**PROCES TESTOWY**

żródło: <https://devenv.pl/jak-wyglada-dobry-proces-testowy/>

# Jak wygląda proces testowy? Część 1

Przez [Janina Adamiec](https://devenv.pl/author/janinaadamiec/) 30 lipca 2018

**Proces testowy** – to jest to o czym chciałabym dzisiaj Wam opowiedzieć. W [poprzednich artykułach](https://devenv.pl/kategoria/devquality/) opisywałam niektóre elementy tego procesu. Teraz postaram się dodać jeszcze parę informacji i zebrać wszystko w całość.

**Aby testowanie dawało dobre rezultaty, musi być zaplanowane i usystematyzowane.** Można powiedzieć, że takie ułożenie testowania to proces testowy. Nie ma jednak jednego jedynego, szablonowego procesu testowego. W praktyce, to co jednych projektach się sprawdza, w innych zupełnie nie będzie miało prawa bytu. Można jednak spróbować wydzielić pewne czynności które będą wspólne.

## Podstawowy proces testowy

Zwykle podstawowy proces testowy opisywany jest jako zestaw czynności takich jak:

* planowanie
* kontrola
* analiza
* projektowanie
* implementacja
* wykonanie
* ocena i raportowanie
* zamykanie czynności testowych

Wprawdzie wypisałam te czynności jedno pod drugim ale nie znaczy to, że one muszą odbywać się jedna po drugiej (chociaż czasem narzuca to metodologia w której pracujemy, lub też rodzaj testów które aktualnie wykonujemy wykonujemy).

## Planowanie

Proces testowy zaczynamy od planowania. Z planowaniem testów na początku projektu jest trochę jak z zakupami. Musimy pomyśleć – co będzie nam potrzebne, co musimy wcześniej zrobić lub mieć by móc iść na zakupy, co mamy kupić itp. To samo dotyczy planowania testów. Przed rozpoczęciem testowania musimy przecież wiedzieć co jest nam potrzebne (np. jakie narzędzia) i jakie czynności powinniśmy wykonać. Powinniśmy się również na tym etapie zastanowić jak będziemy mierzyć nasze testy (wspomniane w poprzednich artykułach metryki).

To jest ten moment kiedy decydujemy jakie poziomy, typy i techniki testów będą wykorzystywane oraz jakie chcemy osiągnąć cele na poszczególnych etapach testowania. Planować możemy a nawet powinniśmy także w trakcie wprowadzania zmian, podejście do testów może w trakcie trwania projektu ulec zmianie. Być może jakaś funkcjonalność będzie wymagała zupełnie innego nakładu pracy niż początkowo nam się wydawało i okaże się, że ponownie trzeba zaplanować proces testowy.

Oczywiście zespół planujący bądź osoba planująca nie może zapomnieć o wzięciu pod uwagę ograniczeń, ryzyk, wymagań oraz obowiązków – specyficznych dla projektów, technologii itp. Jak i dodatkowego czasu na wszystkie elementy około testowe, o których często się zapomina tj. dokumentacja, konfiguracja środowisk itp.

**Faza planowania ma przygotować produkt do fazy testowej**, tak by mieć pewność, że to co zostanie dostarczone będzie satysfakcjonujące i spełniające wymagania klienta.

### No dobrze a co to oznacza w praktyce?

Praktycznym przełożeniem teorii na praktykę jest spisanie test planu. Test plan pomaga nam spiąć całą wiedzę o testach w całość.

W test planie możemy zawrzeć informacje takie jak:

* cel testów i ich ograniczenia
* elementy testowe i ich wersje
* części aplikacji które zostaną poddane testom, odniesienia do wymagań i / lub specyfikacji
* części aplikacji które nie zostaną poddane testom (wraz z podaniem powodu dlaczego nie)
* podejście do testów (np. testujemy tylko eksploracyjnie), typy testów, metody testowania (np. czy manualnie czy automatycznie) itp
* kryteria akceptacji / odrzucenia – określenie kryteriów, które będą używane w celu ustalenia czy aplikacja przeszła pomyślnie test czy też nie
* rzeczy które są dostarczane razem z testami np. plan testów (sam dokument), przypadki testowe, skrypty testowe, dzienniki błędów, raport z testów
* środowisko testowe – opisanie środowiska testowego, jego parametrów itp
* harmonogram – określ ramy czasowe, najważniejsze milestony, warto dodać link do szczegółowego harmonogramu prac jeśli jest taki dostępne
* skład teamu – można opisać skład zespołu, wymienić obowiązki każdego zespołu / roli / osoby.
* zagrożenia i ryzyka – wypunktować zidentyfikowane zagrożenia, opisać plan niwelujący dane ryzyko i plan awaryjny w razie jego wystąpienia
* założenia i zależności – wymienienie założenia, które zostały poczynione podczas przygotowania dokumentu, wymienić zależności (czy od czegoś zależy projekt np. współpraca z innymi dostawcami)
* osoby zatwierdzające plan

Taki plan nie powinien być pisany raz na projekt. Powinien być zawsze aktualizowany w ramach zmieniających się warunkach projektowych. Pamiętaj – przestarzały i nieużywany dokument nikomu nie przyniesie korzyści.

## Kontrola

Może was zdziwi, że opisuję kontrolę jako drugi punkt a nie gdzieś na końcu – kontrola odbywa się przez cały czas. Pod pojęciem kontrola – kryją się dwa kolejne. Monitoring i nadzór.

### Monitorowanie

**Monitorowanie to tak naprawdę sprawdzanie czy idziemy w dobrym kierunku**, czy wszystko idzie zgodnie z zakładanym planem. Przykładem który przychodzi mi do głowy jest scrum. Mamy boarda na którym mamy zaplanowaną pracę na sprint, osoba odpowiedzialna monitoruje czyli sprawdza czy wszystkie taski idą zgodnie z planem. Czy nie ma opóźnień i czy wszystkie taski zostaną dostarczone na czas – czyli na koniec sprintu a w dłuższej perspektywie czy zespół dostarczy produkt na czas,

Dobrze, a w jaki sposób możemy monitorować projekt? Możemy użyć metryk takich jak:

* stosunek ilości zadań zakończonych do zadań zaplanowanych w danym okresie czasu np. w sprincie
* stopień pokrycia np. kodu testami
* liczba pozytywnie zakończonych przypadków testowych
* liczba zgłoszonych błędów czy liczba naprawionych błędów

Tak naprawdę tych metryk jest sporo, dlatego tylko wspominam w przyszłości być może powstanie o tym osobny artykuł.

#### Raportowanie

Do monitorowania projektu zaliczamy też raportowanie na przykład raportowanie testów. Po w trakcie jak i po zakończonych testach dobrze jest stworzyć raport z testów. Poziom jego sformalizowania i forma często jest zależna od projektu. Nie mniej warto wiedzieć co powinno się znaleźć w takim raporcie:

* cel stworzenia dokumentu, kto jest adresatem
* krótki opis aplikacji, systemu, produktu
* zakres testów czyli co podlegało testom a co nie
* opis środowiska testowego – typy testów jakie zostały przeprowadzone np. smoke testy, testy eksploracyjne, testy wydajnościowe, testy regresji
* metryki np. pokrycie kodu testami, wyniki testów, ilość wykonanych test casów z rozpisaniem jakie były wyniki, ilość zgłoszonych błędów
* jakie i czy zostały wszystkie kryteria wejściowe spełnione  
  opis krytycznych problemów i sposobów radzenia sobie z nimi – takie lessons learned
* podsumowanie
* rekomendacje

Taki raport tak jak wspomniałam może być dowolnie modyfikowany, najważniejsze jest by odbiorca uzyskał jasną i czytelną informację o zakończonych testach.

### Nadzór

A co jeśli okazuje się, że coś jest nie tak? Gdy widzimy, że projekt nie idzie zgodnie z planem i mamy opóźnienia? Oczywiście należy działać tak by wszystko ponownie szło zgodnie (czasem z nowym) planem. I tutaj mamy do czynienia z nadzorem.

Wyobraźmy sobie następującą sytuację – raport wykazał iż w tej iteracji zgłoszono znacznie więcej błędów niż zwykle, co wtedy? Wtedy na przykład należy przeprowadzić analizę źródła problemu a w następnej kolejności zastanowić się nad kolejnymi krokami – np czy konieczne jest przedłużenie fazy testowej czy też konieczna będzie redukcja ilości zadań zaplanowanych na następny sprint po to by programiści zdążyli naprawić błędy.

Porównując monitoring i nadzór możemy stwierdzić monitoring to taka obserwacja, a **nadzór już wiąże się z podejmowaniem konkretnych decyzji i działań**.

## Analiza

**Podczas fazy analizy określamy co powinno być przetestowane**, na tym etapie równie definiujemy warunki testowe. Brzmi enigmatycznie ale w tej fazie nasze założenia nabierają tak naprawdę realnych kształtów.

W pierwszym kroku należy określić przedmiot testów i elementy testowe, w kolejnym kroku należy dokonać przeglądu podstawy testów (które będą różne w zależności od poziomu testów), celów testowania czy też dokonać analizy ryzyka. Dzięki czemu uzyskamy coś na kształt wskaźnika, miarę osiągniętego sukcesu, a zarazem będzie to pewna część kryteriów wyjściowych.

### To jeszcze raz kilka słów o test case’ach

Jak wiecie już z mojego artykułu o [przypadkach testowych](https://devenv.pl/przypadki-testowe-co-kazdy-programista-wiedziec-powinien/) – mogą one być tworzone na różnych poziomach ogólności. Im wyższy poziom tym więcej będziemy mieć warunków testowych. Ważne jednak jest by były powiązane z podstawą testów. Dzięki temu będziemy mogli mierzyć nasze metryki np. stopień pokrycia kodu.

Każdy przypadek testowy tak naprawdę możemy przełożyć na konkretny element np. linię kodu, funkcję, moduł itd który pokrywamy konkretnym testem. Rozpatrując każdy przypadek możemy rozpatrzyć możliwość wystąpienia jakiegoś ryzyka, które potem możemy odnieść do wymagania. Gdy mamy przypadek testowy – możemy go odnieść nawet do kilku wymagań.

Na przykład mając 3 metody składające się na jedno wymaganie np.

* getStatus(),
* setStatus(),
* removeStatus()

Nasze wymaganie zostanie pokryte w 100% tylko gdy wszystkie 3 metody zostaną przetestowane.

Podsumowując, w ramach procesu analizy analizy, analizujemy wymagania, ryzyka, specyfikację, kod by dowiedzieć się co chcemy testować. Kolejnym krokiem jest projektowanie testów.

## Projektowanie testów

Dzięki analizie testów wiemy już “co testować” czyli zidentyfikowaliśmy obiekty testów. **Projektowanie testów odpowie nam na pytanie “jak” testować.**

Czynności jakie powinniśmy wykonać w ramach projektowania testów:

* wybrać poziom szczegółowości test casów dla konkretnych testowanych obszarów
* wybrać techniki projektowania (o tym też pisałam w poprzednim artykule)
* tworzenie przypadków testowych

Ogólne przypadki testowe mogą być pisane na przykład gdy nie znamy jeszcze dokładnie wymagań, lub są one mało szczegółowe. Przy wykonywaniem tego rodzaju test casów tester musi odznaczać się doświadczeniem, znajomością technik czy samej aplikacji. Przez to, że test casy są ogólne – jesteśmy w stanie łatwiej uzyskać większy stopień pokrycia. To podejście sprawdza się w projektach z mniejszym naciskiem na dokumentację.

Bardziej szczegółowe przypadki testowe są pisane często w “rygorystycznych” projektach, gdzie wszystko musi być skrzętnie dokumentowane lub też w projektach z mniej doświadczonymi testerami w zespole. Często ułatwiają transfer wiedzy projektowej. Bardziej ogólne przypadki testowe,zwykle są tworzone na początku a w ramach rozwoju projektu, przypadki są doprecyzowywane i rozwijane.

Przypadki testowe powinny być: powtarzalne, niezależne, weryfikowalne

Gdy mamy stworzone przypadki testowe, koniecznym jest nadawanie im priorytetów oraz przypisać do danego poziomu testów. Ostatnim elementem projektowania testów jest zaplanowanie środowiska testowego oraz niezbędnej architektury i narzędzi.

Kolejną fazą jest implementacja i wykonanie testów ale o tym i o reszcie procesu – napiszę w następnym artykule

## Podsumowanie

A teraz chciałabym krótko podsumować to o czym dzisiaj czytaliście. Chyba trochę pod skórą czujecie, że wszystkie te procesy się przenikają ze sobą i są ze sobą ściśle powiązane.

Możemy napisać test plan ale bez pozostałych elementów procesu nie będzie on wiele zawierał ponieważ nie będzie podstawowych informacji. Bez monitoringu nie będzie wiadomości o stanie projektu, ale aby powstał raport musimy wiedzieć co i jak testować. Nie da się opisać uniwersalnego procesu testowego.

Mamy jasno określone wskazówki jak on powinien wyglądać – ale tak jak każda aplikacja jest inna, każdy zespół deweloperski jest inny, tak każdy proces testowy jest inny. To co sprawdzi się w zespole A nie jest powiedziane, że sprawdzi się w zespole B. I to co ważne – **procesy są dla nas, a nie my dla procesów.** Procesy mają nam ułatwiać życie a nie je utrudniać. O czym często zapominamy projektując proces testowy w naszych projektach.

źródło: https://devenv.pl/jak-wyglada-dobry-proces-testowy-2/

# Jak wygląda dobry proces testowy? Część 2

Przez [Janina Adamiec](https://devenv.pl/author/janinaadamiec/) 17 sierpnia 2018

Dzisiaj chciałabym kontynuować swój poprzedni [post](https://devenv.pl/jak-wyglada-dobry-proces-testowy/) o procesie testowym, w którym dowiedzieliście się jak wygląda planowanie, kontrola, analiza i projektowanie testów. W tym artykule dowiecie się **jak wyglądają kolejne etapy procesu testowego takie jak – implementacja, wykonanie, ocena i raportowanie i zamykanie czynności testowych.** Na koniec postaram się wszystko zebrać w całość i krótko podsumować. No to zaczynajmy.

Ostatnio skończyliśmy opisywać nasz proces testowy w miejscu, gdzie mieliśmy stworzone przypadki testowe, nadaliśmy im priorytety i określiliśmy poziomy testów. Kolejnym krokiem jest Implementacja testów.

## Implementacja

Zwykle, większość osób myśląc o testowaniu i o tym co tester robi na co dzień, myśli o tych czynnościach, które wykonywane są w fazie implementacji i wykonywania testów. Czym zatem jest implementacja testów? To jest ten moment, kiedy, nasze założenia wykonane we wcześniejszych fazach stają się faktem, realizacją.

**Na tym etapie zaczynamy implementować stworzone przypadki testowe.** Tak naprawdę w tym momencie możemy ten nasz przypadek testowy przełożyć na kod testu wykonującego nasz przypadek testowy bądź cały scenariusz testowy. Możemy jeszcze dokończyć ich piorytetyzację oraz określić jakie dane testowe będą nam potrzebne podczas ich wykonywania.

No właśnie dane testowe – już o nich wspominałam wcześniej w jednym z poprzednich artykułów. W kontekście tej fazy testów najczęściej mówimy o:  
– implementacji danych w konkretnych przypadkach testowych np. poprawne dane użytkownika w przypadku testowym związanym z logowaniem z użyciem prawidłowych danych logowania  
– implementacja generatorów danych – gdy nasz projekt wymaga dużej ilości danych, o określonej strukturze a na przykład dostępne generatory danych nie spełniają naszych oczekiwań i potrzeb.

Mówiąc o implementacji musimy jeszcze wziąć pod uwagę nasze środowiska testowe:  
– skonfigurować sprzęt, który będziemy używać np. serwery, czytniki, jeśli takowe będą potrzebne itp.  
– zainstalować i skonfigurować niezbędne oprogramowanie np. system operacyjny, bazy danych, środowisko uruchomieniowe  
– tworzenie atrap potrzebnych do wykonania testów, które na przykład używają modułu, który jeszcze nie został stworzony.

Czasami mówi się o fazie implementacji i wykonania testów, bo ta pierwsza faza zwykle jest ściśle powiązana z tą drugą. Zatem gdy mamy już nasze przypadki, dane i przygotowane środowiska testowe możemy zacząć myśleć o wykonaniu testów.

## Wykonanie testów

Wykonywanie testów jest ściśle związane z planem, który stworzyliśmy wcześniej. Czym właściwie jest wykonanie testu? To jest **proces wykonany na module lub systemie, dzięki któremu otrzymujemy wynik**. A tak bardziej obrazowo – gdy wykonujemy przypadek testowy – otrzymamy wynik, na podstawie którego stwierdzimy, że test przeszedł – jeśli rezultat był taki jak się spodziewaliśmy lub też nie – jeśli efekt jest inny od spodziewanego – czyli prawdopodobnie mamy do czynienia z jakimś błędem w kodzie. No właśnie – w kodzie, czy aby na pewno?

Osoby piszące test, również mogą się mylić. Na przykład, jeśli w wyniku pomyłki osoby tworzącej test spodziewamy się, że po wpisaniu poprawnych danych logowania i zatwierdzeniu ich, użytkownik nie zostanie zalogowany, w wyniku testu uzyskamy logowanie zakończone sukcesem uzyskamy wynik testu niezgodny z oczekiwaniami. W tym przypadku to nie jest usterka w programie, ale w teście.

Jaki jest z tego wniosek? Po otrzymaniu negatywnego wyniku testu przed zgłoszeniem go – powinniśmy upewnić się, że otrzymany wynik faktycznie wynika z błędu w kodzie programu a nie z błędnego testu.

Oczywistą rzeczą jest chyba to, że wyniki wykonanych testów muszą być logowane np. zapisywane w postaci raportu. Dzięki czemu możemy kontrolować wyniki i weryfikować, czy są one zgodne z postawionymi oczekiwaniami. To również może stanowić pewne zabezpieczenie na wypadek opóźnienia w projekcie. Mamy jasną informację dla klienta – wystąpiło opóźnienie, ponieważ wtedy i wtedy mieliśmy problemy, które należało rozwiązać. Takie dzienniki testów też powinny zawierać informacje jasno identyfikującą wersję testowanego oprogramowania oraz środowisko testowe.

W tej fazie jeszcze możemy powtarzać czynności testowe np. jeśli stwierdzimy rozbieżności – by upewnić się, że faktycznie to błąd występuje, albo ponownie wykonać test by potwierdzić naprawę błędu, ponownie uruchomić testy po zmianie w testach itp.

Oczywiście założony plan uwzględnia pewne odstępstwa tak by tester mógł przeprowadzić jakieś dodatkowe testy, jeśli wymaga tego sytuacja albo coś szczególnie go zaintryguje.

## Ocena i raportowanie

Jedną z ostatnich faz jest ocena i raportowanie – tak naprawdę **polega na określeniu czy uzyskane wyniki są zgodne z celami testowania.** Ocena powinna być wykonana dla każdego poziomu testów.

Ocena wyników tworzona jest na podstawie monitorowania postępów w testach oraz spełnieniu określonych kryteriów, które zostały zdefiniowane wcześniej. Co to oznacza w praktyce? Moglibyśmy przyjąć, za kryterium wyjścia takie warunki:  
– wszystkie znalezione w danej fazie błędy muszą zostać naprawione  
– wszystkie błędy znalezione w danej fazie, oznaczone jako blocker i critical muszą zostać naprawione – stopień pokrycia kodu testami musi być większy niż 80%  
I tym podobne.

Definiując kryterium wyjścia należy być ostrożnym i brać pod uwagę czynniki poboczne np. długość trwania projektu. Poziom i kategoryczność kryteriów mogą być różna. Część kryteriów wiadomo jest „zero-jedynkowa”, bo nie można wypuścić na produkcję błędów z niedziałającą kluczową funkcjonalnością, ale czasem w sytuacji, gdy spełnienie kryteriów, w określonej sytuacji jest trudne – możliwe jest redefiniowanie kryteriów wyjścia.

Dlaczego tak? W trakcie procesu testowego często nasze założenia są weryfikowane przez rzeczywistość na przykład – zespół nie jest tak wydajny jak nam się wydawało, czy ma jakieś braki w wiedzy, których nie byliśmy świadomi i cały proces się wydłuża przez co nie może poświęcić tyle czasu na np. pisanie testów automatycznych przez co nie jesteśmy w stanie spełnić warunku o minimalnym procentowym pokryciu testami automatycznymi. Nie mniej – to jest broń obusieczna. Obniżenie kryteriów może wpłynąć na jakość produktu. Nie bez powodu mówi się często – albo coś się robi szybko albo dobrze.

Tak naprawdę, gdy mamy już wyniki naszych testów, często generowane automatycznie jesteśmy w stanie stworzyć raport na potrzeby klienta czy też osób wyższego szczebla o stanie projektu. Treść, forma, poziom ogólności zależy od projektu, od samej osoby zainteresowanej. Osoby mające wpływ na kierunek projektu zwykle chcą mieć bardziej szczegółowy raport, wraz z metrykami, z informacjami o znalezionych błędach itp. a czasem wystarczy krótki raport dla klienta czy idziemy zgodnie z planem.

## Czynności zamykające testowanie

Ostatnią fazą są czynności zamykające testowanie. Co kryje się pod tą nazwą? Ja bym powiedziała krótko – podsumowanie. To jest po prostu **podsumowanie tego co się działo w ramach procesu.** W ramach tego osoba odpowiedzialna ma do wykonania następujące zadania:  
– sprawdzić, czy wszystko zostało dostarczone zgodnie z planem i dokumentacją  
– zarządzanie błędami, czyli zamykanie notek, tworzenie nowych, edycja istniejących, jeśli jest taka potrzeba  
– utrzymywanie dokumentacji  
– prace niezbędne do utrzymania środowiska testowego i ponownego użycia w przyszłości  
– spotkanie retrospektywy  
– zachowanie artefaktów procesu testowego

Większość z tych rzeczy wydaje się oczywista, jednakże w natłoku pracy, często pod presją czasu może zdarzyć się, że zapominamy o rzeczach oczywistych. Taka weryfikacja pozwala nam upewnić się, że niczego nie przeoczyliśmy.

## A jak do tego ma się praktyka?

Pierwszym etapem jest planowanie, więc zacznijmy od test planu. W projektach Agile w porównaniu do modelu waterfall, test plan jest napisany, aktualizowany dla każdego releasu. Test plan zawiera typy testów wykonanych w iteracji, wymogi dotyczące danych testowych, infrastruktury, środowiska testowego. Przykładowy test plan zawiera:  
– zakres testów  
– funkcjonalności, które zostaną przetestowane  
– poziomy i typy testów  
– kryteria akceptacji  
– przemyślaną architekturę  
– zasoby  
– ryzyka  
– założenia i zależności

Zwykle proces testowania przebiega w 4 fazach:

Sprint 0, podczas tego sprintu, przeprowadzane są taski inicjujące obejmujące na przykład instalację narzędzi testowych, przygotowanie środowisk testowych, rezerwację zasobów. W tej iteracji możemy też uzgodnić założenia biznesowe, warunki brzegowe, zakres projektu, określić kluczowe wymagania i przypadki użycia, przedstawić zarys architektury, zidentyfikować ryzyka, oszacować wstępne koszty itp. To taka faza pozwalająca rozkręcić projekt. Przygotować się do niego, czyli przygotować plan, środowiska, przygotować dane testowe itp. itd.

Sprinty – Iteracje, to na tym etapie odbywa się większość testów. Ta faza jest zbiorem iteracji w których powstaje system, produkt itp. Drugą fazą są sprinty. Większość testów jest wykonywana w tej fazie. Ta faza to zestaw iteracji których efektem ma być przyrost w tworzonym rozwiązaniu. Testy podczas iteracji możemy podzielić na dwa rodzaje – testy potwierdzające i testy oparte na doświadczeniu. Testy potwierdzające koncentrują się na sprawdzeniu czy system spełnia założenia, drugi rodzaj testów pozwala wykryć problemy które z jakiegoś powodu mogły zostać pominięte lub zignorowane. Również testy potwierdzające możemy podzielić na testy wykonywane podczas developmentu jak i testy akceptacyjne. Oba z nich zwykle są (powinny być) zautomatyzowane, tak by móc wykonywać regularne testy regresji przez cały cykl tworzenia oprogramowania.

Celem tworzenia oprogramowania jest doprowadzenie naszego produktu do takiego stanu, że wiemy, że może być bez obaw używany przez użytkowników – czyli wypuszczenie produktu „live”. Czynności zawarte w tej fazie obejmują przeszklenie użytkowników, wprowadzenie na rynek produktu, tworzenie kopii zapasowych i przywracanie w razie potrzeby, dopracowywanie dokumentacji itp.

Ostateczna faza testów obejmuje pełne testy systemu i testy akceptacyjne. Na tym etapie testy powinny być już znacznie bardziej rygorystyczne by mieć pewność, że wszystko działa jak należy zanim aplikacja trafi na produkcję. Ostatnią fazą jest wersja produkcyjna

Ciekawym zobrazowaniem podejścia do testów jest kwadrat Agile.

## Kwadrant testowania Agile

Kwadrant dzieli cały proces na 4 części i pomaga zrozumieć jak przeprowadzone mogą być testy w Agile.

Część 01 – w tej części najbardziej skupiamy się na jakości kodu. Obejmuje testy jednostkowe i testy komponentów.

Część 02 – ta część zawiera przypadki testowe które są skierowane na spełnienie oczekiwań biznesowych jak i wspierających team. Ta część skupiona jest na wymaganiach, testy obejmują scenariusze, przepływy, testy prototypów, regresji czy testowaniem w parach.

Część 03 – przypadki które w tej części stosujemy zwykle pozwalają nam na zbudowanie zaufania do produktu. Zwykle są to testy eksploracyjne, testy akceptacyjne oraz testy użyteczności.

Część 04 – tutaj zwykle skupiamy się na aspektach niefunkcjonalnych takich jak na przykład wydajność, bezpieczeństwo, stabilność.

Ten diagram pokazuje jak wiele obszarów ma do obsłużenia osoby odpowiedzialne za jakość produktu.

## Podsumowanie

Tak naprawdę metodologia stosowana w projekcie nie wyklucza żadnego z powyższych kroków. W rzeczywistości w projektach wykorzystujących Agile zwykle dosyć dobrze są zdefiniowane zasady testowania i dokumentacji. **Testowanie jest podstawową i kluczową częścią każdego procesu tworzenia oprogramowania.**

Zasada jest prosta, proces testowy należy zacząć tak szybko jak to jest możliwe. Często wymaga to dużego zaangażowania ze strony klienta. Połową sukcesu jest odpowiednie zaplanowanie testów i przygotowanie scenariuszy testowych i danych testowych. Kod powinien być wystarczająco stabilny by móc przejść do dalszych etapów testowania. Można przeprowadzić obszerne testy regresji, aby upewnić się, że wszystkie błędy zostały naprawione i przetestowane. Głównie komunikacja między członkami zespołu jest kluczem do sukcesu.

Dzięki odpowiedniemu zaplanowaniu testów, możemy czerpać korzyści z dostarczania wysokiej jakości produktu za każdym razem. I pamiętajcie – procesy są dla nas – nie my do procesów. Każdy projekt jest inny, w jednym projekcie proces będzie wyglądał w ten sposób, w innych ulegnie on modyfikacjom.