Zarządzanie testami

1. Organizacja testów

Kandydat potrafi omówić zalety i wady niezależnego testowania.

Kandydat potrafi wskazać zadania kierownika testów i testera

1. Niezależne testowanie
   1. Niezależni testerzy (osoby z zewnątrz, beta-testerzy, osobno zatrudniona firma testerska) często zwiększają skuteczność wykrywania defektów (autor i tester w firmie mogą być zbyt mocno zaangażowani w tworzenie produktu i pewne rzeczy po prostu im umkną)
   2. Z drugiej stronie – niezależność nie zastępuje znajomości produktu – osoba niezależna będzie potrzebowała czasu na poznanie systemu
   3. Poziomy niezależności
      1. Brak – gdy testują programiści swój kod (Testy Modułowe (jednostkowe – UT))
      2. Niezależni programiści i testerzy zatrudnieni w naszym zespole – część pracowników na umowie – część kontraktowych
      3. Niezależny zespół testerów w ramach projektu
      4. Niezależni testerzy w innych działach
      5. Niezależni testerzy spoza firmy (zatrudniane sa cale firmy z zewnątrz do testowania)
   4. Najlepiej sprawdza się mieszanka powyższych punktów
   5. Poziom niezależności zależy też od cyklu życia wytwarzania oprogramowania (np. w krótkich 2tygodniowych sprintach może być ciężko zapewnić niezależność testerom, ale już w 2miesięcznych cyklach np. testów systemowych już łatwiej)
   6. Zalety:
      1. Większa szansa na wykrycie innego rodzaju awarii niż wykrywają wewnętrzni testerzy
      2. Możliwość ponownej weryfikacji wymagań, UserStories przez ludzi z zewnątrz
      3. Nie wywiera na nich presji nikt z firmy – nie będą się bali powiedzieć szefowi projektu co sądzą na temat projektu
2. Zadania kierownika i tetera (jest o to pytanie)
   1. Kierownik testów
      1. Opracowuje strategię testów i politykę testów
      2. Planuje testy z uwzględnieniem kontekstów, zapoznaje się z celami testów, czynnikami ryzyka, szacowania ile zajmie wdrożenie testów i ich wykonanie
      3. Planuje zarządzanie defektami (jakiego Bugtrackera używać, jak go używać)
      4. Sporządza i aktualizuje plan testów
      5. Koordynuje realizację planu testów
      6. Broni i przedstawia punkt widzenia testerów
      7. Inicjuje procesy testów wymienione w punkcie 1 (analiza, planowanie, implementacja, nadzór, projektowanie, itd.) – czyli to on organizuje te spotkania i dba o to by wnioski z nich były dobrze spisane
      8. Przygotowuje i dostarcza raporty z postępu testów
      9. Zarządza planem testów i harmonogramem ich wykonania
      10. Decyduje w jakim środowisku (Windows, Mac, Linux) będziemy testować
      11. Dba o rozwój testerów
   2. Tester
      1. Przegląd planu testów
      2. Ocena i analiza wymagań, user stories pod kątem testowalności
      3. Identyfikowanie i tworzenie TestCaseów (TC)
      4. Projektowanie i konfiguracja środowiska do testów (np. instalacja Selenium, pisanie skryptów testowych)
      5. Pozyskuje dane do testów (nasze 25 częstotliwości do radia)
      6. Tworzy szczegółowy harmonogram testów
      7. Wykonuje testy
      8. Automatyzuje testy
3. Planowanie i szacowanie testów

Kandydat potrafi scharakteryzować cel i treść planu testów. Kandydat potrafi rozróżnić poszczególne strategie testów. Kandydat potrafi podać przykłady potencjalnych kryteriów wejścia i wyjścia. Kandydat potrafi wykorzystać wiedzę na temat ustalania priorytetów oraz zależności technicznych i logicznych do zaplanowania wykonania określonego zbioru przypadków testowych.

Kandydat potrafi wskazać czynniki wpływające na pracochłonność testowania.

Kandydat potrafi wyjaśnić różnicę między dwiema technikami szacowania:

techniką opartą na miarach i techniką ekspercką.

(przykład - https://testerzy.pl/artykuly/dokumentacja-testowa-plan-testow)

1. Cel i treść
   1. Określenie zakresu celów testowania i ich ryzyka
   2. Ogólne podejście do testowania
   3. Plan jak będziemy integrować nasze testy i kiedy będziemy testować w naszym cyklu wytwarzania oprogramowania (np. w 2 tygodniowym sprincie – co i jak i kiedy będziemy robic)
   4. Decyzje kadrowe (kto gdzie co będzie testował) i tematyczne (jak testować co)
   5. Plan analizy, planowania, projektowania testów – kiedy i co będziemy robić na tych spotkaniach
   6. Określamy budżet
   7. Ustalamy poziom szczegółowości dokumentacji
2. Strategie testów i podejście testowe (było pytanie!)
   1. Analityczna
      1. Analiza określonego ryzyka (analiza wymagań). Testujemy w oparciu o ryzyko lub analizę wymagań
   2. Oparta na modelu
      1. Testujemy daną funkcję, zachowanie systemu w oparciu o jakąś charakterystykę (niezawodność) – sprawdzamy coś, co możemy zamodelować
   3. Metodyczna
      1. Podchodzimy do testowania z przygotowanym zbiorem testów lub warunków testowych (listy kontrolne) – krok po kroku testujemy
   4. Zgodna z procesem
      1. Testujemy pod kątem standardów, dokumentacji, wynikających z norm branżowych, procesów organizacji
   5. Kierowana
      1. Na podstawie podpowiedzi i porad użytkowników końcowych (klientów)
   6. Minimalizująca regresję
      1. Zakłada ponowne użycie już istniejących testów do nowych części systemu do testowania. Testy są pisane bardziej ogólnie by dopasować do nich więcej realnych przypadków
   7. Reaktywna
      1. Reagujemy na zdarzenia w systemie, nie mamy z góry ustalonego planu, testy tworzymy na podstawie obserwacji zachowania systemu
   8. Podsumowanie – w zależności od tego co testujemy, jaką wiedzę mamy o tym co testujemy, czas, budżet – dobieramy różne z powyższych strategii
3. Kryteria wejścia i wyjścia (Definition of Ready, Definition of Done)
   1. Jest to zbiór informacji potrzebnych do tego by móc zacząć coś robić (lub o czymś rozmawiać) i kiedy skończymy nad tym pracę
      1. Przykład klub książki – Definition Of Ready by wziąć udział w spotkaniu to przeczytanie książki o której będzie mowa
      2. Definition of Done – kończymy spotkanie ustalając która z postaci z książki jest najbardziej winna śmierći głównego bohatera. Argumentujemy listą w podpunktach i głosowaniem 😊
   2. Typowe kryteria wejścia do testowania
      1. Wymagania są gotowe, UserStories są gotowe
      2. Dostępność środowiska testowego (nasze stacje kontroli pojazdów)
      3. Dostępność niezbędnych narzędzi (mierniki lakieru)
      4. Dostępność danych testowych (nasze 25 częstotliwości radia)
   3. Typowe kryteria wyjścia (Definition of Done)
      1. Koniec wykonywania zaplanowanych testów
      2. Osiągnięcie zakładanego poziomu pokrycia testami
      3. Zejście poniżej ustalonej liczby defektów akceptowalnej przez klienta
      4. Zadowalające wskaźniki jakości, wydajności, użyteczności
4. Harmonogramy testów
   1. Ustalamy kolejność uruchamiania testów (np. najpierw te najważniejsze później mniej ważne bo jak ważne obleją to tamte nie mają znaczenia)
   2. Patrzymy na zależności między testami (test silnika musi przejść żebyśmy mogli zacząć test z jazdy samochodem)
   3. Patrzymy na czas trwania testu (jeśli wszystkie testy trwają ponad 2-3godziny może warto uruchamiać je w nocy żeby wyniki było gotowe następnego dnia rano)
   4. Patrzymy na zasoby – czasami nie jesteśmy w stanie uruchomić wszystkich testów na raz albo połowa zespołu ma urlop w lipcu
5. Czynniki wpływające na czasochłonność testowania
   1. Charakterystyka produktu
      1. Czy produkt musi być bezpieczny (ryzyko – testujemy hamulce – lepiej poświęcić tym testom więcej czasu)
      2. Rozmiar produktu (testowanie dużych skomplikowanych systemów trzeba podzielić i liczyć się z ogromną liczbą testów)
      3. Wymagania jakościowe (system musi spełniać szereg wymagań jakościowych – np. nie psuć się przez 2000h działania silnika)
      4. Wymagana szczegółowość dokumentacji (dokumentację pisze się wolno, żmudnie, często się ona zmienia, koszmar)
      5. Zgodność z przepisami – zależymy wtedy od przepisów i dostosowanie się do nich może trwać (samochód musi mieć nalepkę jakości)
   2. Charakterystyka procesu
      1. Młode niedojrzałe firmy mogą testować dłużej
      2. Nie-iteracyjne metody pisania kodu sprzyjają dłuższym testom
      3. Język programowania jakiego używamy – jak bardzo jest powszechny i ilu mamy specjalistów w firmie
      4. Presja czasu (klient chce wszystko na wczoraj i to wpływa na presję wytwarzaną na testerach)
      5. Proces testowy – jak wiele rzeczy robimy sztywno a jak wiele elastycznie i szybciej. Zawsze trzeba wybrać kompromis między szybko a dokładnie
   3. Czynniki ludzkie
      1. Doświadczenie testerów we zespole
      2. Umowy współpracy zespołu i umiejętności managera
   4. Rezultaty testów
      1. Liczba znalezionych błędów
      2. Liczba wymaganych poprawek (im więcej tym dłużej może trwać testowanie każdej poszczególnej)
6. Szacowanie testów – szacujemy ile nam mniej więcej zajmie napisanie testów
   1. Informacja ważna dla managerów i budżetu
   2. Szacujemy bo nikt dokładnie nie wie ile pracy mu to zajmie
   3. Technika oparta na miarach
      1. Porównujemy ile czasu zajęło wykonanie podobnych testów we wcześniejszych projektach lub na podstawie jakichś przyjętych standardowych jakości (np. 10 UT na dzień, 2TC na dzień itd.)
   4. Technika ekspercka
      1. Ekspert (architekt testów, lider, jakiś kozak w firmie) ustala z góry ile dane zajęcie powinno zająć testerowi (testerom)
   5. Raporty pomagają w takim szacowaniu na przyszłość – dlatego w testowaniu tak dużo i często wszystko raportujemy

1. Monitorowanie testów i nadzór nad testami

Kandydat pamięta miary stosowane w odniesieniu do testowania. Kandydat potrafi podsumować cele i treść raportów z testów oraz wskazać ich odbiorców.

1. Celem monitorowania jest gromadzenie i udostępnianie informacji o przebiegu testów i ich pisania
2. Najlepiej zbierać te dane automatycznie podczas zbierania metryk
3. Nadzorując testy odpowiadamy na pytanie „gdzie jesteśmy w projekcie” i jak szybko idziemy do przodu
4. Poprzez nadzór musimy często ponownie ustalać priorytety dla testów (coś zajęło nam za dużo czasu i teraz trzeba się spiąć i wyrzucić z harmonogramu mniej ważne testy)
5. Dostosowywanie kryteriów wejścia i wyjścia dla poprawek (jakiś test należy dostosować bo zmieniła się specyfikacja tego co nim testujemy – doszedł 3 stan do przycisku start/stop)
6. Miary stosowane w testowaniu
   1. Realizacja harmonogramu i budżetu
   2. Aktualna jakość naszego produktu
   3. Czy nasze testowanie nadal jest odpowiednie
   4. Czy testowanie pozwala nam realizować cele systemu
7. Miary dotyczące testów
   1. Procent napisanych TC
   2. Procent przygotowania środowisk testowych
   3. Procent - Ilość wykonanych/niewykonanych TC
   4. Dane o defektach (ich ilość, liczba wykrytych i usuniętych)
   5. Współczynnik awarii
   6. Pokrycie testowe (na każdym etapie – UT, integracyjne, akceptacyjne itd.)
   7. Aktualne koszty testowania
8. Cel i treść raportów
   1. Blablablabla
   2. To co wyżej mierzymy – w raportach zapisujemy
   3. Raport z nadzoru testów
      1. Status testów
      2. Postęp w realizacji planu
      3. Aktualna mierzalna jakość systemu
      4. Co planujemy zrobić w następnym kroku
   4. Raport sumaryczny może mieć formę jak w ISTQB:
      1. Podsumowanie wykonanych testów
      2. Zdarzenia z przebiegu testów (co się ciekawego wydażyło)
      3. Informacje co poszło nie tak, co nam się nie udało
      4. Informacje które ryzyka się potwierdziły
      5. Miary i metryki
      6. Poziom spełnienia DoD
   5. Codzienne raporty zespołu – Daily Scrum (Stand up Meeting)
9. Zarządzanie konfiguracją

Kandydat potrafi podsumować, w jaki sposób zarządzanie konfiguracją

wspomaga testowanie.

1. Zapewnienie integralności między wersjonowaniem (w gicie) kodu, testów i dokumentacji
2. Nazywane SCM (System Configuration Management)
3. Aby to zapewnić należy zagwarantować
   1. Kod jest w gicie i mamy śledzenie kodu z wymaganiami (Traceability)
   2. Testy (TS, TC) są w gicie i też mamy śledzenie do wymagań (Traceability)
   3. Dokumentacja jednoznacznie identyfikuje fragmenty kodu i testów (Unikalne identyfikatory, praca dobrego bibliotekarza😊)
4. SCM należy zdefiniować na etapie planowania testów (1 ogólny wysokopoziomowy meeting lub seria meetingów). Czemu ? Bo później ciężko będzie się przestawić na inny system😊
5. Czynniki ryzyka a testowanie

Kandydat potrafi określić poziom ryzyka na podstawie prawdopodobieństwa i wpływu.

Kandydat potrafi rozróżnić czynniki ryzyka projektowego i produktowego. Kandydat potrafi opisać na przykładach potencjalny wpływ analizy ryzyka produktowego na staranność i zakres testowania.

1. Ryzyko – prawdopodobieństwo że coś pójdzie nie tak, co sprawi że nasze zadanie (zadania) nie będą wykonane na czas lub w ogóle.
2. Ważną częścią analizy ryzyka jest jej wpływ na system (ryzyko że śrubki nie dojdą na czas jest duże, ale jego wpływ na postęp budowy auta mały)
3. Czynniki ryzyka
   1. Produktowe
      1. Ryzyko, że program nie będzie działał zgodnie ze specyfikacją
      2. Architektura nie będzie dość dobra by system spełnił wymagania niefunkcjonalne (multipla – jeździ – ale jest brzydka😊)
      3. Ryzyko błędnych obliczeń w danych warunkach (np. czujnik parkowania w mocnym deszczu)
      4. Zbyt długi czas odpowiedzi systemu w danej sytuacji (np. tempomat za późno postanowi zahamować)
      5. Zbyt nieczytelne komunikaty dla użytkownika na wyświetlaczu
   2. Projektowe
      1. Ryzyko, że nasz harmonogram zostanie zablokowany przez wystąpienie opóźnienia w jednym z kroków (nie doszły tłoki – silnik musi czekać)
      2. Błędne oszacowanie finansowe
      3. Zmiany wprowadzone w ostatniej chwili
   3. Organizacyjne
      1. Zespół ma małe doświadczenie – ryzyko że nie wyszkoli się odpowiednio szybko
      2. Skłócony zespół, architekt nie lubi managera 😊
      3. Brak beta-testerów, łańcucha dostaw tłoków i itd.
   4. Polityczne
      1. Problemy w komunikacji
      2. Brak inicjatywy, środowisko niesprzyjające podejmowaniu własnych inicjatyw
   5. Techniczne
      1. Niedostateczne doprecyzowanie wymagań
      2. Presja czasu
      3. Późno udostępnione środowisko testowe
      4. Nowy język programowania
      5. Dług techniczny 😊
   6. Dostawcy
      1. Brak kluczowego produktu, jest tylko w Azji
      2. Dostawca nie ma pozwolenia na handel w Unii 😉
      3. Jezu jakie to głupie
4. Ryzyko a jakość produktu
   1. Znając ryzyko możemy podjąć odpowiednie decyzje testowe odnośnie planu testów
   2. Ustalić odpowiednie priorytety
   3. Zadbać o dodatkowe procesy nie związane z testowaniem (np. magazynowanie tłoków już w lutym 😊
   4. Tworzenie planów awaryjnych
   5. Ciągła reewaluacja ryzyka
   6. Możliwe zmniejszanie ryzyka przez odpowiednie decyzje zapobiegawcze
5. Zarządzanie defektami

Kandydat potrafi sporządzać raporty o defektach zawierające informacje na

temat defektów wykrytych podczas testowania.

1. Aby kontrolować ilość defektów, ich priorytety, opisy, przypisane role blabla należy używać narzędzi typu Bugtracker (np. nasze trello, Jira, Bugzilla, Mantis itp.)
2. Raport o defekcie musi być odpowiednio szczegółowy
   1. Co nie działa
   2. Co zrobić by zasymulować sytuację w której występuje błąd
   3. Jakie jest oczekiwane działanie
   4. Priorytet
   5. Identyfikator
   6. Przypisany programista lub zespół do naprawy
   7. Powiązane wymaganie
   8. Data zgłoszenia
   9. Zalecenia