej osiągnąć cele testów.

Do testowania poprawności funkcjonalnej nadają się wszystkie techniki czarnoskrzynkowe, techniki testowania oparte na doświadczeniu oraz podejście do testowania oparte na współpracy.

Testowanie adekwatności funkcjonalnej weryfikuje, czy funkcje ułatwiają realizację określonych zadań i celów. Nacisk kładziony jest na to, czy wszystko, co zostało zaimplementowane, spełnia potrzeby użytkowników.

Testowanie adekwatności funkcjonalnej może obejmować przeglądy projektu interfejsu użytkownika, zwłaszcza w przypadku aplikacji interaktywnych. Testowanie dynamiczne rozpoczyna się od testowania systemowego i testowania akceptacyjnego (w sekwencyjnych modelach wytwarzania oprogramowania), a także sesji demonstracyjnych (ang. *demo sessions*) w zwinnym rozwoju oprogramowania.

Do testowania adekwatności funkcjonalnej najbardziej nadaje się testowanie eksploracyjne i podejście do testowania oparte na współpracy. Odpowiednie są również czarnoskrzynkowe techniki testowania oparte na zachowaniu.

4.2. Testowanie użyteczności

Użyteczność odnosi się do szerokiej koncepcji charakterystyk jakościowych związanych z użytkownikiem, obejmujących zdolność interakcji (ang. *interaction capability*) z modelu jakości produktu (ISO/IEC25010, 2023) i korzystność (ang. *beneficialness*) z modelu jakości w użyciu (ISO/IEC 25019, 2023). Więcej informacji na temat testowania użyteczności można znaleźć w (ISTQB UT, 2018) oraz (ISO 9241-210, 2019).

Testowanie użyteczności zazwyczaj koncentruje się na ocenie następujących aspektów:

- zdolności interakcji – umożliwienia użytkownikom wykonywania zadań w określonych kontekstach użytkowania (ISO/IEC 25010, 2023) w sposób skuteczny, wydajny i satysfakcjonujący,
- doświadczenia użytkownika – percepcji użytkowników przed, w trakcie i po interakcji z przedmiotem testów,
- dostępności (ang. *accessibility*) – zapewnieniu, że użytkownicy niepełnosprawni, z różnych środowisk kulturowych lub z barierami językowymi mogą korzystać z systemu zarówno skutecznie, jak i wydajnie.

AT może współpracować przy testowaniu użyteczności od wczesnych faz, wykorzystując swoją wiedzę na temat docelowych grup użytkowników, ich celów, kontekstu użytkowania, potencjalnych trudności w korzystaniu z systemu lub negatywnych doświadczeń użytkowników.

AT może oceniać użyteczność poprzez wniesienie wkładu do następujących technik:

- **Przeglądy użyteczności** (ang. *usability reviews*) – przeprowadzane przez ekspertów ds. użyteczności w celu zidentyfikowania potencjalnych problemów z użytecznością i odchyleń od ustalonych kryteriów. Przeglądy użyteczności mogą mieć różną postać, od nieformalnych przeglądów do inspekcji. AT może dostosować kryteria przeglądu do konkretnych potrzeb grup użytkowników, konkretnych celów i priorytetów biznesowych oraz kontekstu użytkowania (np. dostosowując ogólną listę kontrolną użyteczności).

- **Sesje testów użyteczności** (ang. *usability test sessions*) obejmują przyszłych użytkowników lub ich przedstawicieli próbujących wykonać wcześniej zaprojektowane zadania, aby ocenić, czy zadania te można wykonać skutecznie, wydajnie i satysfakcjonująco. AT może przyczynić się do zaprojektowania scenariuszy dla sesji testów użyteczności zgodnie z personami, grupami użytkowników lub profilami operacyjnymi.
- **Kwestionariusze użytkowników lub ankiety** – zawierają ocenę i informacje zwrotne i mierzą zadowolenie użytkowników. Przykładami kwestionariuszy użytkownika są *Software Usability Measurement Inventory* (SUMI, 1991) i *Website Analysis and Measurement Inventory* (WAMMI, 1999). AT może pomóc w zaprojektowaniu kwestionariusza i analizie odpowiedzi, aby uwzględnić konkretne cele docelowych użytkowników w ich kontekście użytkowania.

W testowaniu dostępności powszechnym celem testów jest weryfikacja zgodności z normami dostępności. Międzynarodowy standard *Web Content Accessibility Guidelines* (WCAG, 2023) definiuje trzy poziomy zgodności (A, AA i AAA), reprezentujące rosnące stopnie dostępności treści internetowych. Przykładami norm krajowych są brytyjska ustawa o równości – *United Kingdom's Equality Act* (EA, 2010) i amerykańska ustawa o niepełnosprawności – *The Americans with Disabilities Act* (ADA, 2010). AT może zidentyfikować wymagany poziom zgodności i konkretne potrzeby zamierzonej grupy docelowej poprzez analizę kontekstu użytkowania.

4.3. Testowanie elastyczności (przenaszalności)

Testowanie elastyczności (ang. *flexibility testing*), znane również jako testowanie przenaszalności (ang. *portability testing*) weryfikuje, czy przedmiot testów można dostosować do zmian w kontekście jego użytkowania lub środowiska systemowego. Model jakości produktu (ISO/IEC 25010, 2023) rozróżnia następujące podcharakterystyki przenaszalności:

- adaptowalność (ang. *adaptability*),
- instalowalność (ang. *installability*),
- skalowalność (ang. *scalability*),
- zastępowalność (ang. *replaceability*).

Przenaszalność ma zarówno aspekty techniczne, jak i nietechniczne. Niniejszy podrozdział koncentruje się na tym, w jaki sposób AT może przyczynić się do testowania adaptowalności i instalowalności. Skalowalność i zastępowalność są związane z aspektami technicznymi i dlatego są omówione odpowiednio w sylabusach testowanie wydajności (ISTQB PT, 2018) oraz techniczny analityk testów (ISTQB TTA, 2021).

Testowanie adaptowalności weryfikuje, czy przedmiot testów może zostać dostosowany (zaadaptowany) lub przeniesiony do docelowego sprzętu, oprogramowania lub innych środowisk operacyjnych lub użytkowych.

AT wspiera testowanie adaptowalności poprzez identyfikację zamierzonych środowisk docelowych (np. wersji obsługiwanych mobilnych systemów operacyjnych i wersji przeglądarek, które mogą być używane) i projektowanie testów uwzględniających kombinacje tych środowisk. Ponieważ wymaga to danych testowych reprezentujących różne konfiguracje parametrów środowiska, często stosuje się techniki takie jak testowanie kombinatoryczne (patrz sekcja 3.1.2). Innym przykładem jest integracja różnych modułów w projektach klientów w celu zapewnienia

kompatybilności w środowiskach docelowych. W zależności od ryzyka produktowego, AT projektuje i wykonuje testy dymne lub bardziej kompleksowy zestaw testowy w celu sprawdzenia, czy przedmiot testów jest prawidłowo dostosowany do środowiska docelowego.

Przestrzegając dobrych praktyk w zakresie testowania adaptowalności (np. wczesne definiowanie środowisk docelowych, stosowanie testów kombinatorycznych, przeprowadzanie testów dymnych w nowych środowiskach i monitorowanie defektów specyficznych dla środowiska), AT może odkryć defekty, które mogą ograniczyć żywotność i użyteczność oprogramowania (np. łatwość użytkowania na ekranach o różnych rozmiarach). Praca AT w ramach testowania adaptowalności powinna być również wspierana przez zautomatyzowane testy międzyplatformowe implementowane przez inżynierów ds. automatyzacji testów. Rygorystyczne testy adaptowalności mogą wcześnie wykryć problemy specyficzne dla środowiska, pomagając uniknąć częstych dużych aktualizacji lub przeprojektowań po wdrożeniu, obniżając tym samym koszty utrzymania.

Testowanie instalowalności weryfikuje, czy przedmiot testów może być instalowany, odinstalowywany, aktualizowany i rekonfigurowany poprawnie w określonych środowiskach. Testowanie instalowalności wykracza poza zwykłe sprawdzenie, czy procedura instalacji przebiega poprawnie od początku do końca.

Typowe cele testów instalowalności, na których skupia się AT, są następujące:

- weryfikacja poprawności wykonania procedur instalacji dla różnych konfiguracji parametrów środowiska. Podobnie jak w przypadku testowania adaptowalności, pomocne są tu techniki testowania takie jak testowanie kombinatoryczne.
- projektowanie i wykonywanie testów w celu określenia, czy przedmiot testów działa prawidłowo po instalacji lub aktualizacji.
- sprawdzenie, jak łatwo jest użytkownikom zainstalować, odinstalować lub zaktualizować oprogramowanie. Obejmuje to także zapoznanie się z dokumentacją instalacyjną.
- przetestowanie zachowania związanego z uprawnieniami, szczególnie w przypadku aplikacji mobilnych (patrz ISTQB MAT, 2019).

4.4. Testowanie zgodności (kompatybilności)

Testowanie zgodności weryfikuje, czy przedmiot testów jest kompatybilny (zgodny) z innymi modułami lub systemami, gdy jest używany. Model jakości produktu (ISO/IEC 25010*,* 2023*)* rozróżnia dwie podcharakterystyki zgodności: współdziałanie (ang. *interoperability*) i współistnienie (ang. *coexistence*). W związku z tym testowanie zgodności można podzielić na następujące dwa rodzaje testów:

- **Testowanie współdziałania**, które weryfikuje zgodność z modułami lub systemami, z którymi przedmiot testów ma współdziałać. Testy współdziałania są zazwyczaj czarnoskrzynkowymi testami funkcjonalnymi, więc zwykle odpowiada za nie AT.
- **Testowanie współistnienia**, które weryfikuje, czy przedmiot testów może współdzielić swoje środowisko docelowe z innymi modułami lub systemami bez zakłóceń. Ten typ testów technicznych omówiono w sylabusie Certyfikowany Tester Poziom Zaawansowany Techniczny Analityk Testów (ISTQB TTA, 2021).

Celem testowania współdziałania jest sprawdzenie, czy dwa lub więcej modułów lub systemów może wymieniać informacje i wzajemnie wykorzystywać wymienione informacje. Gdy wymiana informacji wiąże się z transformacją danych, testowanie współdziałania musi obejmować weryfikację tej transformacji. Testowanie współdziałania jest szczególnie ważne, gdy wiele systemów musi współpracować, udostępniać dane lub wspólnie wykonywać zadania. Dotyczy to zwłaszcza nowoczesnych architektur oprogramowania, takich jak rozwiązania chmurowe, usługi sieciowe, mikrousługi, konteneryzacja i Internet Rzeczy.

Współdziałanie zwykle ma miejsce na różnych poziomach architektury. AT musi zrozumieć możliwe interakcje, aby zdefiniować odpowiednie warunki testowe w celu ich pokrycia. Nie wszystkie interakcje mogą być udokumentowane. AT może pośrednio pobierać informacje o interakcjach z dokumentacji architektonicznej i projektowej. Dlatego kluczowe jest zrozumienie tej dokumentacji, aby zapewnić, że wszystkie ważne aspekty interakcji zostaną przetestowane.

Testy współdziałania mogą wykrywać defekty w:

- transformacji danych na potrzeby wymiany danych,
- interpretacji lub wykorzystaniu wymienianych danych,
- przepływach i protokołach komunikacyjnych,
- zgodności z normami,
- kompleksowej (ang. *end-to-end*) funkcjonalności,
- dokumentacji projektowej.

Testowanie współdziałania jest zwykle stosowane w testowaniu integracyjnym. Do testowania współdziałania dobrze nadają się czarnoskrzynkowe techniki testowania, takie jak techniki testowania oparte na danych (które koncentrują się na wymienianych danych), techniki testowania oparte na zachowaniu (w celu interpretacji lub wykorzystania danych bądź kompleksowej funkcjonalności), a także techniki testowania oparte na regułach (w celu weryfikacji transformacji danych). Techniki testowania oparte na doświadczeniu mogą stanowić użyteczne uzupełnienie testów czarnoskrzynkowych. Przypadki testowe z testów funkcjonalnych lub testów adaptowalności mogą być ponownie wykorzystane do testowania współdziałania.

Przykładami ogólnych standardów współdziałania są (ISO 15745, 2023) i (ISO 16100, 2009). Z kolei (ETSI EG 202 237, 2010) zawiera przykład konkretnej metodologii testowania współdziałania w dziedzinie telekomunikacji.

5. Zapobieganie defektom (225 min.)

Słowa kluczowe

analiza przyczyny podstawowej, czytanie oparte na perspektywie, przeglądanie ad hoc, przeglądanie oparte na liście kontrolnej, przeglądanie oparte na rolach, przeglądanie oparte nascenariuszu, technika przeglądu, testowanie oparte na modelu, wynik testu, zapobieganie defektom

Cele nauczania dla rozdziału 5

5.1 Praktyki zapobiegania defektom

TA-5.1.1 (K2) Wyjaśnić wkład analityka testów w zapobieganie defektom.

5.2 Wspieranie powstrzymania fazowego

TA-5.2.1 (K3) Użyć modelu przedmiotu testów w celu wykrywania defektów w specyfikacji.

TA-5.2.2 (K3) Zastosować technikę przeglądu do podstawy testów w celu znalezienia defektów.

5.3 Łagodzenie ponownego występowania defektów

TA-5.3.1 (K4) Przeanalizować wyniki testów w celu identyfikacji udoskonaleń w wykrywaniu. defektów

TA-5.3.2 (K2) Wyjaśnić, jak klasyfikacja defektów wspiera analizę przyczyny podstawowej.

Wstęp

Celem zapobiegania defektom jest wdrożenie działań, które zmniejszają prawdopodobieństwo (ponownego) wystąpienia defektów w produktach pracy i łagodzą przenikanie defektów do kolejnych faz cyklu wytwarzania oprogramowania. Wysiłki te skutkują szeregiem ważnych korzyści, w tym zmniejszeniem kosztów i pracy, zwiększeniem produktywności i poprawą jakości produktu. Zapobieganie defektom jest obowiązkiem całego zespołu. AT może wykorzystać do tego swoją specyficzną wiedzę i doświadczenie.

Praktyki zapobiegania defektom obejmują:

- zapobieganie wprowadzaniu defektów (część działań związanych z zapewnieniem jakości),
- zapobieganie przedostawaniu się defektów do kolejnych faz (patrz podrozdział 5.2),
- zapobieganie ponownemu występowaniu defektów (patrz podrozdział 5.3).

5.1. Praktyki zapobiegania defektom

AT może przyczynić się do zapobiegania defektom na kilka sposobów, wykorzystując swoją wiedzę dziedzinową, wiedzę testerską i umiejętności analityczne. Przykłady obejmują:

- Udział w analizie ryzyka – zapewnienie, że zidentyfikowane ryzyka są odpowiednio złagodzone (np. poprzez wybór najbardziej odpowiednich technik testowania).
- Przegląd wymagań, modeli i specyfikacji – pozwalający na wczesne wykrycie defektów w podstawie testów, zapobiegając ich przedostaniu się do kodu, a tym samym znacznie zmniejszając koszty ich usuwania.
- Udział w retrospektywach – umożliwiający identyfikację potencjalnych udoskonaleń w analizie, projektowaniu, implementacji i wykonywaniu testów (np. poprzez stosowanie bardziej skutecznych technik testowania, ukierunkowanie testów na określone obszary ryzyka lub ulepszanie danych testowych i środowisk testowych w celu zmniejszenia zarówno wyników fałszywie pozytywnych, jak i fałszywie negatywnych), aby lepiej zapobiegać przedostawaniu się defektów do kolejnych faz cyklu wytwarzania oprogramowania.
- Gromadzenie i ocenę danych dotyczących defektów – wspieranie analizy podstawowej przyczyny i doskonalenie procesów poprzez gromadzenie szczegółowych danych dotyczących defektów w celu umożliwienia ich klasyfikacji i analizy statystycznej, a tym samym ułatwienia przeprowadzenia analizy podstawowej przyczyny.
- Udział w analizie podstawowej przyczyny – zapobieganie ponownemu wystąpieniu defektów poprzez proponowanie działań korygujących w celu usunięcia zidentyfikowanych podstawowych przyczyn.

Oprócz udziału w zapobieganiu defektom, AT ocenia (zwykle w porozumieniu z kierownikiem testów), czy proponowane środki przyniosły pożądany efekt. Przykłady metryk, które pomagają ocenić skuteczność tych środków, obejmują:

- Efektywność usuwania defektów (ang. *Defect Removal Efficiency*, DRE) – mierzy stosunek liczby defektów usuniętych przed wydaniem do całkowitej liczby defektów. Mianownik (wartość nieznana) można oszacować lub zastąpić całkowitą liczbą defektów wykrytych do pewnego momentu (np. do 6 miesięcy po wydaniu). Wysoka wartość DRE wskazuje

na mniejszą liczbę defektów przenikających do produkcji, co może oznaczać lepsze praktyki zapobiegania defektom. Jednak DRE nie rozróżnia, czy defekt wychwycony był poprzez zapobieganie czy wykrywanie.

- Efektywność powstrzymania fazowego (ang. *Phase Containment Effectiveness*, PCE) – mierzy liczbę defektów wprowadzonych i usuniętych w tej samej fazie w stosunku do całkowitej liczby defektów wprowadzonych w tej fazie. Wysoka wartość PCE wskazuje, że mniej defektów przedostaje się do późniejszych faz.
- Koszt jakości (ang. *Cost of Quality*, CoQ) – ilustruje związek między zapobieganiem defektom, wykrywaniem defektów i kosztami ich usuwania (patrz ISTQB TM, 2024, sekcja 3.2.1).

5.2. Wspieranie powstrzymania fazowego

Celem powstrzymania fazowego (ang. *phase containment*) jest wykrywanie i usuwanie defektów w tej samej fazie cyklu życia oprogramowania, w której zostały one wprowadzone. W zwinnym wytwarzaniu oprogramowania przesunięcie w lewo może być użyte w podobny sposób. Polityka ta zmniejsza koszt jakości. Ponieważ podstawa testów jest ważnym źródłem dla analizy i projektowania testów, AT może najlepiej przyczynić się do powstrzymania fazowego poprzez ocenę jakości podstawy testów. Wczesne zwrócenie uwagi na jej jakość minimalizuje późniejszy wysiłek i zapobiega przenikaniu defektów do kolejnych faz. W niniejszym sylabusie omówiono dwa typowe podejścia AT do znajdowania defektów w podstawie testów: modelowanie do celów testowania oraz przegląd podstawy testów z wykorzystaniem różnych technik przeglądu (ISO/IEC 20246, 2017).

5.2.1. Wykorzystanie modeli do wykrywania defektów

Modelowanie dostarcza abstrakcje systemu na ustalonym poziomie precyzji i szczegółowości. Pomaga interesariuszom lepiej zrozumieć opracowywany system. Jak omówiono poniżej, modelowanie może wspierać powstrzymanie fazowe na co najmniej trzy sposoby:

- poprzez wykrywanie defektów w specyfikacjach,
- poprzez wykrywanie defektów w modelach,
- poprzez wykrywanie defektów w przedmiotach testów przy zastosowaniu testowania opartego na modelu.

Wykrywanie defektów w specyfikacjach. Specyfikacje są często dostarczane jako nieformalnie napisany lub nieustrukturyzowany tekst. AT może reprezentować te specyfikacje za pomocą formalnych modeli. Podczas tworzenia modelu warunki testowe (np. wymagania) są mapowane na elementy modelu i łączone z nimi w celu zapewnienia śledzenia powiązań. Formalizacja i wizualizacja warunków testowych w modelu skutecznie ujawnia defekty, takie jak niekompletność, niespójności lub niejednoznaczności. Siłą modelowania jest to, że specyfikacja jest przekształcana, podczas gdy przeglądy sprawdzają ją w jej oryginalnej formie. W rezultacie AT przyczynia się również do znalezienia odpowiednich rozwiązań dla wykrytych defektów.

Modele oparte na danych obejmują modele dziedzin i modele testowania kombinatorycznego. Umożliwiają one wykrywanie defektów dziedziny, takich jak nakładające się klasy równoważności, luki w pokryciu dziedziny, puste klasy równoważności lub brakujące lub nieprawidłowe kombinacje parametrów.

Modele oparte na zachowaniu obejmują macierze CRUD, modele stanowe lub modele scenariuszy. Pozwalają one AT na wykrywanie niekompletnych lub niespójnych cykli życia encji, brakujących lub wadliwych przejść między stanami, zakleszczeń (ang. *deadlocks*), nieskończonych pętli, niejednoznacznych lub niespójnych zachowań systemu lub brakującej obsługi wyjątków.

Modele oparte na regułach, takie jak tablice decyzyjne lub relacje metamorficzne, umożliwiają AT wykrywanie defektów w regułach biznesowych, takich jak pominięcia, niespójności, niejednoznaczności, nadmiarowości, a także scenariusze podatne na błędy lub złożoną logikę biznesową.

Modelowanie może również pomóc w wykrywaniu defektów, takich jak niespójności w nazewnictwie, wartościach danych (np. wartości brzegowe) oraz danych wejściowych lub wyjściowych. Może również pomóc w identyfikacji brakujących, niekompletnych, niejednoznacznych lub niepotrzebnych informacji.

Wykrywanie defektów w modelach. Jeśli podstawa testów zawiera modele, AT może przeanalizować te modele w celu wykrycia defektów. W niniejszym sylabusie omówiono wykrywanie defektów w trzech typowych modelach wykorzystywanych w testowaniu oprogramowania: diagramach przejść pomiędzy stanami (patrz sekcja 3.2.2), diagramach aktywności (patrz sekcja 3.2.3) i tablicach decyzyjnych (patrz sekcja 3.3.1). Przykłady defektów w wyżej wymienionych modelach obejmują:

- diagramy przejść pomiędzy stanami – brakujące/nieprawidłowe stany, nieprawidłowe przejścia, nieprawidłowe warunki dozoru lub akcje, nadmiarowe lub nieosiągalne stany oraz niedeterministyczne zachowanie,
- diagramy aktywności – brakujące, nieosiągalne lub blokujące akcje, nieprawidłowa kolejność akcji, nieprawidłowe, niewyłączne lub niekompletne warunki dozoru w wierzchołkach decyzyjnych, brakujące punkty synchronizacji lub nieprawidłowo zsynchronizowane przepływy równoległe, które mogą prowadzić do niezamierzonego zachowania,
- tablice decyzyjne – nakładające się, niespójne lub niewykonalne reguły, niekompletność (np. brakujące kombinacje warunków lub brakujące działania dla danej kombinacji warunków).

Wszystkie modele mogą również zawierać defekty takie jak błędy składni, literówki, duplikaty i niespójne nazewnictwo elementów modelu.

Wykrywanie defektów przy zastosowaniu testowania opartego na modelu (ang. *Model-Based Testing*, MBT). W tym podejściu do testowania narzędzie MBT projektuje i generuje przypadki testowe oraz inne testalia w oparciu o model MBT i kryteria wyboru testów zdefiniowane przez AT. MBT jest skuteczne w znajdowaniu anomalii w specyfikacji, ponieważ pozwala na kompleksowe pokrycie i systematyczną eksplorację oczekiwanego zachowania przedmiotu testów zgodnie z modelem MBT. Modele MBT, zwłaszcza te graficzne, sprzyjają komunikacji z interesariuszami, tworząc wspólną percepcję i zrozumienie podstawy testów. Więcej szczegółów na temat MBT można znaleźć w (ISTQB MBT, 2024).

5.2.2. Stosowanie technik przeglądu

Dobrze zdefiniowana podstawa testów zapewnia jakość produktów pracy i powodzenie projektów. Przeglądanie podstawy testów pomaga wcześnie identyfikować i usuwać defekty, zapobiegając ich przenikaniu do kolejnych faz.

Podczas przeglądu indywidualnego, w celu zidentyfikowania defektów w podstawie testów, AT może zastosować różne techniki przeglądu. Wybór najbardziej odpowiedniej techniki przeglądu może zwiększyć wydajność i skuteczność przeglądu. Wybór ten powinien uwzględniać takie czynniki jak cele przeglądu, cele projektu, dostępne zasoby, typ podstawy testów, powiązane ryzyka, dziedzinę biznesową i kulturę firmy.

Niniejszy sylabus omawia pięć technik przeglądu powszechnie stosowanych przez AT.

Przeglądanie ad hoc (ang. *ad hoc reviewing*) jest przeprowadzane przez przeglądających w sposób nieformalny, bez ustrukturyzowanego procesu. Przeglądający otrzymują niewiele lub nie otrzymują żadnych wskazówek dotyczących sposobu wykonywania przeglądu. Przegląd ad hoc wymaga niewielkiego przygotowania i w dużym stopniu zależy od umiejętności przeglądających. Podczas przeglądu przeglądający czytają podstawę testów i dokumentują anomalie, gdy je napotkają. Ta technika, pozostawiona bez zarządzania, może prowadzić do dużej liczby zduplikowanych raportów o anomaliach od wielu przeglądających.

Przeglądanie oparte na listach kontrolnych (ang. *checklist-based reviewing*) obejmuje ocenę podstawy testów na podstawie wcześniej zdefiniowanej listy kontrolnej. Listy kontrolne przypominają przeglądającym o sprawdzeniu określonych punktów i pomagają zdepersonalizować przegląd. Listy kontrolne mogą być ogólne lub specyficzne dla konkretnych charakterystyk jakościowych, celów testów lub typu podstawy testów. AT dostosowuje listę kontrolną do typu podstawy testów, poziomu ryzyka lub warunków testowych. Gwarantuje to, że przegląd skupi się na najistotniejszych aspektach podstawy testów. Listy kontrolne powinny być regularnie aktualizowane o wcześniej pominięte defekty. Aktualizowanie list kontrolnych zapobiega przeoczeniu nowo zidentyfikowanych anomalii. Listy kontrolne nie są wyczerpujące, dlatego też AT nie jest ograniczony do sprawdzania wyłącznie pozycji wymienionych w liście kontrolnej. Maksymalizuje to wykrywanie defektów i pozwala na wychwycenie anomalii, których listy kontrolne mogą nie obejmować.

Przeglądanie oparte na scenariuszach (ang. *scenario-based reviewing*) obejmuje symulację procesu lub działania w celu zidentyfikowania anomalii i udoskonalenia podstawy testów. Ta technika przeglądu jest najbardziej skuteczna, gdy podstawa testów ma format oparty na scenariuszu, taki jak przypadek użycia lub diagram aktywności. W takich przypadkach przeglądający mogą przeprowadzać „suche przebiegi" (ang. *dry runs*) w oparciu o oczekiwane wykorzystanie produktu pracy. Scenariusze zapewniają cenne wytyczne, ale przeglądający nie są ograniczeni do udokumentowanych scenariuszy i powinni również w swoim myśleniu wykraczać poza zadane scenariusze, aby zidentyfikować więcej anomalii.

Przeglądanie oparte na rolach (ang. *role-based reviewing*) polega na przypisywaniu określonych ról lub obowiązków przeglądającym. Typowe role opierają się na określonych typach użytkowników końcowych (np. doświadczony, niedoświadczony, osoba starsza, dziecko) lub określonych rolach w organizacji (np. administrator lub zwykły użytkownik). Każdą rolę można opisać za pomocą tzw. persony (tj. konkretnej, ale fikcyjnej postaci zaprojektowanej w celu

reprezentowania cech, potrzeb, celów i preferencji określonej grupy użytkowników). Rozdzielając obowiązki pomiędzy przeglądających na podstawie ich ról, przeglądy oparte na rolach pozwalają poszczególnym osobom skupić się na określonych aspektach podstawy testów, zapewniając kompleksowe pokrycie, a jednocześnie unikając zgłaszania powielonych anomalii.

Czytanie oparte na perspektywie (ang. *perspective-based reading*) obejmuje przeglądanie podstawy testów z różnych perspektyw lub punktów widzenia (np. projektanta, testera, marketera, administratora, użytkownika końcowego). Prowadzi to do bardziej dogłębnego indywidualnego przeglądu z mniejszym ryzykiem zgłaszania powielonych anomalii wśród przeglądających. Ponadto czytanie oparte na perspektywie wymaga od przeglądających podjęcia próby wykorzystania analizowanej podstawy testów do wygenerowania produktu pracy, który powinni z niej uzyskać. Przykładowo, tester może próbować wygenerować wstępną wersję testów akceptacyjnych w oparciu o specyfikację wymagań, aby sprawdzić, czy zawiera ona wszystkie niezbędne informacje.

5.3. Łagodzenie ponownego występowania defektów

AT może proaktywnie pomóc minimalizować ponowne występowanie defektów w oprogramowaniu. Niniejszy sylabus omawia dwa podejścia związane z tym działaniem: analizowanie wyników testów w celu udoskonalenia analizy i projektowania testów oraz wspieranie analizy przyczyny podstawowej za pomocą klasyfikacji defektów.

5.3.1. Analiza wyników testów w celu udoskonalenia wykrywania defektów

Wyniki testów pozwalają AT identyfikować awarie, ale także dostarczają informacji zwrotnych, które pomagają AT poprawić skuteczność wykrywania defektów. Poniżej opisano kilka powszechnie stosowanych technik analizy wyników testów.

Analiza przewidywanych vs. rzeczywistych klastrów defektów. Mała liczba komponentów zwykle zawiera większość defektów (patrz ISTQB CTFL, podrozdział 1.3). Po wykonaniu testów, AT może przewidzieć obszary podatne na defekty i porównać przewidywane klastry defektów z rzeczywistymi. W przypadku rozbieżności można zastosować bardziej rygorystyczne testy w obszarach, w których znaleziono więcej defektów niż oczekiwano. Podczas określania klastrów mierzalne kryteria powinny zapewniać jasność i spójność (np. gęstość defektów i ich krytyczność). Wśród tych kryteriów istotną rolę powinna odgrywać krytyczność defektów. Małe skupiska krytycznych lub poważnych defektów są zwykle ważniejsze (tj. wymagają bardziej rygorystycznych testów) niż większe skupiska drobnych lub kosmetycznych defektów.

Analiza odsetka wykrytych defektów (ang. *Defect Detection Percentage*, DDP). DDP jest jedną z najważniejszych miar skuteczności dla poziomu testów. Podczas obliczania DDP dla danego poziomu testów liczba defektów, które nie zostały wykryte, powinna być ograniczona do tych, które mogły zostać wykryte na danym poziomie testów. Aby zapewnić spójność pomiaru, należy ustalić wyraźne granice liczenia defektów, takie jak limity czasowe (np. defekty wykryte w określonych ramach czasowych po wydaniu) i kryteria wykluczenia (np. nieuwzględnianie defektów w komponentach innych firm lub w określonych środowiskach klienta). Niska wartość DDP dla danego poziomu testów wskazuje na wysoki odsetek uniknięty defektów, co oznacza, że wykrywanie defektów na tym poziomie testów jest nieskuteczne. W takim przypadku AT powinien przeanalizować przyczyny i zaproponować środki zaradcze, dzięki czemu poziom testów

będzie bardziej ukierunkowany i rygorystyczny. DDP najlepiej jest podzielić według poziomów krytyczności, ponieważ priorytet redukcji defektów przenikających do kolejnych faz zwykle zależy od krytyczności tych defektów.

Analiza pokrycia strukturalnego ocenia zakres, w jakim testy pokryły określone obszary przedmiotu testów. Identyfikacja obszarów o niskim pokryciu pozwala AT ukierunkować wysiłki testowe na te obszary. Zwiększenie ich pokrycia pomaga odkryć nowe, wcześniej niezauważone defekty. Pokrycie strukturalne, takie jak pokrycie instrukcji lub gałęzi, jest zwykle mierzone za pomocą narzędzi testowych. Wybierając dodatkowe obszary do pokrycia, należy wziąć pod uwagę ich poziomy ryzyka.

Analiza luk testowych ocenia zakres, w jakim testy sprawdzały ostatnie zmiany w kodzie. Dzięki temu AT może skoncentrować dodatkowe wysiłki testowe na obszarach szczególnie podatnych na błędy (tj. na nowych zmianach, które nie zostały w ogóle przetestowane) zamiast na wszystkich obszarach o niskim pokryciu (np. kodzie, który nie zmienił się od dłuższego czasu i został przetestowany w poprzednich wersjach).

Analiza wzorca pojawiania się defektów. Liczbę lub gęstość defektów wykrytych w kolejnych fazach projektu (np. iteracjach) można porównać z wzorcami opisującymi teoretyczny rozkład tych wartości w czasie. Klasycznym przykładem wzorca pojawiania się defektów jest model Rayleigha (Elsayed, 2021). Ma on pojedyncze ekstremum i jest prawoskośny, co pokazuje, że oczekiwana liczba znalezionych defektów najpierw rośnie w czasie, a po osiągnięciu maksymalnej wartości powoli spada w kierunku zera. Na podstawie analizy takiego wzorca można wywnioskować siłę istniejących przypadków testowych i potencjał ich udoskonalenia. Na przykład, jeśli wykrywalność defektów pozostaje na stałym, niskim poziomie, podczas gdy wzorzec sugeruje, że powinna ona rosnąć, może to oznaczać, że istniejące testy są zbyt słabe i nie są w stanie wykryć dodatkowych defektów.

Opisane powyżej metody analityczne wykorzystują różne metryki związane z defektami, takie jak liczba defektów lub DDP. Metryki te są obliczane na podstawie wyników testów (np. liczba testów zaliczonych/niezaliczonych), raportów defektów i metryk strukturalnych (np. pokrycie kodu lub gęstość defektów). Należy jednak pamiętać, że wskaźniki te nie zawsze są tak łatwe do odczytania z wyników testów, jak mogłoby się wydawać. Przykładowo, liczba niezaliczonych testów niekoniecznie jest tożsama z liczbą defektów wykrytych przez te testy. Aby obliczyć rzeczywistą liczbę wykrytych defektów, należy dokładnie przeanalizować wyniki procesu debugowania, ponieważ relacja między wynikami testów a defektami może być typu wiele do wielu. Kilka testów może wykryć ten sam defekt lub jeden test może wykryć kilka defektów. Co więcej, krytyczność defektów może różnić się od krytyczności testów, ponieważ krytyczny przypadek testowy może zostać niezaliczony z powodu kosmetycznego defektu.

5.3.2. Wspieranie analizy przyczyny podstawowej przez klasyfikację defektów

Analiza przyczyny podstawowej (ang. *Root Cause Analysis*, RCA) to technika służąca do identyfikowania i eliminowania podstawowych przyczyn defektu, a nie tylko jego objawów. RCA wspiera ustrukturyzowane podejście do doskonalenia jakości, a jej głównym celem jest zapobieganie ponownemu wystąpieniu defektów. AT wykorzystuje różne techniki do identyfikacji przyczyn podstawowych defektów i awarii (np. taksonomie defektów, technika pięciu pytań „dlaczego” [ang. *five-whys*], diagramy przyczynowo-skutkowe, analiza Pareto).

W klasycznej RCA eksperci merytoryczni szczegółowo badają defekt po jego usunięciu. Jednak zazwyczaj istnieje wiele defektów, które należy przeanalizować. Dlatego też planowanie działań zapobiegawczych dla każdego defektu byłoby bardzo nieefektywne i czasochłonne. Jednym ze sposobów podejścia do tego problemu jest sklasyfikowanie defektów, a następnie przeprowadzenie RCA dla *typów* zidentyfikowanych defektów.

Klasyfikacja defektów (ang. *defect classification*) opiera się na uznaniu, że poszczególne defekty zawierają wiele informacji na temat procesu rozwoju i systemu podlegającego testowaniu. Klasyfikacja defektów pozwala AT na wyodrębnienie informacji o różnych aspektach procesu rozwoju z defektu i przekształcenie ich w pomiar procesu. To z kolei daje wgląd w rodzaje błędów popełnianych podczas prac wytwórczych, co jest pomocne w doskonaleniu procesu. Klasyfikacja defektów wypełnia lukę między ilościowymi statystykami defektów a jakościową RCA. Aby skutecznie wspierać RCA, defekty powinny być klasyfikowane jednolicie w całym cyklu życia oprogramowania, od wczesnych testów po produkcję.

AT powinien wspierać swoją organizację w standaryzacji klasyfikacji defektów oprogramowania. Usprawnia to komunikację i wymianę informacji dotyczących defektów między programistami i organizacjami, ułatwiając przeprowadzenie RCA.

Przykłady metod klasyfikacji defektów to:

- ortogonalna klasyfikacja defektów (ang. *Orthogonal Defect Classification*, ODC) (Chillarege, 1992), która klasyfikuje każdy defekt względem ośmiu ortogonalnych (tj. wzajemnie wykluczających się) atrybutów, gromadzonych zarówno po zgłoszeniu defektu, jak i po jego usunięciu,
- norma IEEE 1044, IEEE *Standard Classification for Software Anomalies* (IEEE 1044, 2009), klasyfikacja anomalii oprogramowania zapewniająca podstawowy zestaw atrybutów do klasyfikacji awarii i defektów,
- klasyfikacja oparta na krytyczności (ang. *severity-based classification*), która klasyfikuje defekty na podstawie ich krytyczności (np. krytyczne, poważne, drobne, trywialne),
- modele taksonomii defektów (ang. *defect taxonomies*), takie jak (Beizer, 1990) lub (Catolino *et al.*, 2019).

Defekty mogą być również klasyfikowane ze względu na związane z nimi atrybuty jakości opisane w modelach jakości oprogramowania, takich jak (ISO/IEC 25010, 2023) lub model FURPS (Grady *et al.*, 1987).

Więcej informacji na temat RCA można znaleźć w sylabusie (ISTQB ITP, 2011).

6. Bibliografia

Sylabusy ISTQB®

ISTQB AcT, ISTQB®, 2019. *Certified Tester Specialist Mobile Acceptance Testing*, wersja 1.0.

ISTQB AI, ISTQB®, 2021. *Certified Tester AI Testing*, wersja 1.0.

ISTQB CTFL, ISTQB®, 2024. *Certified Tester Foundation Level*, wersja 4.0.1.

ISTQB ITP, ISTQB®, 2011. *Certified Tester Expert Level Improving the Test Process*, wersja 1.0.

ISTQB MAT, ISTQB®, 2019. *Certified Tester Specialist Mobile Application Testing*, wersja 1.0.

ISTQB MBT, ISTQB®, 2024. *Certified Tester Model-Based Tester*, wersja 1.1.

ISTQB PT, ISTQB®, 2018. *Certified Tester Performance Testing*, wersja 1.0.

ISTQB TAE, ISTQB®, 2024. *Certified Tester Test Automation Engineering*, wersja 2.0.

ISTQB TM, ISTQB®, 2024. *Certified Tester Advanced Level Test Management*, wersja 3.0.

ISTQB TTA, ISTQB®, 2021. *Certified Tester Advanced Level Technical Test Analyst*, wersja 4.0.

ISTQB UT, ISTQB®, 2018. *Certified Tester Usability Testing*, wersja 1.0.

(*Polskie wersje niektórych z powyższych sylabusów znajdują się na stronie www.pqb.org.pl*)

Słowniki pojęć

ISTQB® Glossary, ISTQB®, 2024 Standard *Glossary of Terms Used in Software Testing* [online]. [data dostępu 15.09.2024]. https://glossary.istqb.org.

Agile Alliance, www.agilealliance.org/agile-practice-guide/

IEEE-Pascal, *IEEE Pascal for software engineering*, pascal.computer.org

IREB-CPRE, *Glossary for Requirements Engineering*, www.ireb.org/en/cpre/glossary

Normy

ETSI EG 202 237 v1.2.1: *Internet Protocol Testing* (IPT), 2010. European Telecommunications Standards Institute.

IEEE 1044: *Standard Classification for Software Anomalies*, 2009. Institute of Electrical and Electronics Engineers.

ISO 15745: *Industrial automation systems and integration – Open systems application integration framework*, 2003. International Organization for Standardization.

ISO 16100: *Industrial automation systems and integration – Manufacturing software capability profiling for interoperability*, 2009. International Organization for Standardization.

ISO 26262: *Road vehicles – functional safety – Part 6: Product development at the software level*, 2018. International Organization for Standardization.

ISO 9241-210: *Ergonomics of human-system interaction, part 210: Human-centred design for interactive systems*, 2019. International Organization for Standardization.

ISO/IEC 20246: *Software and systems engineering – Work product reviews*, 2017. International Organization for Standardization.

ISO/IEC 25010: *Systems and software Quality Requirements and Evaluation (SQuaRE) – Product quality model*, 2023. International Organization for Standardization.

ISO/IEC 25019: *Systems and software Quality Requirements and Evaluation (SQuaRE) – Quality-in-use model*, 2023. International Organization for Standardization.

ISO/IEC/IEEE 29119-3: *Software testing, Part 3: Test documentation*, 2021. International Organization for Standardization.

ISO/IEC/IEEE 29119-4: *Software testing, Part 4: Test techniques*, 2021. International Organization for Standardization.

ISO/IEC/IEEE 29119-5: *Software testing, Part 5: Keyword-Driven Testing*, 2016. International Organization for Standardization.

OMG BPMN: *Business Process Model and Notation*, *version 2.0*, 2013. Object Management Group, OMG®.

OMG DMN: *Decision Model and Notation, version 1.5*, 2024. 2024-01. Object Management Group.

OMG UML: *Unified Modeling Language, version 2.5*, 2017. Object Management Group.

RTCA DO-178C: *Software Considerations in Airborne Systems and Equipment Certification*, 2011. Radio Technical Commission for Aeronautics, RTCA Inc.

Książki

AMMANN, Paul; OFFUTT, Jeff, 2008. *Introduction to Software Testing*. Cambridge University Press.

BEIZER, Boris, 1990. *Software Testing Techniques (2nd Ed.)* International Thomson Computer Press.

BINDER, Robert V., 2000. *Testing Object-Oriented Systems*. Addison-Wesley.

ELSAYED, Elsayed A., 2021. *Reliability Engineering*. Wiley.

FORGÁCS, Istvan; KOVÁCS, Attila, 2019. *Practical Test Design: Selection of traditional and automated test design techniques*. BCS, The Chartered Institute for IT.

FORGÁCS, Istvan; KOVÁCS, Attila, 2024. *Modern Software Testing Techniques*. Apress.

GRADY, Robert; CASWELL, Deborah, 1987. *Software Metrics: Establishing a Company-wide Program*. Prentice Hall.

HENDRICKSON, Elisabeth, 2013. *Explore It!* Pragmatic Bookshelf.

JORGENSEN, Paul, 2014. *Software Testing. A Craftsman's Approach*. CRC Press.

KANER, Cem; FALK, Jack; NGUYEN, Hung Q., 1999. *Testing Computer Software*. Wiley.

KANER, Cem; PADMANABHAN, Sowmya; HOFFMAN, Douglas, 2013. *The Domain Testing Workbook*. Context Driven Press.

KOOMEN, Tim; AALST, Leo van der; BROEKMAN, Bart; VROON, Michiel, 2006. *TMap Next for result-driven testing*. UTN Publishers.

KUHN, D. Richard; YU, Lei; KACKER, Raghu N., 2013. *Introduction to Combinatorial Testing*. CRC Press.

Artykuły

ALYAHYA, Sultan, 2020. Crowdsourced Software Testing: A Systematic Literature Review. *Information and Software Technology*. Vol. 127, str. 106363. DOI: 10.1016/j.infsof.2020.106363.

ANTONIOL, Giuliano; BRIAND, Lionel; DI PENTA, Massimiliano; LABICHE, Yvan, 2002. A case study using the round-trip strategy for state-based class testing, *Proceeding 13th International Symposium on Software Reliability Engineering,* str. 269–279. ISBN 0-7695-1763-3. DOI: 10.1109/ISSRE.2002.1173268.

ARCURI, Andrea; IQBAL, Muhammad Zohaib; BRIAND, Lionel, 2012. Random Testing: Theoretical Results and Practical Implications. IEEE *Transactions on Software Engineering* – TSE. Vol. 38, str. 258–277. DOI: 10.1109/TSE.2011.121.

BARR, Earl; HARMAN, Mark; MCMINN, Phil; SHAHBAZ, Muzammil; YOO, Shin, 2014. The Oracle Problem in Software Testing: A Survey. *IEEE Transactions on Software Engineering*. Vol. 41, no. 5, str. 507–525. DOI: 10.1109/TSE.2014.2372785.

BOCHMANN, Gregor; PETRENKO, Alexandre, 1994. Protocol Testing: Review of Methods and Relevance for Software Testing. ISSTA '94: *Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis,* str. 109–124. DOI: 10.1145/186258.187153.

CATOLINO, Gemma; PALOMBA, Fabio; ZAIDMAN, Andy; FERRUCCI, Filomena, 2019. Not All Bugs Are the Same: Understanding, Characterizing, and Classifying Bug Types. *Journal of Systems and Software*. Vol. 152, str. 165–181. DOI: 10.1016/j.jss.2019.03.002.

CHILLAREGE, R. *et al.*, 1992. Orthogonal Defect Classification – A Concept for In-Process Measurements. Software Engineering, *IEEE Transactions on Software Engineering*. Vol. 18, str. 943–956. DOI: 10.1109/32.177364.

CHOW, Tsun, 1978. Testing Software Design Modeled by Finite-State Machines. *IEEE Transactions on Software Engineering*. Vol. 4, pp. 178–187. DOI: 10.1109/TSE.1978.231496.

COHEN, D.M.; DALAL, S.; KAJLA, A.; PATTON, G.C., 1994. The Automatic Efficient Test Generator (AETG) system. *Proceedings of 1994 IEEE International Symposium on Software Reliability Engineering*, str. 303–309. ISBN 0-8186-6665-X. DOI: 10.1109/ISSRE.1994.341392.

ENGSTRÖM, Emelie; RUNESON, Per; SKOGLUND, Mats, 2010. A systematic review on regression test selection techniques. *Information and Software Technology*. Vol. 52, str. 14–30. DOI: 10.1016/j.infsof.2009.07.001.

GHAZI, Ahmad Nauman; GARIGAPATI, Ratna; PETERSEN, Kai, 2017. Checklists to Support Test Charter Design in Exploratory Testing. *Agile Processes in Software Engineering and Extreme Programming. XP 2017*. ISBN 978-3-319-57632-9. DOI: 10.1007/978-3-319-57633-6_17.

HAREL, David, 1987. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*. Vol. 8, str. 231–274. DOI: 10.1016/0167-6423(87)90035-9.

HUANG, Rubing; SUN, Weifeng; XU, Yinyin; CHEN, Haibo; TOWEY, Dave; XIA, Xin, 2019. A Survey on Adaptive Random Testing. *IEEE Transactions on Software Engineering*. Vol. 47, no. 10, str. 2052–2083. DOI: 10.1109/TSE.2019.2942921.

JENG, Bingchiang; WEYUKER, Elaine, 1994. A Simplified Domain-Testing Strategy. *ACM Transactions on Software Engineering and Methodology*. Vol. 3, str. 254–270. DOI: 10.1145/196092.193171.

JUERGENS, Elmar; PAGANO, Dennis; GOEB, Andreas, 2018. Test Impact Analysis: Detecting Errors Early Despite Large, Long-Running Test Suites. *Whitepaper, CQSE GmbH*.

KUHN, D.; WALLACE, Dolores; JR, A.M., 2004. Software Fault Interactions and Implications for Software Testing. *IEE Transactions on Software Engineering*. Vol. 30, str. 418–421. DOI: 10.1109/TSE.2004.24.

LEICHT, Niklas; BLOHM, Ivo; LEIMEISTER, Jan Marco, 2017. Leveraging the Power of the Crowd for Software Testing. *IEEE Software*. Vol. 34, str. 62–69. DOI: 10.1109/MS.2017.37.

OFFUTT, A. Jefferson, 1992. Investigations of the software testing coupling effect. *ACM Transactions on Software Engineering Methodology*. Vol. 1, str. 5–20.

RECHTBERGER, Vaclav; BURES, Miroslav; AHMED, Bestoun, 2022. Overview of Test Coverage Criteria for Test Case Generation from Finite State Machines Modelled as Directed Graphs. *IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, Valencia, Spain, str. 207–214.

RWEMALIKA, Renaud; KINTIS, Marinos; PAPADAKIS, Mike; LE TRAON, Yves; LORRACH, Pierre, 2019. On the Evolution of Keyword-Driven Test Suites. *12th IEEE Conference on Software Testing, Validation and Verification (ICST)*, str. 335–345.

SEGURA, Sergio; FRASER, Gordon; SANCHEZ, Ana; RUIZ-CORTÉS, Antonio, 2016. A Survey on Metamorphic Testing. *IEEE Transactions on Software Engineering*, str. 46–53.

SEGURA, Sergio; TOWEY, Dave; ZHOU, Zhi Quan; CHEN, Tsong Yueh, 2020. Metamorphic Testing: Testing the Untestable. *IEEE Software*. Vol. 42, no. 9, str. 805–824.

WU, Huayao; NIE, Changhai; PETKE, Justyna; JIA, Yue; HARMAN, Mark, 2020. *An Empirical Comparison of Combinatorial Testing, Random Testing and Adaptive Random Testing. IEEE Transactions on Software Engineering*. Vol. 46, no. 3, str. 302–320.

Źródła internetowe

ADA, 2010. U.S. Department of Justice, Civil Rights Division. Americans with Disabilities Act: Guidance & Resource Materials [online]. [data dostępu 15.09.2024], www.ada.gov/resources/.

CZERWONKA, Jacek, 2004. *Pairwise Testing: Combinatorial Test Case Generation* [online]. [data dostępu 15.09.2024], www.pairwise.org.

EA, 2010. *United Kingdom's Equality Act: Information and guidance on the Equality Act* [online]. [data dostępu 15.09.2024], www.gov.uk/guidance/equality-act-2010-guidance.

GDPR, 2016. Komisja Europejska. *General Data Protection Regulation: Regulation (EU) 2016/679* [online]. [data dostępu 15.09.2024], https://commission.europa.eu/law/law-topic/data-protection/legal-framework-eu-data-protection_en?.

HIPAA, 1996. U.S. Department of Health and Human Services. *Insurance Portability and Accountability Act: Standards for Privacy of Individually Identifiable Health Information (Privacy Rules)* [online]. [data dostępu 15.09.2024], https://www.hhs.gov/hipaa/for-professionals/privacy/laws-regulations/index.html.

SSTD, 2020. *Site of Software Test Design* [online]. [data dostępu 15.09.2024], https://test-design.org/.

SUMI, 1991. *Software Usability Measurement Inventory: Standard evaluation questionnaire for assessing quality of use* [online]. [data dostępu 15.09.2024], sumi.uxp.ie.

WAMMI, 1999. *Website Analysis and Measurement Inventory: Professional service for analyzing user experience* [online]. [data dostępu 15.09.2024], www.wammi.com.

WCAG, 2023. *Web Content Accessibility Guidelines 2.2: W3C Recommendation* [online]. [data dostępu 15.09.2024], www.w3.org/TR/WCAG22/.

Powyższe odniesienia wskazują na informacje dostępne w Internecie i innych miejscach. Mimo że odniesienia te zostały sprawdzone w momencie publikacji niniejszego sylabusa, ISTQB® nie ponosi odpowiedzialności, jeśli odniesienia te są już niedostępne.

7. Dodatek A – cele nauczania i poziomy wiedzy

Szczegółowe cele nauczania dotyczące niniejszego sylabusa są przedstawione na początku każdego rozdziału. Każdy temat opisany w sylabusie będzie objęty egzaminem zgodnie z jego celem nauczania. Cele nauczania rozpoczynają się od czasownika określającego czynność odpowiadającą wymaganemu poziomowi wiedzy, zgodnie z poniższym wykazem.

Poziom 1: zapamiętać (K1)

Kandydat pamięta, rozpoznaje lub przypomina sobie dany termin lub dane pojęcie.

Czasowniki określające czynność: przypomnieć sobie, rozpoznać.

Uwaga: sylabusy na poziomie zaawansowanym nie zawierają konkretnych celów nauczania na poziomie K1. Jednakże treść sylabusa zawarta w rozdziałach 1–5 oraz definicje wszystkich terminów wymienionych jako słowa kluczowe tuż pod nagłówkami rozdziałów powinny zostać zapamiętane (K1), nawet jeśli nie zostały one wyraźnie wymienione w celach nauczania.

Poziom 2: zrozumieć (K2)

Kandydat potrafi uzasadnić lub wyjaśnić stwierdzenia dotyczące danego zagadnienia, a także podsumować, porównać, sklasyfikować i podać przykłady pojęć z zakresu testowania.

Czasowniki określające czynność: sklasyfikować, porównać, rozróżniać, odróżnić, wyjaśnić, podać przykłady, zinterpretować, podsumować.

Przykłady	Uwagi
Rozróżniać testowanie poprawności funkcjonalnej, adekwatności funkcjonalnej i kompletności funkcjonalnej	Szuka różnic pomiędzy pojęciami
Podać przykłady wymagań dla środowiska testowego	.
Podsumować zaangażowanie analityka testów w różnych modelach wytwarzania oprogramowania	

Poziom 3: zastosować (K3)

Kandydat potrafi wykonać odpowiednią procedurę, gdy zostanie mu postawione znane mu zadanie bądź wybrać poprawną procedurę i zastosować ją w danym kontekście.

Czasowniki określające czynność: zastosować, wdrożyć, przygotować, użyć.

Przykłady	Uwagi
Zastosować testowanie dziedziny	Powinno odnosić się do procedury, techniki, procesu itp.
Przygotować karty opisu testu dla testowania w sesjach	
Użyć modelu przedmiotu testów w celu wykrywania defektów w specyfikacji	Może być użyte w celu nauczania, który każe kandydatowi użyć techniki lub procedury. Podobne do ‘zastosować’

Poziom 4: przeanalizować (K4)

Kandydat potrafi rozdzielić informacje związane z procedurą lub techniką na poszczególne części składowe w celu lepszego zrozumienia oraz potrafi odróżnić fakty od wniosków. Typowym zastosowaniem jest analiza dokumentu, oprogramowania lub sytuacji projektowej oraz zaproponowanie odpowiednich działań w celu rozwiązania problemu lub wykonania zadania.

Czasowniki: określające czynność: przeanalizować, zdekonstruować, nakreślić, ustalić priorytety, wybrać.

Przykłady	Uwagi
Przeanalizować wpływ zmian w celu określenia zakresu testowania regresji	Podlegające egzaminowaniu tylko w połączeniu z mierzalnym celem analizy. Powinno mieć formę "przeanalizuj X w celu Y" (lub podobną)
Wybrać odpowiednią technikę testowania w celu złagodzenia ryzyk produktowych	Potrzebne, kiedy wybór wymaga analizy

Materiały dodatkowe w zakresie poziomów poznawczych celów nauczania:

Anderson, L. W., Krathwohl, D. R. (red.), 2001, *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*, Allyn & Bacon.

8. Dodatek B – macierz powiązań między celami biznesowymi a celami nauczania

Cele biznesowe – poziom zaawansowany analityk testów		BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8	BO9
BO1	Wspiera i przeprowadza odpowiednie testy w ramach cyklu wytwarzania oprogramowania	6								
BO2	Stosuje podejście do testowania oparte na ryzyku		3							
BO3	Wybiera i stosuje odpowiednie techniki testowania, aby wspierać osiąganie celów testów			13						
BO4	Zapewnia dokumentację o odpowiednim poziomie szczegółowości i jakości				5					
BO5	Określa odpowiednie rodzaje testów funkcjonalnych, które należy przeprowadzić					1				
BO6	Bierze udział w testach niefunkcjonalnych						3			
BO7	Wnosi wkład w zapobieganie defektom							5		
BO8	Poprawia wydajność procesu testowego, w tym przy użyciu narzędzi								4	
BO9	Określa wymagania dotyczące środowisk testowych i danych testowych									2

Cele nauczania – poziom zaawansowany analityk testów	Poziom K	BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8	BO9
1. Zadania analityka testów w procesie testowym (225 min.)										
1.1 Testowanie w cyklu wytwarzania oprogramowania										
TA-1.1.1 Podsumować zaangażowanie analityka testów w różnych modelach wytwarzania oprogramowania	K2	X								
1.2 Zaangażowanie w aktywności testowe										
TA-1.2.1 Podsumować zadania wykonywane przez analityka testów w ramach analizy testów	K2	X								
TA-1.2.2 Podsumować zadania wykonywane przez analityka testów w ramach projektowania testów	K2	X								
TA-1.2.3 Podsumować zadania wykonywane przez analityka testów w ramach implementacji testów	K2	X								
TA-1.2.4 Podsumować zadania wykonywane przez analityka testów w ramach wykonywania testów	K2	X								
1.3 Zadania związane z testaliami										
TA-1.3.1 Rozróżniać przypadki testowe wysokiego i niskiego poziomu	K2				X					
TA-1.3.2 Wyjaśnić kryteria jakości dla przypadków testowych	K2				X					
TA-1.3.3 Podać przykłady wymagań dla środowiska testowego	K2				X					X
TA-1.3.4 Wyjaśnić problem wyroczni testowej i jego możliwe rozwiązania	K2			X	X					
TA-1.3.5 Podać przykłady wymagań dla danych testowych	K2				X					X

Cele nauczania – poziom zaawansowany analityk testów	Poziom K	BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8	BO9
TA-1.3.6 Zastosować testowanie oparte na słowach kluczowych, aby opracować skrypty testowe	K3	X							X	
TA-1.3.7 Podsumować typy narzędzi do zarządzania testaliami	K2								X	
2. Zadania analityka testów w testowaniu opartym na ryzyku (90 min.)										
2.1 Analiza ryzyka										
TA-2.1.1 Podsumować wkład AT w analizę ryzyka produktowego	K2		X							
2.2 Kontrola ryzyka										
TA-2.2.1 Przeanalizować wpływ zmian w celu określenia zakresu testów regresji	K4		X						X	
3. Analiza i projektowanie testów (615 min.)										
3.1 Techniki testowania oparte na danych										
TA-3.1.1 Zastosować testowanie dziedziny	K3			X						
TA-3.1.2 Zastosować testowanie kombinatoryczne	K3			X						
TA-3.1.3 Podsumować korzyści i ograniczenia testowania losowego	K2			X						
3.2 Techniki testowania oparte na zachowaniu										
TA-3.2.1 Wyjaśnić testowanie CRUD	K2			X						
TA-3.2.2 Zastosować testowanie przejść pomiędzy stanami	K3			X						
TA-3.2.3 Zastosować testowanie oparte na scenariuszach	K3			X						
3.3 Techniki testowania oparte na regułach										
TA-3.3.1 Zastosować testowanie w oparciu o tablice decyzyjne	K3			X						
TA-3.3.2 Zastosować testowanie metamorficzne	K3			X						
3.4 Techniki testowania oparte na doświadczeniu										
TA-3.4.1 Przygotować karty opisu testów dla testowania w sesjach	K3			X						
TA-3.4.2 Przygotować listy kontrolne wspierające testowanie oparte na doświadczeniu	K3			X						
TA-3.4.3 Podać przykłady korzyści i ograniczeń testowania w tłumie	K2			X						
3.5 Zastosowanie najlepszych technik testowania										
TA-3.5.1 Wybrać właściwe techniki testowania w celu łagodzenia ryzyk produktowych	K4		X	X						
TA-3.5.2 Wyjaśnić korzyści i ryzyka związane z automatyzacją projektowania testów	K2								X	
4. Testowanie charakterystyk jakościowych (60 min.)										
4.1 Testowanie funkcjonalności										
TA-4.1.1 Rozróżniać testowanie poprawności, adekwatności i kompletności funkcjonalnej	K2					X				
4.2 Testowanie użyteczności										
TA-4.2.1 Wyjaśnić wkład AT w testowaniu użyteczności	K2						X			

Cele nauczania – poziom zaawansowany analityk testów	Poziom K	BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8	BO9
4.3 Testowanie elastyczności (przenaszalności)										
TA-4.3.1 Wyjaśnić wkład AT w testowanie adaptowalności i instalowalności	K2						X			
4.4 Testowanie zgodności (kompatybilności)										
TA-4.4.1 Wyjaśnić wkład AT w testowanie współdziałania	K2						X			
5. Zapobieganie defektom (225 min.)										
5.1 Praktyki zapobiegania defektom										
TA-5.1.1 Wyjaśnić wkład AT w zapobieganie defektom	K2							X		
5.2 Wspieranie powstrzymania fazowego										
TA-5.2.1 Użyć modelu przedmiotu testów w celu wykrywania defektów w specyfikacji	K3							X		
TA-5.2.2 Zastosować technikę przeglądu do podstawy testów w celu znalezienia defektów	K3							X		
5.3 Łagodzenie ponownego występowania defektów										
TA-5.3.1 Przeanalizować wyniki testów w celu identyfikacji udoskonaleń w wykrywaniu defektów	K4							X		
TA-5.3.2 Wyjaśnić, jak klasyfikacja defektów wspiera analizę przyczyny podstawowej	K2							X		

9. Dodatek C – nota wydania

Wersja 4.0 sylabusa ISTQB® Certyfikowany Tester Poziom Zaawansowany Analityk Testów to znacząca aktualizacja. Z tego powodu nie ma szczegółowych informacji o zmianach w poszczególnych rozdziałach i sekcjach. Poniżej znajduje się podsumowanie najważniejszych zmian. Dodatkowo, w oddzielnym dokumencie zawierającym informacje o zmianach, ISTQB® przedstawia szczegółową zgodność między celami nauczania w wersjach 3.1.2 i 4.0 sylabusa.

W tej aktualizacji wprowadzono następujące zmiany:

Struktura sylabusa

Wszystkie cele nauczania zostały zmienione tak, by były atomowe, by można było je łatwo powiązać z treścią sylabusa i żeby każda sekcja sylabusa była powiązana z jakimś celem nauczania. Każdy cel nauczania jest opisany w sekcji o tym samym numerze (np. TA-1.1.1 jest omówiony w sekcji 1.1.1). Celem było ułatwienie czytania, zrozumienia, nauki i tłumaczenia tej wersji, koncentrując się na zwiększeniu praktycznej użyteczności i równowadze między wiedzą a umiejętnościami.

W wersji 4.0 znajduje się 36 celów nauczania w porównaniu z 31 w wersji 3.1.2. Kilka celów nauczania K4 zostało obniżonych do K2 lub K3. W przypadku celów nauczania związanych z technikami testowania motywacją było to, że należy skupić się na *stosowaniu* technik testowania, a nie na *analizowaniu* dokumentacji w celu dalszego projektowania testów. Niektóre cele nauczania zostały połączone w jeden cel nauczania. Poniższa tabela przedstawia szczegółowe statystyki dotyczące liczby celów nauczania i czasu szkolenia.

Wersja sylabusa	Liczba celów K2 (czas)	Liczba celów K3 (czas)	Liczba celów K4 (czas)	Razem celów (czas)
Bieżąca (4.0)	22 (330 min.)	11 (660 min.)	3 (225 min.)	36 (1 215 min.)
Poprzednia (3.1.2)	16 (240 min.)	5 (300 min.)	10 (750 min.)	31 (1 290 min.)

Klasyfikacja technik testowania

Struktura rozdziału 3 odzwierciedla bardziej szczegółową klasyfikację technik testowania w porównaniu z wersją 3.1.2. Czarnoskrzynkowe techniki testowania są obecnie podzielone na techniki oparte na danych, oparte na zachowaniu i oparte na regułach, zgodnie z typem modelu przedmiotu testów.

Rozszerzenie zakresu rozdziału 5

Rozdział 5 w poprzedniej wersji sylabusa poświęcony był wyłącznie przeglądom. W wersji 4.0 został on rozszerzony o omówienie innych istotnych form praktyk zapobiegania defektom stosowanych przez analityka testów, takich jak wykorzystanie modeli do wykrywania defektów w specyfikacjach, analiza wyników testów w celu poprawy wykrywania defektów oraz wykorzystanie klasyfikacji defektów do wsparcia analizy przyczyny podstawowej.

Zmiany w celach nauczania

- Rozdział 1 został przekształcony w sekcję dotyczącą czynności testowych oraz sekcję dotyczącą testaliów.

- Temat dotyczący przypadków testowych wysokiego poziomu i przypadków testowych niskiego poziomu (dawny TA-1.4.2) został obniżony z K4 do K2 (TA-1.3.1).
- Cel nauczania K3 dotyczący testowania opartego na ryzyku (dawny TA-2.1.1) został podzielony na dwa cele: K2 TA-2.1.1 związany z analizą ryzyka i K4 TA-2.2.1 związany z kontrolą ryzyka.
- Podział na klasy równoważności (dawny K4 TA-3.2.1) i analiza wartości brzegowych (dawny K4 TA-3.2.2) zostały zastąpione bardziej ogólnym K3 TA-3.1.1 dotyczącym testowania dziedziny.
- Testowanie sposobem par (dawny K4 TA-3.2.6) i drzewa klasyfikacji (dawny K2 TA-3.2.5) zostały połączone w jeden, bardziej ogólny K3 TA-3.1.2 dotyczący testowania kombinatorycznego.
- Testowanie przejść pomiędzy stanami (dawny K4 TA-3.2.4) zostało zastąpione przez K3 TA -3.2.2, który koncentruje się na pokryciu N-przełączeń i pokryciu pętli. Usunięto nadmiarowości w stosunku do sylabusa poziomu podstawowego.
- Testowanie oparte na przypadkach użycia (dawny K4 TA-3.2.7) zostało zastąpione bardziej ogólnym K3 TA-3.2.3 dotyczącym testowania opartego na scenariuszach.
- Testowanie tablic decyzyjnych (dawny K4 TA-3.2.3) zostało obniżone do K3 (TA-3.3.1).
- Testowanie eksploracyjne (dawny K3 TA-3.3.2) zostało zastąpione dwoma oddzielnymi celami nauczania: dotyczącym przygotowywania kart opisu testu (K3 TA-3.4.1) oraz przygotowywania list kontrolnych wspierających testowanie oparte na doświadczeniu (K3 TA-3.4.2). Usunięto nadmiarowości w stosunku do sylabusa poziomu podstawowego.
- Cztery cele nauczania związane z porównywaniem technik testowania i stosowaniem najbardziej odpowiedniej techniki (dawne: K4 TA-3.2.8, K2 TA-3.3.3, K2 TA-3.4.1, K2 TA-3.3.1) zostały połączone w jeden cel nauczania K4 TA-3.5.1.
- Cztery cele nauczania dotyczące testowania funkcjonalności (dawne: K2 TA-4.2.1, K2 TA-4.2.2, K2 TA-4.2.3 i K4 TA-4.2.7) zostały połączone w jeden K2 TA-4.1.1. Temat został uproszczony, ponieważ testowanie funkcjonalności zostało w dużej mierze opisane w kontekście technik testowania w rozdziale 3.
- Tematy z rozdziału 6 (Narzędzia testowe) zostały przeniesione do podrozdziału 1.3, skupiającego się na bardziej praktycznych kwestiach. Stary K3 TA-6.2.1 został nieznacznie zmieniony na K3 TA-1.3.6 „Zastosować testowanie oparte na słowach kluczowych, aby opracować skrypty testowe”. Stary K2 TA-6.3.1 został nieznacznie zmieniony na K2 TA-1.3.7 „Podsumować typy narzędzi do zarządzania testaliami”.
- Dwa podobne cele nauczania dotyczące przeglądów (dawne K3 TA-5.2.1 i K3 TA-5.2.2) zostały połączone w jeden cel nauczania K3 TA-5.2.2.

Nowe tematy

- Kryteria jakości dla przypadków testowych (K2 TA-1.3.2)
- Wymagania dotyczące środowiska testowego (K2 TA-1.3.3)
- Określanie wyroczni testowych (K2 TA-1.3.4)
- Wymagania dotyczące danych testowych (K2 TA-1.3.5)
- Testowanie losowe (K2 TA-3.1.3)
- Testowanie CRUD (K2 TA-3.2.1)
- Testowanie metamorficzne (K3 TA-3.3.2)
- Testowanie w tłumie (K2 TA-3.4.3)

- Korzyści i ryzyka związane z automatyzacją projektowania testów (K2 TA-3.5.2)
- Wkład analityka testów w zapobieganie defektom (K2 TA-5.1.1)
- Wykorzystanie modeli do wykrywania defektów w specyfikacjach (K3 TA-5.2.1)
- Analiza wyników testów w celu poprawy wykrywania defektów (K4 TA-5.3.1)
- Wspieranie analizy przyczyny podstawowej poprzez klasyfikację defektów (K2 TA-5.3.2).

Aktualizacja norm

Zmiany w najnowszych wersjach międzynarodowych norm dotyczących jakości oprogramowania (ISO/IEC 25010, 2023) i technik testowania (ISO/IEC/IEEE 29119-4, 2021) spowodowały dostosowanie do nich odpowiednich treści sylabusa.

10. Dodatek D – wykaz skrótów

Skrót	Definicja
AI	sztuczna inteligencja (ang. *artificial intelligence*)
AT	analityk testów (ang. *test analyst*)
BPMN	model i notacja procesu biznesowego (ang. *Business Process Model and Notation*)
CRUD	tworzenie, odczyt, aktualizacja, usuwanie (ang. *Create*, *Read*, *Update*, and *Delete*)
CSV	wartości oddzielone przecinkiem (ang. *comma-separated value*)
DDP	procent wykrytych defektów (ang. *defect detection percentage*)
DRE	efektywność usuwania defektów (ang. *defect removal effectiveness*)
JSON	notacja obiektowa JavaScript (ang. *JavaScript Object Notation*)
MBT	testowanie oparte na modelu (ang. *model-based testing*)
ODC	ortogonalna klasyfikacja defektów (ang. *orthogonal defect classification*)
PCE	efektywność powstrzymania fazowego (ang. *phase containment effectiveness*)
RCA	analiza przyczyny podstawowej (ang. *root cause analysis*)
RM	relacja metamorficzna (ang. *metamorphic relation*)
RODO	rozporządzenie o ochronie danych osobowych (ang. *General Data Protection Regulation*, *GDPR*)
SDLC	cykl wytwarzania oprogramowania (ang. *software development lifecycle*)
TM	testowanie metamorficzne (ang. *metamorphic testing*)
UML	uniwersalny język modelowania (ang. *Unified Modeling Language*)
UX	doświadczenie użytkownika (ang. *user experience*)
XML	rozszerzalny język znaczników (ang. *Extensible Markup Language*)

11. Dodatek E – terminy specyficzne dla dziedziny

Termin	Definicja
analiza Pareto	Podejście służące do identyfikacji obszarów problemowych lub zadań, które przyniosą największe korzyści.
badanie użytkowników	Dyscyplina polegająca na poznawaniu potrzeb i procesów myślowych użytkowników poprzez badanie sposobu, w jaki wykonują zadania, obserwowanie ich interakcji z komponentem lub systemem lub poprzez analizę i interpretację danych.
macierz CRUD	Macierz wskazująca typy działań funkcji dotyczących podmiotów w ramach systemu.
mapa podróży użytkownika	Dokumentacja wizualizacji doświadczeń użytkownika, która pokazuje kroki podejmowane przez użytkownika w procesie realizacji celu.
semantyka danych	Znaczenie i interpretacja danych.
technika pięciu pytań „dlaczego”	Technika iteracyjnego zadawania pytań stosowana w celu ustalenia pierwotnej przyczyny usterki lub problemu poprzez pięciokrotne powtórzenie pytania „dlaczego?”, za każdym razem kierując bieżące „dlaczego” do odpowiedzi na poprzednie „dlaczego”.

12. Dodatek F – model jakości oprogramowania

ISO/IEC 25010:2023 (wersja aktualna)	ISO/IEC 25010:2011 (wersja poprzednia)	Uwagi	Sylabusy ISTQB®
Przydatność funkcjonalna	**Przydatność funkcjonalna**		**TA**
Kompletność funkcjonalna	Kompletność funkcjonalna		
Poprawność funkcjonalna	Poprawność funkcjonalna		
Adekwatność funkcjonalna	Adekwatność funkcjonalna		
Wydajność	**Wydajność**		PT, TTA
Zachowanie w czasie	Zachowanie w czasie		
Zużycie zasobów	Zużycie zasobów		
Pojemność	Pojemność		
Zgodność (kompatybilność)	**Zgodność (kompatybilność)**		**TA**, TTA
Współistnienie	Współistnienie		TTA
Współdziałanie	Współdziałanie		**TA**
Zdolność interakcji	**Użyteczność**	**Zmiana nazwy**	UT, **TA**
Rozpoznawalność	Stosowność		
Łatwość nauki	Łatwość uczenia się		
Łatwość obsługi	Łatwość obsługi		
Błędoodporność	Ochrona przed błędami użytkownika		
Angażowalność	Estetyka interfejsu użytkownika	Zmiana nazwy	
Inkluzywność	Dostępność	Rozdzielono i zmieniono nazwę	
Pomocność			
Samoopisywalność		Nowa podcharakterystyka	
Niezawodność			TTA
Bezawaryjność	Dojrzałość	Zmiana nazwy	
Osiągalność	Osiągalność		
Tolerowanie usterek	Tolerowanie usterek		
Odtwarzalność	Odtwarzalność		
Zabezpieczenia	**Zabezpieczenia**		SEC, TTA
Poufność	Poufność		
Integralność	Integralność		
Niezaprzeczalność	Niezaprzeczalność		
Rozliczalność	Rozliczalność		
Wiarygodność	Autentykacja		
Odporność na ataki		Nowa podcharakterystyka	
Utrzymywalność	**Utrzymywalność**		TTA
Modułowość	Modułowość		
Reużywalność	Łatwość ponownego użycia		
Analizowalność	Analizowalność		
Modyfikowalność	Modyfikowalność		
Testowalność	Testowalność		
Elastyczność	**Przenaszalność**	**Zmiana nazwy**	TTA, **TA**, PT
Adaptowalność			**TA**, TTA
Skalowalność		Nowa podcharakterystyka	PT
Instalowalność	Instalowalność		**TA**, TTA
Zastępowalność	Zastępowalność		TTA
Bezpieczeństwo		**Nowa charakterystyka**	AuT
Ograniczalność operacyjna		Nowa podcharakterystyka	
Rozpoznawalność ryzyka		Nowa podcharakterystyka	
Bezpieczność awarii (ang. *fail safe*)		Nowa podcharakterystyka	
Ostrzegalność o zagrożeniu		Nowa podcharakterystyka	
Bezpieczność integracji		Nowa podcharakterystyka	

Powyższa tabela przedstawia charakterystyki jakościowe modelu jakości produktu ISO/IEC 25010 (ISO/IEC 25010, 2023). Wskazuje ona, które charakterystyki/podcharakterystyki jakościowe są uwzględnione w niniejszym sylabusie, a które są dyskutowane w innych sylabusach ISTQB® (techniczny analityk testów (TTA), testowanie wydajności (PT), testowanie użyteczności (UT), tester oprogramowania automotive (AuT) i testowanie zabezpieczeń (SEC). Jeśli kilka sylabusów obejmuje daną charakterystykę jakościową jako pierwszy wymieniony jest ten, który opisuje ją najdokładniej. Tabela zawiera również porównanie aktualnego modelu ISO/IEC 25010 z wersją z 2011 r. używaną w poprzedniej wersji niniejszego sylabusa.

13. Dodatek G – znaki towarowe

ISTQB® jest zarejestrowanym znakiem towarowym International Software Testing Qualifications Board.

UML® jest zarejestrowanym znakiem towarowym Object Management Group (OMG).

BPMN™ jest znakiem towarowym Object Management Group (OMG).

14. Indeks