

Requirements Based Testing - RBT

Skuteczne testowanie w oparciu o wymagania

Naszym głównym zadaniem jako menedżerów ds. jakości oraz testerów jest znajdowanie błędów w tworzonym oprogramowaniu. Na szczęście obecnie praca wielu z nas wykracza już poza ujawnianie oraz śledzenie "pluskw" i obejmuje bardziej krytyczne zagadnienia związane ze sprawdzaniem, czy oprogramowanie które będzie wytwarzać nasza firma (jeszcze przez jego finalnym wyprodukowaniem) spełni oczekiwania użytkowników. Aby tego dokonać wiele organizacji zaczęło wykorzystywać Testowanie Oparte Na Wymaganiach (RBT).

Dzięki zastosowaniu RBT praca zespołów testowych staje się efektywniejsza - testy bezpośrednio nawiązujące do konkretnych wymagań funkcjonalnych umożliwiają skuteczne dotarcie do źródła problemu poprzez bezpośrednie powiązanie z wymaganiami na dowolnym etapie cyklu życia aplikacji. W rezultacie otrzymujemy systematyczne i efektywne pokrycie obszaru testów, co sprawia, że w zasadzie testujemy to, co ma największe znaczenie dla naszego klienta.

Wyzwanie jakie napotykamy, wdrażając RBT w naszej organizacji, to dopasowanie go do istniejących procesów Działu Jakości i dotychczasowych praktyk testowania. Z własnego doświadczenia mogę zaproponować trzy praktyczne sugestie, jak ulepszyć podejście RBT:

- **Testujmy wcześniej i często**, tak aby testowanie stało się czynnością równoległą do procesu tworzenia, rozciągało na wszystkie role, uświadamiając wszystkim uczestnikom i sponsorom projektu znaczenie jakości
- **Testujmy z głową, nie instynktownie** - zapewniając metodyczność i powtarzalność w planie testów zwiększamy przewidywalność i mierzalność procesu testowania
- **Testujmy z wykorzystaniem metryk** - pozwoli to określić status produkcji i działalności IT, umożliwiając zarządowi wgląd w stan wszystkich projektów IT w firmie oraz właściwie ocenić nasz wkład i zaangażowanie

Kryzys jakości

Wiele analiz, w tym te opracowane przez Standish Group Chaos Report, opisuje dotychczasowe standardy w przemyśle IT: większość projektów IT wykracza poza założony czas oraz budżet. Niewystarczająca jakość wytwarzanego oprogramowania jest najważniejszym czynnikiem odpowiedzialnym za niepowodzenia i często prowadzi do przebudowania kodu, wpływając na zakres i jakość produktu finalnego. Ponowne prace nad tymi samymi fragmentami kodu wydłuża czas powstawania aplikacji i znacząco pochłania zasoby, także finansowe. Aby zmniejszyć zagrożenia wynikające z błędów, koniecznie musimy zdać sobie sprawę z coraz silniejszych tendencji dbania o jakość powstających aplikacji już od samego początku ich życia.

Doświadczenia firm z branży IT oraz badania trendów rynkowych jasno wskazują na dwa podstawowe powody niskiej jakości tworzonych aplikacji:

- źle sformułowane wymagania
- niewłaściwe ich pokrycie przez testy

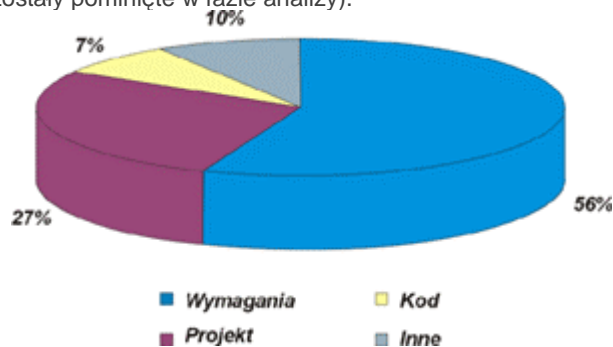
Wady w specyfikacji wymagań

Wielu z nas często spotyka się ze skargami użytkowników stwierdzających ewidentne braki w oprogramowaniu, które przeszło przez rygorystyczną kontrolę działu jakości i szereg testów. Najczęstszy powód? Wymagania były niewłaściwie sformułowane już od samego początku projektu.

Wymagania do rozbudowanych systemów często są ustalane w dwóch równoległych procesach, które następnie ewoluują równoległe w całym cyklu życia aplikacji. Wspomniane dialogi powstają w wyniku postawienia sobie

dwóch pytań: "co potrzebujemy zbudować?" oraz "co możemy zbudować?" Jakość tych dialogów często określa ostateczną jakość budowanej aplikacji.

Badania wykonane przez Jamesa Martina¹ pokazują, że 56 procent wszystkich zidentyfikowanych błędów w projektach informatycznych ma swoje korzenie w fazie tworzenia wymagań. To samo badanie wykazuje, że około połowę błędów powstaje w wyniku niewłaściwie napisanych, dwuznacznych, niejasnych i niepoprawnych wymagań. Druga połowa wad oprogramowania może zostać przypisana niewystarczającej specyfikacji (np. braku wymagań, które po prostu zostały pominięte w fazie analizy).



Dystrybucja Wad w Projektach Oprogramowania

Inne studia ujawniają podobne odkrycia:

- 82 procent przypadków wielokrotnie tworzonego kodu jest związanych się z błędami w wymaganiach²
- Problemy w obszarze wymagań w 44 procentach przypadków są powodem odwołania projektu³
- Tylko 54 procent początkowych wymagań projektowych jest właściwie zrozumiane⁴
- Tylko 45 procent zebranych wymagań zostaje właściwie użyte⁵

Podsumowując możemy stwierdzić, że dwa najważniejsze problemy związane z jakością wymagań to:

- wymagania i specyfikacje są niekompletne (z powodu braku informacji od użytkowników i sponsorów)
- zebrane wymagania są nienajlepszej jakości (głównie w wyniku braku zrozumienia potrzeb obu stron, znalezienia wspólnego języka oraz metod skutecznej weryfikacji zebranych założeń)

Problemy z pokryciem wymagań testami

Kiedy temat jakości pojawia się na końcowym etapie cyklu wytwarzania aplikacji, testowanie wymaga uprzedniego zakończenia fazy kodowania. W tym momencie zespół testerów znajduje się pod presją czasu, a ich zadaniem staje się jak najszybsze zweryfikowanie poprawności funkcjonalnej aplikacji. Ten etap często postrzegany jest jako wąskie gardło, które opóźnia wdrożenie aplikacji. W takich warunkach nie tylko trudnym jest upewnienie się, co do poprawności wymagań, ale przede wszystkim ułożenie takiego test planu, który zapewni poprawność i pełne pokrycie wymagań, jak również widoczność różnych aspektów jakościowych testowanej aplikacji. Oczywiście jest, że praca testera przy powyższych założeniach staje się nieefektywna i, co tu ukrywać, frustrująca.

Osobnym wyzwaniem jest osiągnięcie zadowalającego poziomu pokrycia testami. Złożoność dzisiejszych aplikacji stanowczo nam w tym nie pomaga. Coraz trudniejsze staje się wykonanie wszystkich możliwych scenariuszy, głównie ze względu na liczbę alternatywnych ścieżek przechodzenia przez aplikację. Jakakolwiek próba usystematyzowania (lub zautomatyzowania) procesu połączenia wszystkich możliwych przypadków

¹ James Martin, "An Information Systems Manifesto"

² Martin & Leffinwell

³ Standish Group: Chaos Report

⁴ Standish Group: Chaos Report

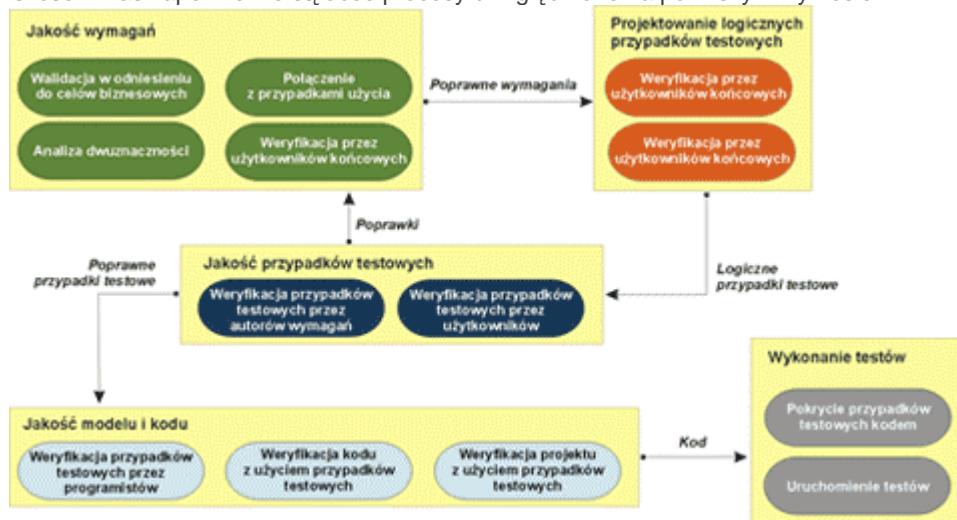
⁵ Jacobs

doprowadza nas po prostu to tak dużej liczby kombinacji scenariuszy testowych, że testowanie staje się bardzo trudnym i długotrwałym, a przez to i nieoptymalnym z punktu widzenia organizacji zadaniem.

W dodatku, w wielu przedsiębiorstwach zarządzanie zmianą, głównie ze względu na częstotliwość i skalę zmian w wymaganiach, staje się coraz trudniejsze. Stąd bez odpowiedniego wsparcia w zakresie zarządzania zmianą, często gubimy się jeśli chodzi o śledzenie powiązań pomiędzy zmieniającymi się wymaganiami oraz odzwierciedlenia tych zmian w przypadkach testowych.

Sprawdzone praktyki RBT

Podejście RBT, jeśli zostało właściwie zdefiniowane, adresuje bezpośrednio zależność pomiędzy testami funkcjonalnymi a źródłem znalezionych błędów. Nawet, jeśli nasza organizacja nie używa podobnych praktyk dzisiaj, większości z nas zapewne nie są obce procesy uwzględnione na poniższym wykresie.



Przebieg procesu RBT (kliknij aby powiększyć)

Równoległe z aktualnymi procesami testowymi, organizacje mogą spróbować zaadoptować następujące praktyki RBT:

- **Testujmy wcześniej i często**

W momencie, gdy mamy przygotowane wymagania oraz gotowy projekt i fragmenty kodu, przejrzymy je pod kątem celów biznesowych, przypadków użycia (use cases) i przypadków testowych (test cases). Starajmy się testować nasz projekt na jak najwcześniejszym etapie, ponieważ usterki zlokalizowane we wczesnych fazach rozwoju projektu są tańsze i łatwiejsze do usunięcia, w wyniku czego możemy się spodziewać znacząco mniejszej ilości "niespodzianek" na dalszych jego etapach. Testowanie powinno stać się czynnością równoległą do procesu tworzenia aplikacji. W ten sposób sponsorzy projektu w większym stopniu uświadomią sobie rolę jakości w całym procesie. Dzięki takiemu podejściu, testowanie będzie postrzegane już nie jako tzw. "wąskie gardło" w produkcji oprogramowania, a bardziej jako kluczowy czynnik w ogólnym procesie zapewnienia jakości powstających systemów.

Ponieważ sukces projektu informatycznego może być bezpośrednio powiązany z solidnym zrozumieniem i zdefiniowaniem wymagań, promowanie RBT staje się doskonałą wizytówką firm dbających o jakość swoich produktów. Najlepsze praktyki w procesie Jakości Wymagań RBT zawierają:

- Walidacja wymagań w odniesieniu do celów biznesowych - optymalizacja zakresu projektu przez zapewnianie, że każde wymaganie zaspokaja przynajmniej jeden biznesowy cel. Jeżeli brakuje powiązań pomiędzy wymaganiami a celami biznesowymi, konieczna jest weryfikacja tych pierwszych.

- Przegląd wymagań przez użytkowników - zmiany dokonywane w wymaganiach powinny być przeglądane i akceptowane przez użytkowników i odbiorców końcowych. W ten sposób upewnimy się, że dodatkowa praca wynikająca z modyfikacji wymagań nie będzie zmarnowana.
- Tworzenie przypadków użycia - każdy przypadek użycia powinien być powiązany z odpowiednim wymaganiem. Jeśli któryś z przypadków użycia nie ma takiego przyporządkowania oznacza, że wymagania są niekompletne.
- Wykorzystanie technik analizy języka - wyszukiwanie i poprawianie problematycznych wyrażań / sformułowań w opisie wymagań, pozwala na uniknięcie ich wieloznaczności i niejasności oraz wyeliminowanie pomyłek. Pozostawienie takich fraz bez poprawy powoduje iż mogą być one w późniejszych krokach interpretowane w różny sposób przez różne osoby, co z kolei może prowadzić do zakłopotania i błędów w kolejnych etapach. Dwuznaczne określenia produkują również "nietestowalne" przypadki użycia.

- **Testujmy z głową, nie instynktownie**

Większość przedsiębiorstw ceni doświadczonych testerów, licząc, że "wyłapią" błędy które inni, mniej doświadczeni testerzy mogliby przeoczyć. O ile jednak pojedynczy "ekspert ds. jakości" może być kuszącym ograniczeniem kosztów firmy, o tyle sytuacja taka zwiększa ryzyko polegania na instynkcie jednej osoby zamiast na grupowym racjonalnym podejściu.

Obecnie systematyczne i rygorystyczne planowanie przypadków testowych nie jest powszechnie stosowaną praktyką. Częściej spotykamy się raczej z podejściem bardziej intuicyjnym, polegającym na wyczuciu, co jednak w efekcie może prowadzić do nieprzewidywalnej jakości produktów końcowych. Doświadczenia firm stosujących podejście bazujące na RBT wskazują na fakt, że stosowanie metodycznego i systematycznego projektowania przypadków testowych jest najefektywniejszą polityką w zakresie jakości, zwłaszcza w perspektywie rozwoju firmy. Rygorystyczne i systematyczne zasady projektowania przypadków testowych uniezależniają proces kontroli jakości od konkretnych testerów. W zamian wnoszą metodyczną powtarzalność do procesu planowania testów, zapewniając tym samym przewidywalną powtarzalność pokrycia testami. Organizacje bazujące na tej metodzie dużo łatwiej mogą wprowadzać techniki optymalizacji procesu testowania, redukując przypadki testowe do liczby efektywnie pokrywającej wymagania, dzięki czemu uzyskują przyspieszenie cyklu testowego oraz utrzymują ten proces na poziomie ułatwiającym zarządzanie nim.

W większości organizacji niemal wszyscy analitycy biznesowi używają "naturalnego" języka do opisu wymagań. Wiemy, iż używanie takiego języka może utrudnić osiągnięcie pełnego pokrycia aplikacji testami, ponieważ testerzy bazując na potocznie sformułowanych wymaganiach muszą odwoływać się do własnej intuicji, aby określić stopień pokrycia testami obszaru tak zdefiniowanych wymagań. Innymi słowy, pracując z potocznie sformułowanymi wymaganiami organizacje nie mają możliwości dokładnej kontroli pokrycia ich przypadkami testowymi, co skutkuje w późniejszych błędach w wypuszczonej wersji aplikacji.

W celu usystematyzowania pokrycia wymagań odpowiadającymi im testami firmy i działu produkujące oprogramowanie powinny skupić się na sformalizowaniu reprezentacji wymagań. Kiedy uda się to osiągnąć, będzie możliwe stworzenie szkieletu przypadków testowych zapewniających optymalne pokrycie wymagań, docelowo zaś powstaną z nich już właściwe testy, które będziemy mogli uruchamiać i przeprowadzać.

Aby wprowadzić usystematyzowanie i odpowiednią strukturę do wymagań sformułowanych przy pomocy naturalnego języka opisu można skorzystać z wielu dostępnych technik. Ich celem jest odkrycie związków przyczynowo-skutkowych zawartych w wymaganiach, a przez to przedstawienie ich w postaci zestawu warunków (przyczyn) oraz wynikających z nich akcji (skutków). Tabele przyczynowo skutkowe

są jedną z takich technik. Innym sposobem na osiągnięcie podobnego rezultatu jest przedstawienie wymagań w postaci wykresów przepływu (flowchart), które w naturalny sposób pokazują związki przyczynowo-skutkowe, jak również gałęzie warunkowe odpowiednich akcji.

To swoiste "tłumaczenie" wymagań pozwala na dużo łatwiejsze zbudowanie w oparciu o nie konkretnych przypadków testowych. Logiczny zestaw tych ostatnich może być stworzony automatycznie bądź "ręcznie" tak, by stanowił odbicie zdefiniowanych i przetłumaczonych w powyższy sposób wymagań. Należy pamiętać, że tak stworzony zestaw przypadków testowych może zawierać nachodzące na siebie testy. W celu zoptymalizowania ich liczby przy zachowaniu pełnego pokrycia wymagań możemy skorzystać z tablic decyzyjnych (w przypadku zastosowania tabel przyczynowo-skutkowych na etapie strukturyzowania wymagań). W przypadku, jeśli w tym celu opracowane zostały uprzednio wykresy zamiast tabel, optymalizacja sprowadza się do znalezienia unikatowych ścieżek przepływu na wykresach. Do tego celu można wykorzystać wiele sprawdzonych algorytmów.

Nawet po zaprojektowaniu przypadków testowych, wciąż będziemy mieli do czynienia z prawdopodobieństwem występowania w nich błędów. Aby wykryć potencjalne błędy, organizacje stosujące podejście RBT angażują zarówno analityków jak i użytkowników końcowych do weryfikacji przypadków testowych zanim powstaną na ich podstawie właściwe testy. To podejście pozwala wszystkim zainteresowanym ponownie zapoznać się z wymaganiami i przypadkami testowymi, co daje możliwość ich dokładniejszej weryfikacji, jak również wyłapania błędów powstałych w procesie przenoszenia wymagań z języka opisowego na systematyczny.

Dodatkowo, firmy charakteryzujące się podejściem do cyklu tworzenia aplikacji jako całości, starają się interaktywnie włączyć zadania Działu Jakości w prace analityków i programistów, co staje się możliwe właśnie dzięki przejściu na usystematyzowane i zoptymalizowane przypadki testowe wypracowane w poprzednich fazach.

W opisanym przedsiębiorstwie projekt aplikacji bywa przeglądany pod kątem przypadków testowych, ponieważ stanowią one niejako inną formę zdefiniowanych wymagań. W ten sposób zespoły projektowe nabierają pewności, że projekt spełnia założenia i oczekiwania w postaci wymagań. Jeżeli model ich nie spełnia może to być oznaką, że albo nie odpowiada tymże, i wymaga dalszej pracy, albo że jest problem w samych wymaganiach, co pociąga za sobą dodatkową pracę analityków.

Aby uniknąć kosztownych przeróbek, przypadki testowe powinny być przeglądane również przez programistów tworzących kod. To zapewni im dobre zrozumienie, które elementy oraz w jakim zakresie będą testowane, a jednocześnie da strukturalny zestaw wymagań.

Ostatecznie pojedyncze moduły kodu powinny zostać zweryfikowane pod kątem uporządkowanych wymagań tak, aby upewnić się, że powstały kod w pełni odpowiada założeniom. Praktyka pokazuje, że dużo łatwiej jest nam dopasować postać algorytmiczną kodu do uporządkowanych wymagań niż do ich niestrukturalnej postaci.

- **Testujmy z wykorzystaniem metryk**

Istnieje opinia, że czego nie da się zmierzyć, nie istnieje. Używając metody RBT firmy mogą nie tylko zarządzać procesem zarządzania jakością, ale i go usprawniać. W procesie RBT może być stosowanych wiele różnych miar kwantyfikujących status projektu oraz aktywność jego członków. Stanowi to dużą pomoc dla menedżerów działu oraz menedżerów projektu, pozwalając na dokładny wgląd w zakres całego portfolio IT. Przykłady informacji, które powinny być mierzone:

- procent wymagań przejrzanych przez projektantów i programistów
- procent wymagań, które zawierają dwuznaczności
- procent wymagań o strukturalnej formie

- o procent formalnych wymagań pokrytych przypadkami testowymi
- o logiczne i faktyczne pokrycie kodu

Rola śledzenia zmian w RBT

Śledzenie zmian odgrywa także istotną rolę w organizacjach wykorzystujących RBT - od podtrzymywania stałego przepływu informacji o zmianach w stosunku do wymagań, przypadków testowych i testów jest krytyczne. Te informacje są niezbędne dla prawidłowego monitorowania postępu i stanu projektu, jak również do właściwego zarządzania zmianami wymagań. Bez tej wiedzy trudne jest dokładne określenie, które przypadki testowe i testy powinny zostać zmodyfikowane w przypadku zmiany w wymaganiach.

Nawet jeśli rozumiemy znaczenie śledzenia zmian, w wielu firmach tworzących oprogramowanie ta wiedza wciąż pozostaje bardzo trudna do uchwycenia we właściwy sposób. Podstawowym powodem tego jest to, że większość narzędzi dostępnych obecnie na rynku wymaga od niemal wszystkich członków zespołów projektowych ręcznego wprowadzania i zarządzania śledzeniem zmian. Z oczywistych powodów takie podejście jest raczej niemożliwe do zaakceptowania. Aby podjąć temu wyzwaniu, należy poważnie zastanowić się nad rozwiązaniami umożliwiającymi połączenia pomiędzy produktami poszczególnych faz wytwarzania oprogramowania.

Pierwszy krok do Zarządzania Jakością Cyklu Życia Aplikacji

Stosowanie solidnych praktyk RBT przez organizacje tworzące oprogramowanie bardzo szybko dostarcza im odpowiednie narzędzia i procesy umożliwiające maksymalizację wartości biznesowej działalności.

Zespoły wdrażające podejście RBT poprzez fakt, że robią ten pierwszy krok w celu zwiększenia jakości tworzonych produktów, stają się tym samym inicjatorami szeregu zmian prowadzących do implementacji Zarządzania Jakością w Cyklu Życia Aplikacji (Lifecycle Quality Management - LQM). Poprzez położenie nacisku na zapewnienie jakości na wszystkich etapach wytwarzania oprogramowania, a nie koncentrowanie się na jakości samego kodu, działania z obszaru LQM zapewniają firmom podniesienie norm jakości i usług oraz systematyczną redukcję kosztów związanych z wielokrotnym powtarzaniem tej samej pracy lub prób opanowania złożonych projektów.

Ponadto, działania LQM znacząco przyspieszają ścieżkę do Optymalizacji Procesu Dostarczania Oprogramowania (Software Delivery Optimization - SDO), która z kolei pomaga przekształcić tworzenie i rozwój oprogramowania w zarządzalny proces biznesowy, zapewniając tym samym zwiększoną kontrolę, przewidywalność oraz wydajność w całym procesie dostarczania oprogramowania.

O autorze:

Moty Aharonovitz pracuje jako senior director of Product Strategy w firmie Borland. Ma ponad 15 lat doświadczenia w pracy nad rozwojem oprogramowania, dzięki czemu aktywnie wspiera i rozwija wizję Optymalizacji Dostarczania Oprogramowania (SDO) w przedsiębiorstwach i firmach informatycznych oraz czynnie wspiera rozwiązania z zakresu Zarządzania Jakością w Cyklu Życia aplikacji. Kontakt: moty.aharonovitz@borland.com.