Podniesienie poziomu wydajności kontroli jakości (QA) oprogramowania

developerWorks

Autorem artykułu jest

Leonard DiMaggio

inżynier i kierownik ds. jakości oprogramowania, IBM

Cytat z magazynu The Rational Edge:
„Po osiągnięciu przez zespół programistów wysokiego poziomu sprawności znajdywanie sposobów na wprowadzanie dalszych udoskonaleń może stanowić prawdziwe wyzwanie. Z perspektywy kierownika ds. kontroli jakości artykuł ten oferuje praktyczną informację — zespoły pracujące w prawie każdym obszarze cyklu prac programistycznych mogą jeszcze bardziej zwiększyć wydajność swojej pracy".

Niedawno rozmawiałem na temat testowania oprogramowania ze swoim przyjacielem, który jest grubą rybą w niewielkiej firmie produkującej oprogramowanie. Wypowiadał się entuzjastycznie o pracy swojego zespołu ds. kontroli jakości, ale pomimo wysiłków tego zespołu, w tym opracowania bogatego zestawu zautomatyzowanych testów, przyjaciel wciąż martwił się tym, że pełne przetestowanie kolejnej wersji produktu firmy zajmuje aż tyle czasu. „Musimy zwiększyć wydajność pracy naszego zespołu ds. kontroli jakości" – powiedział. „Pracują doskonale, ale muszę coś zrobić, aby weszli na wyższy poziom".

Jego lamenty na temat zwiększenia wydajności zespołu ds. kontroli jakości sprawiły, że pomyślałem o… golfie. (Zostańcie ze mną jeszcze przez chwilę). Golf jest grą, której bardzo trudno jest się nauczyć. Ale prawdziwy problem z golfem polega na tym, że im jest się lepszym, tym trudniej udoskonalić sposób gry. Trudno jest przejść z wyników rzędu 100 punktów na poziom 90. Jeszcze trudniej jest przejść do wyników rzędu 80 punktów, a już całkiem trudno osiągać wyniki na poziomie 70 punktów. Z chwilą kiedy już osiągnie się taki poziom, utrzymanie się na nim kosztuje wiele pracy. Wiem to z własnego doświadczenia. Przed ślubem, zaciągnięciem kredytu hipotecznego i pojawieniem się dzieci grałem z handicapem 6 lub 7 na długim, trudnym polu golfowym. Teraz grywam z handicapem 9 lub 10 na znacznie krótszym i łatwiejszym polu. Dlaczego? Jedna z przyczyn jest taka, że teraz gram w golfa tyle razy w ciągu roku, ile kiedyś grałem w ciągu tygodnia. Mam nadzieję wkrótce rozwiązać ten problem – właśnie kupiłem dzieciakom (w wieku trzech i pięciu lat) ich własne kije golfowe.

Podobnie jest z testowaniem oprogramowania. Zespół ds. kontroli jakości może stać się znacznie bardziej efektywny i wydajny, jeśli zamieni podejście ad hoc do testowania na formalne, powtarzalne procesy obejmujące planowanie, analizę przyczyn usterek i testy automatyczne. Działania te nie są proste, ale w przypadku dzisiejszych zespołów ds. kontroli jakości oprogramowania stanowią już standard. Ale co można jeszcze zrobić po wprowadzeniu tych wszystkich procesów, aby znów przejść „na wyższy poziom"?

Nie odpowiedziałem wtedy koledze na to pytanie. Od tego czasu jednak często o tym myślę i sądzę, że odpowiedź tkwi w działaniach, które opisuję w tym artykule.

Zrozumieć cel: co oznacza słowo „wydajny”?

Ilekroć zaczynam pracę nad zadaniem dotyczącym zwiększenia wydajności zespołu ds. kontroli jakości, mój pierwszy krok polega na upewnieniu się, że rozumiem ostateczny cel tego zadania. Powinniśmy jasno zdefiniować, co rozumiemy pod słowem „wydajność”. Standardowa definicja tego słowa (z serwisu dictionary.com) brzmi następująco:

ef•fi•cient, *adj. (wydajny/skuteczny, przym.)*

1. Działający bezpośrednio w celu osiągnięcia określonego wyniku/efektu/skutku, czyli powodu wydajności. Patrz synonimy pod hasłem effective (efektywny/skuteczny).
2. a. Działający lub produkujący wydajnie przy minimalizacji odpadów, kosztów i niepotrzebnych działań.
   b. Cechujący się wysokim wskaźnikiem efektów do nakładów.

Cofnijmy się o krok wstecz i zdefiniujmy, co oznacza wydajność w przypadku zespołu ds. kontroli jakości oprogramowania. W tym celu musimy się cofnąć jeszcze o jeden krok wstecz i przypomnieć sobie o głównym celu takiego zespołu i ogólnie testowania oprogramowania.

Zespół ds. kontroli jakości oprogramowania ma wiele obowiązków. Zespół ten musi przeglądać specyfikacje funkcji i przedstawiać swoje opinie autorom. Musi budować narzędzia do testowania i opracowywać plany testów. Zespół musi przeprowadzać testy i przedstawiać raporty na temat wyników testowania. I jeszcze wiele, wiele więcej. Ale wszystkie te działania są tylko środkiem prowadzącym do celu. A tym celem jest znajdywanie błędów w oprogramowaniu; zwłaszcza takich błędów, których wcześniej nie wykryto i które mogą mieć negatywne skutki dla klientów. Dlatego właśnie testuje się oprogramowanie: aby znajdywać błędy. Wszystkie badania, planowanie, przeglądy, prace nad automatyzacją testów i obsługa serwisowa są wykonywane w jednym celu: z myślą o wykrywaniu błędów w oprogramowaniu. Gdy mówimy zatem o wydajności zespołu ds. kontroli jakości oprogramowania, w rzeczywistości chodzi nam o oto, aby ten zespół był w stanie szybciej znajdować poważniejsze błędy.

Mój przyjaciel (szef firmy programistycznej, o którym wspomniałem we wstępie), wypowiadając się na temat wydajności swojego zespołu ds. kontroli jakości, chciał, aby był on w stanie zweryfikować kolejną wersję produktu powiedzmy w cztery tygodnie zamiast ośmiu. W zasadzie cel ten oznacza, że zespół ten musi szybciej wykrywać poważne błędy. Aby skoncentrować się na szybszym wyszukiwaniu takich poważnych błędów, członkowie zespołu muszą unikać tracenia czasu na inne zadania napotykane podczas szukania.

Tak więc pierwszy krok na drodze do zwiększenia wydajności zespołu polega na uważnym przyjrzeniu się, gdzie jego członkowie teraz działają nieefektywnie. Innymi słowy, czy robią coś, co nie pomaga im w znajdowaniu poważnych błędów. A to prowadzi nas do następnego działania.

„Nie oszukujmy się”: rozpoznanie nieefektywnych elementów w pracy zespołu

Istnieje pewne znakomite powiedzenie przypisywane Johnowi D. Rockefellerowi: „Nie oszukujmy się”. Może dlatego właśnie Rockefeller osiągnął tyle sukcesów: był zawsze uczciwy względem siebie.

Jeśli chce się zidentyfikować obecne obszary zmniejszonej wydajności zespołu ds. kontroli jakości, trzeba przyjrzeć się krytycznie procedurom stosowanym przez zespół w celu osiągnięcia efektywności. Najprawdopodobniej procedury te są efektem rzeczywistych doświadczeń oraz adaptacji innych dobrze znanych procedur, a także mnóstwa ciężkiej pracy. Trzeba być przygotowanym na zmianę, a być może nawet na całkowite odrzucenie niektórych spośród tych procedur.

Co może być trudne do realizacji. Przecież to właśnie te procedury przyczyniły się głównie do osiągnięcia przez zespół dotychczasowej efektywności. Pracownicy będą do nich w poważnym stopniu emocjonalnie przywiązani. Ale jeśli chce się wprowadzić zespół na „wyższy poziom” wydajności, trzeba być gotowym do zmiany każdego aspektu funkcjonowania tego zespołu.

Trzeba sobie odpowiedzieć na następujące pytanie: Czego się szuka, analizując procedury zespołu? Należy pamiętać, że chce się zidentyfikować wykonywane zadania, które nie prowadzą do znalezienia nowych, poważnych błędów,

a także zadania, które są po prostu zbyt czaso- i pracochłonne z punktu widzenia błędów, do których wykrycia doprowadziły. W zasadzie należy skonfrontować swoje plany, nadzieje i wizje z rzeczywistością. Zespół ds. kontroli jakości oprogramowania zazwyczaj spędza czas na projektowaniu testów, pisaniu kodu, refaktoryzacji, debugowaniu, wykonywaniu i analizowaniu. Na początku projektu zespół prawdopodobnie opracowuje harmonogram prac i plan wykorzystania zasobów kadrowych, potrzebnych do realizacji tych zadań. Następnie, gdy zmieniają się ogólne cele projektu, harmonogram taki się prawdopodobnie wyrzuca i nigdy do niego nie wraca. Należy postępować wręcz przeciwnie – trzeba określić różnice między planowaną a faktyczną czaso- i pracochłonnością wykonywanych zadań. Następnie należy porównać wyniki tej analizy z rezultatami (tzn. z liczbą i specyfiką błędów), które te wysiłki przyniosły.

I odpowiedzieć sobie na pytanie, gdzie wydajny zespół ds. kontroli jakości może tracić czas w trakcie realizacji projektu? (I jak można temu zaradzić?). Jeden z obszarów, który mogę wskazać, dotyczy „dźwigania bagażu".

Dźwiganie bagażu

Jeśli zespół ds. kontroli jakości pracował przez jakiś czas razem i osiągnął znaczną efektywność testowania, musiał opracować sporą liczbę testów zautomatyzowanych oraz środowisko do ich uruchamiania. Ale mogą być wśród nich testy, które przestały już być użyteczne. W miarę powstawania coraz większej ilości testów zespół może spędzać coraz więcej czasu na obsłudze tych testów i ich środowiska.

Ostatnio spotkałem kogoś, kto pracuje w odnoszącej duże sukcesy firmie produkującej oprogramowanie i którego zespół ds. kontroli jakości miał właśnie taki problem. Przez ponad dziesięć lat pracy zespół ten opracował dosłownie tysiące zautomatyzowanych testów. Testów było tak dużo, że ich wykonanie zajmowało w rzeczywistości prawie cały dzień. Było tam tak dużo testów napisanych w ciągu tych lat przez tak wiele osób, że nikt nie miał bladego pojęcia, co wszystkie te testy tak naprawdę sprawdzały.

I to jest właśnie ten bagaż związany z automatyzacją testów. Każdy taki test został kiedyś napisany z jak najbardziej uzasadnionego powodu. Ale w miarę upływu czasu racjonalne uzasadnienie stosowania tych testów gdzieś zaginęło. Testy zostały również zorganizowane w postaci wielu różnych, ale wciąż bardzo dużych pakietów testów. W rezultacie wykonanie nawet jednego pakietu testów zajmowało kilka godzin. A zespół zawsze pisał nowe testy dotyczące nowych lub zmodyfikowanych funkcji produktu. Aby zwiększyć wydajność, trzeba było wyeliminować testy, które stały się już bezużyteczne.

Jak można zwiększyć wydajność testowania? Przez wykonywanie mniejszej liczby testów. Należy pomijać testy, których skuteczność nie uzasadnia już ich wykonywania. Trzeba się skoncentrować na pisaniu i wykonywaniu testów ważniejszych. Innymi słowy, należy pisać i wykonywać testy, które umożliwią wyszukiwanie nowych, poważnych błędów.

W porządku. Wygląda na to, że to nic trudnego. Ale w jaki sposób można określić, które testy są na tyle ważne, aby je wykonywać i zachować? Wszystko zależy od tego, które części danego produktu są najbardziej narażone na wystąpienie błędów. To znaczy, które są najbardziej narażone na to dzisiaj.

Zrozumienie ryzyka zagrażającego produktowi oraz potrzeba dostosowywania się każdorazowo do zmian w ryzyku

Czasami takie ryzyko stanowi sam kod, gdyż jest nowy lub podlega innego rodzaju zmianom. Glenford Myers opowiada w książce pt. „The Art of Software Testing" (Sztuka testowania oprogramowania – jest to wciąż najlepsza książka o testowaniu oprogramowania, jaką kiedykolwiek czytałem[1]) o tym, jak nadal znajduje się nowe błędy w tych samych obszarach programu, w których już wcześniej naprawiono inne. Może się tak zdarzyć z wielu powodów. Na przykład dlatego, że w kodzie roi się od błędów, projekt jest niewłaściwy lub ciągle się go zmienia. Często problem występuje też wtedy, gdy nic złego nie dzieje się z pierwotnym kodem, ale dodaje się do niego nowe funkcje.

Poziom ryzyka może się zmieniać w czasie, gdy nowe funkcje się starzeją. Jeśli dana część kodu nie zmienia się w czasie, na skutek wykrycia błędów lub wprowadzania rozszerzeń, poziom ryzyka może maleć w porównaniu z innymi fragmentami kodu. Tak więc należy zawsze mieć świadomość aktualnej sytuacji i korygować plany, aby obsługiwać te fragmenty kodu, które są aktualnie obciążone największym ryzykiem. Zasoby firmy są zawsze ograniczone, natomiast liczba możliwych testów jest zawsze nieskończona.

Czasami poziom ryzyka nie ma nic wspólnego ze stanem, w jakim znajduje się kod, ale wynika z tego, że dana funkcja (lub konfiguracja) ma newralgiczne znaczenie dla produktu ogólnie lub dla określonych klientów. Pracowałem kiedyś dla firmy produkującej rozwiązania sieciowe, które udostępniła do przetestowania jako wersję beta produktu, chociaż brakowało obsługi jednego rodzaju modemu. I co się stało? Jak łatwo się domyślić, pierwszy klient korzystający z wersji beta chciał użyć właśnie tego nieobsługiwanego modemu.

Oprócz zdania sobie sprawy, gdzie leży największe ryzyko, trzeba też stale dostosowywać swoje plany, tak aby nadążały za zmieniającym się stanem oprogramowania. Oprogramowanie jako medium jest tak plastyczne[2], że łatwo jest w nim szybko wprowadzać radykalne zmiany, ale jednocześnie równie szybko wprowadza się nowe zagrożenia. Próbujemy wszystko przewidzieć, ale w trakcie testowania zawsze uczymy się czegoś nowego i musimy zmieniać plan, aby uwzględnić w nim nowe lub zmienione testy (albo usunąć inne).

Podsumowanie

* Trudno jest spowodować, aby niewydajny zespół stał się wydajny, ale jeszcze trudniej jest zwiększyć wydajność już wydajnego zespołu.
* Jeśli pragniemy określić, czy i na ile nasz zespół jest wydajny, nie należy się oszukiwać. Trzeba dokonać krytycznego przeglądu procesów i dotrzeć do sedna tego, na co i w jaki sposób poświęcany jest czas zespołu. Trzeba też pogodzić się z tym, że pewnego dnia niektóre testy trzeba będzie wyrzucić.
* Co oznacza określenie, że zespół ds. kontroli jakości jest wydajny? Oznacza to, że zespół ten szybko znajduje poważne błędy.
* Aby zwiększyć wydajność i wykrywać poważne błędy, należy skoncentrować się na obszarach narażonych na ryzyko.
* Należy pamiętać, że obszary narażone na ryzyko ciągle się zmieniają i trzeba się do tego na bieżąco dostosowywać.

Przypisy

1 Glenford J. Myers, Corey Sandler, Tom Badgett i Todd M. Thomas, The Art of Software Testing, wydanie 2-gie. John Wiley and Sons, 2004.

2 Frederick P. Brooks, Jr., „No Silver Bullet: Essence and Accidents of Software Engineering". http://portal.acm.org/citation.cfm?id=26440.26441&dl=GUIDE&dl=GUIDE&CFID=66886534&CFTOKEN=1562060

Informacje o autorze

Leonard DiMaggio jest inżynierem i kierownikiem ds. jakości oprogramowania w IBM Rational.
Przed rozpoczęciem pracy w Rational kierował zespołami testującymi oprogramowanie w firmach BBN, GTE i Genuity. DiMaggio opublikował artykuły na temat testowania oprogramowania w magazynach STQE, Software Development, Dr. Dobbs Journal oraz QAI Journal.

Źródło:
http://www.ibm.com/developerworks/rational/library/feb06/dimaggio/index.html?S_TACT=105AGX15&S_CMP=LP