Testowanie poziom zwinny ISTQB by Wojciech Baczyński

1.1 Podstawy zwinnego wytwarzania oprogramowania

1.1.1 Kandydat pamięta podstawowe zasady zwinnego wytwarzania oprogramowania w oparciu o Manifest Agile

Manifest Agile zawiera cztery główne zasady:

- ludzie i współpraca ponad procesy i narzędzia,
- działające oprogramowanie ponad obszerną dokumentację,
- współpraca z klientem ponad formalne ustalenia,
- reagowanie na zmiany ponad podążanie za planem.

1.1.2 Kandydat rozumie korzyści wynikające z podejścia “cały zespół”

Wykorzystanie podejścia „cały zespół” do wytwarzania produktów stanowi jedną z głównych korzyści zwinnego wytwarzania oprogramowania. Podejście „cały zespół” ma wiele zalet, między innymi:

- Poprawia współpracę i komunikację w zespole,
- Umożliwia wykorzystanie synergii umiejętności członków zespołu z pożytkiem dla całego projektu,
- Czyni wszystkich odpowiedzialnymi za jakość.

1.1.3 Kandydat rozumie korzyści wynikające z wczesnego i częstego otrzymywania informacji zwrotnej

Zyskami z szybkiego i częstego otrzymywania informacji zwrotnej są, m.in.:

- Uniknięcie niezrozumienia wymagań, wykrycie czego w późnych etapach cyklu wytwórczego oraz poprawienie będzie dużo bardziej kosztowne,
- Wyjaśnianie zgłoszeń klienta odnośnie właściwości. Wczesne i regularne wyjaśnianie żądań podczas wytwarzania, zwiększa prawdopodobieństwo, że kluczowe właściwości będą dostępne dla użytkownika wcześniej i że produkt będzie bardziej odzwierciedlał to, co klient potrzebuje,
- Natychmiastowe ostrzeganie zespołu wytwórczego o problemach jakościowych przy wykorzystaniu procesu ciągłej integracji. Problemy mogą być łatwiej izolowane i rozwiązywane niż gdyby zostały wykryte później w cyklu wytwórczym,
- Informowanie zespołu zwinnego o jego wydajności i zdolności do dostarczenie produktu na czas,
- Promowanie stałego tempa prac projektowych.

1.2 Aspekty podejść zwinnych

1.2.1 Kandydat pamięta podejścia zwinne do wytwarzania oprogramowania

Nie istnieje jedno podejście do zwinnego wytwarzania oprogramowania, ale wiele różnych podejść. Każde z nich inaczej implementuje wartości i zasady z Manifestu Agile. Do popularnych podejść należą:

- eXtreme Programming,

- scrum,
- kanban.

1.2.2 Kandydat potrafi napisać historyjki użytkownika we współpracy z wytwórcami, przedstawicielami biznesu i właścicielem produktu

Historyjki użytkownika muszą obejmować zarówno właściwości funkcjonalne jak i niefunkcjonalne. Każda historyjka powinna zawierać kryteria akceptacji dla tych właściwości. Te kryteria powinny być definiowane we współpracy pomiędzy przedstawicielami biznesu, deweloperami i testerami. Kryteria te dostarczają deweloperem i testerom dokładniejszej wizji właściwości, którą reprezentanci biznesu będą sprawdzać. Zespół zwinny uważa zadanie za ukończone, gdy zbiór kryteriów akceptacyjnych jest spełniony.

1.2.3 Kandydat rozumie, jak spotkania retrospektywne mogą być wykorzystywane jako mechanizm do doskonalenia procesu w projektach zwinnych

W wytwarzaniu zwinnym, retrospektywa to spotkanie na końcu każdej iteracji, na którym dyskutuje się o tym, co się udało, co można poprawić, jak wdrożyć poprawki i powtórnie osiągnąć sukces w następnej iteracji. Retrospektywa obejmuje takie zagadnienia jak proces, ludzie, organizacja, relacje i narzędzia. Regularnie przeprowadzane spotkanie retrospektywne, po których następują odpowiednie działania, są krytyczne dla samoorganizacji i ciągłego doskonalenia wytwarzania oraz testowania.

1.2.4 Kandydat rozumie wykorzystanie i cele ciągłej integracji

Dostarczanie przyrostów produktu wymaga niezawodnego, pracującego, zintegrowanego oprogramowania na koniec każdego sprintu. Ciągła integracja podejmuje to wyzwanie poprzez regularne łączenie wszystkich zmian wykonanych w oprogramowaniu i integrację wszystkich zmienionych modułów przynajmniej raz dziennie. Zarządzanie konfiguracją, kompilacja, budowanie wersji, wprowadzanie i testowanie tworzą pojedynczy, zautomatyzowany, regularnie powtarzany proces. Ponieważ deweloperzy integrują swoją pracę stale, budują stale i testują stale, błędy w kodzie są wykrywane szybciej.

1.2.5 Kandydat zna różnice pomiędzy planowaniem iteracji i wydania, a także wie, w jaki sposób tester dodaje wartości każdej z tych aktywności

Planowanie wydania przewiduje wydanie produktu, często kilka miesięcy od rozpoczęcia projektu. Planowanie wydań definiuje i przedefiniowuje backlog produktu i może zawierać przerabianie dużych historyjek użytkownika w zbiór mniejszych historyjek. Planowanie wydania dostarcza podstawy do podejścia do testów oraz do planu testów i rozciąga się na wszystkie iteracje. Plany wydania są wysokopoziomowe.

Testerzy są zaangażowani w planowanie wydania będąc szczególnie użytecznymi przy następujących czynnościach:

- Definiowanie testowalnych historyjek użytkownika, włączając w to kryteria akceptacji,
- Branie udziału w analizie ryzyk projektowych i jakościowych,
- Szacowanie nakładu pracy związanego z historyjkami użytkownika,
- Definiowanie potrzebnych poziomów testów,
- Planowanie testów dla wydania.

Podczas planowania iteracji, zespół wybiera historyjki użytkownika z ustawionych według priorytetu w backlogu produktu, uszczegóławia historyjki użytkownika, wykonuje analizę ryzyka dla historyjek, oraz szacuje pracę

potrzebną do wykonania każdej historyjki. Jeżeli jakaś historyjka użytkownika jest zbyt mglista i nie powiodą się próby jej rozjaśnienia, zespół może odrzucić tę historyjkę i wziąć następną historyjkę zgodnie z priorytetami. Reprezentanci biznesu mają obowiązek odpowiadać na pytania zespołu na temat każdej historyjki, tak by zespół wiedział jak implementować każdą historyjkę i jak ją testować.

Testerzy są zaangażowani w planowanie iteracji będąc szczególnie użytecznymi w następujących czynnościach:

- Udział w szczegółowej analizie ryzyka dla historyjek użytkownika,
- Określanie testowalności historyjek użytkownika,
- Tworzenie testów akceptacyjnych dla historyjek użytkownika,
- Dzielenie historyjek użytkownika na zadania (zwłaszcza zadania testowe),
- Szacowanie pracochłonności zadań testowych.
- Identyfikowanie funkcjonalnych i niefunkcjonalnych własności systemu, które trzeba przetestować,
- Wspieranie i udział w automatyzacji testów na różnych poziomach testowania.

2.1 Różnice pomiędzy tradycyjnym i zwinnym podejściem do testowania

2.1.1 Kandydat potrafi opisać różnice pomiędzy testowaniem w projektach zwinnych i tradycyjnych (nie zwinnych)

Jedną z podstawowych różnic pomiędzy tradycyjnymi a zwinnymi cyklami życia jest idea bardzo krótkich iteracji, z których każda kończy się działającym oprogramowaniem dostarczającym interesariuszom biznesowym wartościowej funkcjonalności. Na początku projektu następuje okres planowania wydania. Następnie mamy ciąg iteracji. Na początku każdej iteracji występuje okres jej planowania. Po wypracowaniu zakresu iteracji, wybrane historyjki użytkownika są wytwarzane, integrowane z systemem oraz testowane. Tego typu iteracje są wysoce dynamiczne, z czynnościami wytwarzania, integracji i testowania trwającymi przez cały czas iteracji, z istotnym zrównolegleniem i zachodzeniem na siebie tych czynności. Czynności testowe wykonywane są przez cały czas (np. codziennie), a nie tylko jako końcowe czynności iteracji.

2.1.2 Kandydat potrafi opisać, w jaki sposób czynności kodowania i testowania są zintegrowane w podejściu zwinnym

Projekty zwinne często bardzo mocno wykorzystują narzędzia automatyzujące tworzenie, testowanie oraz zarządzanie tworzeniem oprogramowania. Deweloperzy wykorzystują narzędzia do analizy statycznej, testów jednostkowych i do mierzenia ich pokrycia. Programiści ciągle komitują kod i testy jednostkowe do systemów zarządzania konfiguracją przy użyciu struktur budowania i testowania. Takie struktury umożliwiają ciągłą integrację oprogramowania z systemem; analiza statyczna i testy jednostkowe są powtarzane, gdy tylko nowe oprogramowanie zostanie wprowadzane.

2.1.3 Kandydat potrafi opisać rolę niezależnego testowania w projektach zwinnych

Niezależni testerzy są często bardziej efektywni w znajdowaniu defektów. W niektórych zespołach zwinnych, deweloperzy wytwarzają większość testów w formie testów automatycznych. Jeden lub więcej testerów może być członkiem zespołu, wykonując wiele zadań testowych. Jednakże umieszczenie testera w zespole rodzi pewne ryzyko utraty jego niezależności i braku obiektywnej oceny.

Inne zespoły zwinne utrzymują w pełni niezależny, wydzielony zespól testerski i angażują testerów „na żądanie” na końcu każdego sprintu. Taki proces pozwala na zachowanie niezależności, co sprawia, że testerzy mogą dokonywać obiektywnej, bezstronnej oceny oprogramowania. Jednakże brak czasu, brak zrozumienia nowych

właściwości produktu oraz problemy w relacjach z interesariuszami biznesowymi i deweloperami mogą często powodować kłopoty przy takim podejściu.

Trzecia opcja to posiadanie niezależnego, oddzielonego zespołu testowego, gdzie testerzy są przydzieleni do zespołu zwinnego na zasadzie długoterminowej na początku projektu, co umożliwia im utrzymanie niezależności z jednoczesnym poznaniem produktu i nawiązaniem mocnych relacji z innymi członkami zespołu. Dodatkowo, niezależny zespół testowy może utrzymywać kilku wyspecjalizowanych testerów (poza zespołem zwinnym) pracujących nad zadaniami długoterminowymi lub nienależącymi do sprintu, jak np. tworzenie narzędzi do automatycznego testowania, wykonywanie testów niefunkcjonalnych, tworzenie i utrzymywanie środowisk i danych testowych oraz zajmowanie się poziomami testów, które trudno wpasować w sprinty (np. testy integracji systemów).

2.2 Status testowania w projektach zwinnych

2.2.1 Kandydat potrafi opisać podstawowy zbiór produktów używanych do informowania o statusie testów w projekcie zwinnym, włączając w to postęp testów i jakość produktu

By zapewnić stałe, szczegółowe wizualne przedstawienie aktualnego statusu całego zespołu, włączając w to status testów, zespoły mogą wykorzystywać zwinną tablicę zadań (ang. Agile task board). Karty historyjek, zadania deweloperskie, zadania testowe i inne zadania powstałe podczas planowania iteracji (patrz sekcja 1.2.5) są prezentowane na tablicy zadań, często przy pomocy kolorowych kart, gdzie kolor określa typ zadania. Podczas iteracji, zarządzanie postępem prac odbywa się poprzez przesuwanie odpowiednich kart zadań na tablicy miedzy kolumnami takimi jak: „do zrobienia", „w toku", „weryfikacja" oraz „zrobione". Zespoły zwinne mogą używać narzędzi do obsługi kart historyjek oraz tablicy zadań, które to narzędzia mogą automatyzować uaktualnianie tablicy rozdzielczej i statusu.

2.2.2 Kandydat potrafi opisać proces ewoluowania testów w czasie wielu iteracji i potrafi wyjaśnić, dlaczego automatyzacja testów jest istotna przy zarządzaniu ryzykiem regresji w projektach zwinnych

W projektach zwinnych, produkt rośnie po zakończeniu każdej iteracji. Tym samym zwiększa się zakres testów. Wraz z testami zmian kodu dokonanych w czasie bieżącej iteracji, testerzy sprawdzają także, czy nie obniżono jakości właściwości, które były wytworzone iprzetestowane w poprzednich iteracjach. Ryzyko regresu właściwości jest wysokie w wytwarzaniu zwinnym, ze względu na rozległe zmiany w kodzie (dodawanie, modyfikowanie i usuwanie linii kodu z wersji na wersję). Ponieważ reagowanie na zmiany jest podstawową zasadą zwinności, dlatego też, w celu zaspokojenia potrzeby biznesowej, zmianom mogą podlegać uprzednio dostarczone właściwości. By utrzymać prędkość wytwarzania bez zaciągania dużego długu technicznego, krytyczna staje się jak najwcześniejsza inwestycja zespołu w automatyzację testów na wszystkich poziomach testów. Krytyczne jest także, by wszystkie produkty testowe takie jak: testy automatyczne, manualne przypadki testowe, dane testowe i inne artefakty testowe były aktualizowane w każdej iteracji. Dlatego też zaleca się, by wszystkie produkty testowe były zarządzane przy użyciu narzędzia do zarządzania konfiguracją. Zapewnia to kontrolę wersji, ułatwia dostęp wszystkim członkom zespołu, ułatwia zmiany wymagane przez zmieniającą się funkcjonalność, a równocześnie ciągle zachowuje historyczne informacje o produktach testowych.

2.3 Rola i umiejętności testera w zespole zwinnym

2.3.1 Kandydat rozumie umiejętności (ludzkie, dziedzinowe i testowe) testera w projekcie zwinnym

Testerzy w zespołach zwinnych powinni:

- Być pozytywni i zorientowani na rozwiązywanie problemów we współpracy z członkami zespołu i interesariuszami,
- Wykazywać krytyczne, zorientowane na jakość, sceptyczne myślenie o produkcie,
- Aktywnie zdobywać informacje od interesariuszy (nie polegać wyłącznie na spisanej specyfikacji),
- Dokładnie sprawdzać i raportować wyniki testów, postęp testów i jakość produktu,
- Skutecznie pracować z przedstawicielami klienta i interesariuszami definiując testowalne historyjki użytkownika (przede wszystkim kryteria akceptacyjne),
- Współpracować w zespole, pracując w parach z programistami lub innymi członkami zespołu,
- Szybko reagować na zmiany, przez np. zmienianie, dodawanie i poprawianie przypadków testowych,
- Planować i organizować swoją własną pracę.

2.3.2 Kandydat rozumie rolę testera w projekcie zwinnym

Rola testera w zespole zwinnym obejmuje czynności, które generują i dostarczają informacje zwrotne dotyczące nie tylko statusu testów i jakości produktu, ale także jakości procesu. Poza czynnościami opisanymi w innych miejscach tego sylabusa, czynności te obejmują:

- Zrozumienie, implementowanie i poprawianie strategii testów,
- Mierzenie i raportowanie pokrycia testowego względem wszystkich dostępnych wymiarów pokrycia,
- Zapewnienie poprawnego użycia narzędzi testowych.
- Konfigurowanie, używanie i zarządzanie środowiskiem testowym i danymi testowymi,
- Raportowanie defektów i współpraca z zespołem przy ich rozwiazywaniu,
- Trenowanie (coaching) innych członków zespołu w różnych aspektach związanych z testowaniem.
- Zapewnienie, że odpowiednie zadania testowe są harmonogramowane podczas planowania wydania i iteracji,
- Aktywna współpraca z deweloperami, interesariuszami biznesowymi i właścicielami produktu, by poprawnie interpretować wymagania, zwłaszcza w zakresie ich testowalności, spójności i kompletności,
- Proaktywne uczestnictwo w retrospektywach zespołu, sugerowanie i implementowanie udoskonaleń.

3.1 Metody testowania zwinnego

3.1.1 Kandydat pamięta pojęcia: wytwarzanie sterowane testami (TDD), wytwarzanie sterowane testami akceptacyjnymi (ATDD), wytwarzanie sterowane zachowaniem (BDD)

Wytwarzanie sterowane testami to technika wytwarzania kodu kierowana przez zautomatyzowane przypadki testowe. Proces wytwarzania sterowanego testami zawiera:

- Dodanie testu, który stanowi wyobrażenie programisty o pożądanej funkcjonalności małego fragmentu kodu,
- Wykonanie testu, który nie powinien przejść, bo kod jeszcze nie istnieje,
- Naprzemienne pisanie kodu i uruchamianie testu, aż test przejdzie,
- Refaktoryzacja kodu po tym jak test przeszedł, ponowne uruchomienie testu w celu upewnienia się, że nadal przechodzi po tym jak kod został zrefaktoryzowany,
- Powtarzanie procesu dla następnego małego fragment kodu z uruchamianiem zarówno poprzednich testów, jak i nowo dodanych.

Wytwarzanie sterowane testami akceptacyjnymi definiuje kryteria akceptacji oraz testy podczas tworzenia historyjek użytkownika (patrz podrozdział 1.2.2). ATDD jest podejściem opartym na współpracy, które pozwala każdemu z interesariuszy na zrozumienie, jak każdy z modułów ma się zachowywać oraz co mają zrobić programiści, testerzy oraz reprezentanci biznesu, żeby zapewnić dany sposób zachowania. Proces wytwarzania sterowanego testami akceptacyjnymi został wyjaśniony w podrozdziale 3.2.2. ATDD tworzy reużywalne testy do wykorzystania w testach regresyjnych. Tworzeniei wykonywanie takich testów jest wspierane przez specyficzne narzędzia, często w procesie ciągłej integracji. Narzędzia te potrafią połączyć się z warstwami danych i usług, co pozwala na wykonywanie testów systemowych i akceptacyjnych. Wytwarzanie sterowane testami akceptacyjnymi pozwala na szybkie naprawienie defektów oraz walidację zachowania właściwości. Pomaga ono stwierdzić, czy kryteria akceptacyjne zostały spełnione.

Wytwarzanie sterowane zachowaniem pozwala programiście na skupienie się testowaniu kodu w oparciu o wymagane zachowanie oprogramowania. Ponieważ testy oparte są o faktyczne zachowanie oprogramowania, są one łatwiejsze do zrozumienia przez członków zespołu oraz interesariuszy. Przy definiowaniu kryteriów akceptacji w oparciu o format “mając/kiedy/wtedy” mogą być używane konkretne struktury (ang. framework) do wytwarzania sterowanego zachowaniem:

- Mając pewien kontekst początkowy,
- Kiedy nastąpi zdarzenie,
- Wtedy zapewnione jest pewne wyjście.

3.1.2 Kandydat pamięta pojęcie piramidy testowa

System software’owy może być testowany na różnych poziomach. Typowe poziomy testów to, od najniższego do najwyższego, testy jednostkowe, integracyjne, systemowe i akceptacyjne. Piramida testowa akcentuje potrzebę posiadania dużej liczby testów na niskim poziomie (dół piramidy). Gdy wytwarzanie przechodzi do wyższych poziomów, liczba testów maleje (szczyt piramidy). Na ogół testy jednostkowe i integracyjne są automatyzowane i tworzone przy pomocy narzędzi wykorzystujących interfejs programowania aplikacji (API). Na poziomie testów systemowych i akceptacyjnych, testy automatyczne są tworzone z wykorzystaniem narzędzi opartych o interfejs użytkownika. Pojęcie piramidy testowej jest oparte na testowej zasadzie wczesnego zapewnienia jakości i testowania, tzn. eliminacji usterek na jak najwcześniejszym etapie cyklu życia oprogramowania.

3.1.3 Kandydat potrafi opisać kwadranty testowe i ich związki z poziomami testów i typami testów

W modelu kwadrantów testowych, testy mogą być zorientowane na biznes (użytkownik) lub technologię (programista). Pewne testy wspomagają prace wykonywane przez zespół zwinny i potwierdzają zachowanie oprogramowania, inne z kolei weryfikują sam produkt. Testy mogą być w pełni manualne, w pełni zautomatyzowane, kombinacją testów manualnych i automatycznych, lub testami manualnymi wspomaganymi przez narzędzia. Istnieją następujące cztery kwadranty:

- Kwadrant Q1 jest na poziomie jednostkowym, zorientowany na technologię i wspomaga deweloperów. W tym kwadrancie znajdują się testy jednostkowe. Testy te powinny być zautomatyzowane i umieszczone w procesie ciągłej integracji,
- Kwadrant Q2 jest na poziomie systemowym, zorientowany na biznes i potwierdza zachowania się produktu. W tym kwadrancie zawarte są testy funkcjonalne, przykłady, testowanie historyjek, prototypy oparte na doświadczeniu użytkowników oraz symulacje. Te testy sprawdzają kryteria akceptacyjne. Mogą

one być manualne lub zautomatyzowane. Często są tworzone podczas wytwarzania historyjek użytkownika idlatego też poprawiają jakość historyjek. Są użyteczne do tworzenia zestawów automatycznych testów regresji,
- Kwadrant Q3 jest na poziomie systemowym lub akceptacji użytkownika, zorientowany na biznes i zawiera testy, które krytykują produkt używając realistycznych scenariuszy i danych. Ten kwadrant zawiera testy eksploracyjne, scenariusze, sprinty procesów, testowanie użyteczności, testy akceptacyjne użytkownika, testy alfa i beta. Te testy są często manualne i zorientowane na użytkownika,
- Kwadrant Q4 jest na poziomie systemowym lub akceptacji operacyjnej, zorientowany na technologię i zawiera testy, które krytykują produkt. Ten kwadrant zawiera testy wydajnościowe, obciążeniowe, przeciążające, testowanie skalowalności, zabezpieczeń, pielęgnowalności, zarządzani pamięcią, testowanie kompatybilności i współdziałania, migracji danych, infrastruktury oraz testowanie odzyskiwania. Testy te są często automatyzowane.

3.1.4 Kandydat potrafi dla danego projektu zwinnego, pełnić rolę testera w zespole scrumowym

Praca zespołowa to podstawowa zasada w wytwarzaniu zwinnym. Zwinność kładzie nacisk na zaangażowanie całego zespołu, składającego się z deweloperów, testerów i przedstawicieli biznesu, pracujących razem. Poniżej podano najlepsze praktyki organizacyjne i behawioralne w zespołach scrumowych:

- Interdyscyplinarny: Zespół pracuje sprawnie razem nad strategią testów, planowaniem testów, specyfikacją testów, wykonywaniem testów, oceną wyników testów i raportowaniem wyników z testów,
- Samoorganizujący się: Zespół może składać się z samych deweloperów, ale – jak podano w sekcji 2.1.5 – w sytuacjach idealnych do zespołu powinien dołączyć jeden lub więcej testerów,
- W jednym miejscu: Testerzy siedzą razem z deweloperami i właścicielem produktu,
- Współpracujący: Testerzy współpracują z innymi członkami zespołu, innymi zespołami, interesariuszami, właścicielem produktu i Scrum Masterem,
- Pełnomocny: Decyzje techniczne dotyczące projektowania i testowania są podejmowane przez członków zespołu jako całość (programiści, testerzy i Scrum Master), we współpracy z właścicielem produktu i z innymi zespołami w razie potrzeby,
- Zaangażowany: Tester jest zobowiązany do kwestionowania i oceniania zachowania oraz właściwości produktu, z uwzględnieniem oczekiwań i potrzeb klientów oraz użytkowników,
- Przejrzysty: Postęp prac programistycznych i testerskich jest widoczny na zwinnej tablicy zadań (patrz sekcja 2.2.1)
- Wiarygodny: Tester musi zapewnić wiarygodność strategii testowej, jej wdrożenia iwykonania. W przeciwnym wypadku interesariusze nie będą mieli zaufania do wyników testów. Często wiarygodność testów osiąga się przez dostarczanie interesariuszom informacji na temat procesu testowego,
- Otwarty na informację zwrotną: Informacja zwrotna jest istotnym aspektem osiągnięcia sukcesu w projekcie, zwłaszcza w projekcie zwinnym. Retrospektywy pozwalają zespołowi na uczenie się na sukcesach i porażkach,
- Elastyczny: Testowanie musi umożliwiać reakcję na zmiany, podobnie jak inne czynności w projektach zwinnych.

Te najlepsze praktyki maksymalizują prawdopodobieństwo sukcesu testowania w projektach scrumowych.

3.2 Ocena ryzyk jakościowych i szacowanie wysiłku testowego

3.2.1 Kandydat potrafi ocenić ryzyka jakościowe produktu w projekcie zwinnym

W projektach zwinnych, analiza ryzyka jakościowego występuje na dwóch poziomach:

- Podczas planowania wydania: przedstawiciele biznesu znający właściwości do wydania dostarczają wysokopoziomowego opisu ryzyk i cały zespół, razem z testerami, może pomagać w identyfikacji i ocenie ryzyka,
- Podczas planowania iteracji, cały zespół włączając w to testerów identyfikuje i ocenia ryzyka jakościowe produktu.

Proces analizy ryzyka jakościowego w projekcie zwinnym może zostać przeprowadzony następującymi krokami:

1. Zbierz razem członków zespołu zwinnego, razem z testerem (testerami).
2. Wypisz wszystkie pozycje z backlogu produktu dla bieżącej iteracji (np. na tablicy zadań).
3. Wypisz wszystkie ryzyka jakościowe związane z każdą pozycją, biorąc pod uwagę wszystkie atrybuty jakościowe.
4. Oceń wszystkie zidentyfikowane ryzyka, co polega na wykonaniu dwóch czynności: kategoryzacji ryzyka oraz ustalenia poziomu ryzyka na podstawie prawdopodobieństwa występowania i wpływu usterek.
5. Ustal dokładność testowania proporcjonalnie do poziomu ryzyka.
6. Wybierz odpowiednie techniki testowania do łagodzenia każdego z ryzyk bazując na ryzyku, jego poziomie oraz odpowiadającemu mu atrybutowi jakości.

3.2.2 Kandydat potrafi oszacować nakład pracy testowej na podstawie zawartości iteracji i ryzyk jakościowych produktu

Podczas planowania wydania zespół zwinny szacuje pracochłonność wymaganą do skompletowania wydania. Szacunki te zawierają również pracochłonność testowania. Często wykorzystywaną techniką przez zespoły zwinne jest poker planistyczny, technika bazująca na konsensusie. Właściciel produktu lub klient czyta historyjkę użytkownika osobom estymującym. Każdy z estymujących ma talię kart z wartościami bliskimi ciągowi Fibonacciego (tj. 0, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...) lub innemu ciągowi rosnącemu (np. rozmiary koszul od XS do XXL). Wartości na kartach reprezentują ilość punktów historyjek, osobodni lub innych jednostek, w których zespół estymuje. Rekomendowany jest ciąg Fibonacciego, ponieważ liczby w tym ciągu odzwierciedlają wzrost niepewności, która rośnie wraz ze zwiększaniem się rozmiaru historyjki. Wysokie wyniki szacowania zwykle oznaczają, że historyjka nie została zrozumiana lub, że powinna zostać podzielona na kilka mniejszych.

Estymujący przeprowadzają dyskusję na temat szacowanej właściwości i w ramach potrzeb zadają pytania właścicielowi produktu.

3.3 Techniki w projektach zwinnym

3.3.1 Kandydat potrafi zinterpretować odpowiednie informacje wspomagające czynności testowe

Historyjki użytkownika to główna podstawa testów. Inne możliwe podstawy testów to:

- doświadczenie z poprzednich projektów,
- istniejące funkcje, właściwości i atrybuty jakościowe systemu,
- kod, architektura oraz projekt,
- profile użytkowników (kontekst, konfiguracja systemu i zachowanie użytkownika),
- informacja o defektach z obecnych i poprzednich projektów,
- kategoryzacja usterek w taksonomii defektów;

- dostępne standardy,
- ryzyka jakościowe (patrz sekcja 3.2.1).

3.3.2 Kandydat potrafi wyjaśnić interesariuszom biznesowym jak definiować testowalne kryteria akceptacyjne

By kryteria akceptacji były testowalne, muszą poruszać następujące kwestie (o ile są one adekwatne):

- Zachowanie funkcjonalne: Zewnętrznie obserwowalne zachowanie z czynnościami użytkownika, jako wejściami operującymi w pewnych konfiguracjach,
- Cechy jakościowe: Jak system wykonuje określone zachowanie. Cechy jakościowe można również nazywać atrybutami jakościowymi lub wymaganiami niefunkcjonalnymi. Typowe cechy jakościowe to wydajność, niezawodność, użyteczność itp.,
- Scenariusze (przypadki użycia): Ciąg akcji pomiędzy zewnętrznym aktorem (często użytkownikiem) i systemem, by osiągnąć określony cel lub zadanie biznesowe,
- Reguły biznesowe: Czynności, które mogą być wykonywane w systemie tylko pod pewnymi warunkami zdefiniowanymi przez zewnętrzne procedury lub ograniczenia, np. procedury stosowane przez firmy ubezpieczeniowe by radzić sobie z roszczeniami,
- Interfejsy zewnętrzne: Opisy połączeń pomiędzy rozwijanym systemem i światem zewnętrznym. Interfejsy zewnętrzne można podzielić na różne typy (interfejs użytkownika, interfejs do innych systemów itp.),
- Ograniczenia: Każde projektowe lub implementacyjne ograniczenie, które zawęża możliwości dewelopera. Urządzenia z oprogramowaniem osadzonym często muszą spełniać wymagania fizyczne takie jak wielkość, waga czy połączenia interfejsów,
- Definicje danych: Klient może opisać format i typ danych, dopuszczalne i domyślne wartości dla elementów danych wchodzących w skład złożonej struktury danych biznesowych (np. kod pocztowy).

3.3.3 Mając daną historyjkę użytkownika, kandydat potrafi napisać przypadki testowe dla wytwarzania sterowanego testami akceptacyjnymi

Każdy poziom testów ma swoja własna definicję ukończenia. Poniższa lista przedstawia przykłady tego kryterium dla różnych poziomów testów:

- Testy modułowe
    - 100% pokrycia decyzji gdzie tylko jest to możliwe, z dokładnym przejrzeniem każdej niewykonalnej ścieżki.
    - Analiza statyczna wykonana dla całego kodu.
    - Brak nierozwiązanych głównych defektów (uporządkowanych zgodnie z priorytetami i wagami).
    - Brak nieakceptowalnego długu technicznego pozostałego w projekcie i kodzie.
    - Przejrzany cały kod oraz testy modułowe i ich wyniki.
    - Wszystkie testy modułowe zautomatyzowane.
    - Ważne atrybuty jakościowe znajdują się w wyznaczonych granicach (np. wydajność).
- Testy integracyjne
    - Wszystkie wymagania funkcjonalne przetestowane, włączając w to zarówno testy pozytywne jak i negatywne, z pewną liczbą testów opartych na wielkości, złożoności i ryzykach.
    - Wszystkie interfejsy pomiędzy modułami przetestowane.

    - Pokryte wszystkie ryzyka jakościowe zgodnie z uzgodnionym rozmiarem testów. o Rozwiązane wszystkie ważne usterki (uporządkowane zgodnie z priorytetami wg ryzyka i ważności).
    - Zaraportowane wszystkie znalezione defekty.
    - Wszystkie możliwe testy regresyjne są zautomatyzowane i przechowywane we wspólnym repozytorium.

- Testy systemowe

    - Całościowe (end-to-end) testy historyjek użytkownika, właściwości i funkcji.
    - 100% pokrycia ról użytkowników.
    - Pokryte najważniejsze atrybuty jakościowe systemu (np. wydajność, odporność, niezawodność).
    - Testowanie przeprowadzone w środowisku (środowiskach) podobnym do produkcyjnego, w miarę możliwości włączając w to cały sprzęt i oprogramowanie dla wszystkich wspieranych konfiguracji.
    - Wszystkie ryzyka jakościowe pokryte zgodnie z uzgodnionym rozmiarem testów.
    - Zautomatyzowane wszystkie testy regresyjne (tam gdzie tylko jest to możliwe). o Wszystkie testy automatyczne przechowywane we wspólnym repozytorium.
    - Wszystkie znalezione defekty zostały zaraportowane i jeżeli to możliwe naprawione.
    - Wszystkie ważne defekty naprawione (uporządkowane zgodnie z priorytetami wg ryzyka i ważności).

Ponieważ metoda wytwarzania sterowanego testami akceptacyjnymi jest podejściem “najpierw test”, przypadki testowe są tworzone przed implementacją historyjki użytkownika. Przypadki testowe są tworzone przez zespół zwinny, w skład, którego wchodzi deweloper, tester i przedstawiciel biznesu i mogą być wykonywane ręcznie lub zautomatyzowane. Pierwszym krokiem jest warsztat, podczas którego historyjka użytkownika jest analizowana, omawiana i spisywana przez deweloperów, testerów i przedstawicieli biznesu. W tym procesie naprawiane są wszelkie braki, niejednoznaczności oraz błędy.

Następnym krokiem jest utworzenie testów. Może to robić cały zespół, lub sam tester. Tak, czy inaczej, niezależna osoba jak przedstawiciel biznesu zatwierdza przypadki testowe. Testy są przykładami opisującymi specyficzne cechy historyjki użytkownika. Przykłady te pomogą zespołowi poprawnie zaimplementować historyjkę użytkownika. Ponieważ przykłady i testy są jednym i tym samym, używa się tych pojęć naprzemiennie. Praca rozpoczyna się od podstawowych przykładów i otwartych pytań.

Zazwyczaj, pierwsze przypadki testowe są pozytywne. Testują domyślne zachowanie (bez wyjątków lub obsługi błędów) poprzez ciąg akcji wykonywanych, sprawdzających czy wszystko idzie jak oczekiwano. Po tym, jak wykonano testy pozytywne, zespół powinien stworzyć testy negatywne oraz pokryć wymagania niefunkcjonalne (np. wydajność, użyteczność). Testy pisane są tak, by każdy interesariusz był w stanie je zrozumieć. Zawierają zdania budowane w języku naturalnym, jednocześnie podając konieczne warunki wstępne (jeśli takowe występują), wejścia oraz odpowiadające im wyjścia.

Przykłady muszą pokrywać wszystkie cechy historyjki użytkownika, ale nie powinny nic dodawać do historyjki. Oznacza to, że nie powinien istnieć żaden przykład, który opisuje taki aspekt historyjki użytkownika, który nie jest w niej opisany. Co więcej żadne dwa przykłady nie powinny opisywać tej samej cechy historyjki użytkownika.

3.3.4 Mając daną historyjkę użytkownika, kandydat potrafi napisać przypadki testowe, zarówno funkcjonalne jak i niefunkcjonalne używając technik czarnoskrzynkowych

W testowaniu zwinnym wiele testów jest tworzonych równolegle z tworzeniem oprogramowania. Tak samo jak programiści tworzą oprogramowanie w oparciu o historyjki użytkownika i kryteria akceptacji, testerzy tworzą testy

również w oparciu o historyjki użytkownika i kryteria akceptacji. Niektóre testy, takie jak testy eksploracyjne oraz inne testy oparte na doświadczeniu są tworzone później podczas fazy wykonania testów (patrz podrozdział 3.3.4). Testerzy mogą wykorzystywać do projektowania testów tradycyjne techniki czarnoskrzynkowe, jak klasy równoważności, analiza wartości brzegowych, tablice decyzyjne oraz testy przejść między stanami. Na przykład analiza wartości brzegowych może zostać użyta do wyboru wartości testowych, gdy klient ma ograniczoną liczbę rzeczy, które może wybrać podczas zakupu.

3.3.5 Kandydat potrafi wykonać testy eksploracyjne, by wspomóc testowanie w projekcie zwinnym

Najlepsze rezultaty osiąga się, gdy testowanie eksploracyjne uzupełnia się innymi technikami opartymi na doświadczeniu, jako części reaktywnej strategii testów, którą łączy się z innymi podejściami do testów, takimi jak analityczne testowanie oparte na ryzyku, analityczne testowanie oparte na wymaganiach, testowanie oparte na modelu czy testowanie przeciwregresywne. Strategie testowania oraz ich łączenie opisane są w sylabusie Poziomu Podstawowego. W testowaniu eksploracyjnym projektowanie i wykonywanie testów wykonuje się równocześnie, w oparciu o wcześniej przygotowaną kartę testów. Karta testów dostarcza warunków testowych, które powinny zostać pokryte podczas ograniczonej czasowo sesji testowej. Podczas testów eksploracyjnych, wyniki ostatnich testów prowadzą do następnych testów. Te same techniki biało- i czarnoskrzynkowe, których używa się przy wykonywaniu testów zaprojektowanych, mogą być używane również w testach eksploracyjnych. Karta testów może zawierać następujące informacje:

- Aktor: Zamierzony użytkownik systemu.
- Cel: Temat karty, w tym szczególny cel, jaki aktor chce osiągnąć, tzn. warunki testowe.
- Konfiguracja: Co powinno być przygotowane, by rozpocząć wykonywanie testów.
- Priorytet: Względna istotność tej karty, oparta na priorytecie powiązanej historyjki użytkownika lub poziomie ryzyka.
- Odnośniki: Specyfikacja (np. historyjka użytkownika), ryzyko i inne źródła informacji.
- Dane: Jakiekolwiek dane potrzebne do wykonania karty.
- Czynności: Lista pomysłów na to, co aktor może chcieć wykonywać w systemie (np. “Zalogować się do systemu jako superużytkownik”), a także co mogłoby być intersujące do przetestowania (zarówno w testach pozytywnych jak i negatywnych).
- Uwagi wyroczni testowej: Jak oceniać produkt, by określić poprawne wyniki (np. uchwycić co pojawia się na ekranie i porównać to z tym, co napisano w instrukcji użytkownika).
- Wariacje: Alternatywne akcje i oceny uzupełniające pomysły opisane w czynnościach.