QUALE

Quale 3/2014

Funkcja oceny testowania

Tomasz Osojca

Podstawy podstaw, czyli raport defektu

Marta Firlej

Pop-psychologia w IT a zarządzanie

Bogdan Bereza

Automation testing is not testing!

Spis treści

KRZYWYM OKIEM

11. Testowanie w rytmie Pop
Bartłomiej Prędki

13. Testowanie z Rockowym pazurem
Krzysztof Chytła

CO JEST CZYM CZEGO

15. Jeśli chcesz coś naprawdę zepsuć, użyj do tego komputera
Tomasz Osojca

TESTOWANIE OPROGRAMOWANIA

5. Automation testing is not testing
Tomasz Olszewski

17. Funkcja oceny testowania
Tomasz Osojca

33. Podstawy podstaw, czyli raport defektu
Marta Firlej

REDAKCJA

Redaktor naczelny
Karolina Zmitrowicz
karolina.zmitrowicz@quale.pl

Zastępca redaktora naczelnego:
Krzysztof Chytła
krzysztof.chytla@quale.pl

Redaktorzy:
Bartłomiej Prędki
bartlomiej.predki@quale.pl
Michał Figarski
michal.figarski@quale.pl

Współpraca:
Tomasz Olszewski
tomasz.olszewski@quale.pl
Piotr Poznański
piotr.poznanski@quale.pl

Website:
www.qualemagazine.com (ENG)
www.quale.pl (PL)

Facebook:
http://www.facebook.com/qualemagazine

Spis treści

OPROGRAMOWANIE DOSKONAŁE

25. Moje testowanie
Radosław Smilgin

WARZELNIA JAKOŚCI

28. „BDD u mnie nie działa”
Daniel Dec

AGILE - ZRÓB TO SAM!

40. Praktyki i wspierające je narzędzia w projektach zwinnych
Rafał Stańczak

BIZNES

45. Wirtualizacja. Przewaga konkurencyjna to garść wizytówek do zaufanych partnerów

48. Społeczna odpowiedzialność sprzyja rozwojowi biznesu

ZARZĄDZANIE

51. Pop-psychologia w IT Zarządzanie
Bogdan Bereza

58. Jak złapać testera
Justyna Rybarczyk i Agnieszka Skokowska

ORGANIZACJE

58. Fundacja Governica

WYDARZENIA

60. TestingCup po raz trzeci

SEETEST
2014 SOUTH EAST EUROPEAN
SOFTWARE TESTING CONFERENCE BUCHAREST

25-26 September
BUCHAREST, ROMANIA

Speakers:

- Rex Black, USA
- Erik van Veenendaal, Holland
- Graham Bath, Germany
- Geoff Thompson, UK
- Yaron Tsubery, Israel
- Vipul Kocher, India

20% EARLY BIRD DISCOUNT!
1 - 31 AUGUST
REGISTER NOW!

www.seetest.org info@seetest.org

SEETB South East European Testing Board

ANIS employers' association of the software and services industry

QUALITY HOUSE

Tomasz Olszewski

Automation testing is not testing

Artykuł jest odpowiedzią na publikację Wiktora Żołnowskiego z numeru 2 Quale "(Manual) Test is dead".

Numer 2 można pobrać ze strony magazynu Quale.

Sobotni wieczór

Otwieram ostatni numer Quale, spoglądam na okładkę i zamieram. (Manual) Test is dead! Staram się, by moje wypowiedzi nie były emocjonalne, ale to mnie osobiście zabolało. Nie zwlekając ani chwili rozpoczynam czytanie. Seria nieartykułowanych dźwięków z mojej strony zwraca uwagę mojej żony, która zaczyna dopytywać o moje zachowanie. Kolega Wiktor, niczym Alfred Hitchcock w swoich filmach, rozpoczął swoją wypowiedź od trzęsienia ziemi. Na szczęście trochę potem stonował wypowiedź, ale tak czy inaczej nie mogę zostawić tego bez odpowiedzi.

Prolog

Otwieram portal z ogłoszeniami o pracę i równie radośnie wpisuje w wyszukiwarkę "Tester Oprogramowania". Podobnie jak Wiktor przeglądam pierwsze 20 wyników i co widzę? W co drugim ogłoszeniu widzę ofertę zatrudnienia dedykowanego fachowca w dziedzinie testowania oprogramowania z konkretnymi wymaganiami odnośnie znajomości technik testowania i doświadczenia, podczas gdy druga połowa wyników wyszukiwania poszukuje testera z umiejętnościami automatyzacji. Oni szukają - nie bójmy się tego powiedzieć - testera developera. Osoby, która jest konstruktywna i destruktywna zarazem. Na schizofrenię cierpi 0,4-0,6% dorosłej populacji i myślę, że ta liczba odzwierciedla odsetek developerów / testerów, którzy mogą podjąć się tego, jakże trudnego, zadania. Wydaje mi się również, że Ci ludzie już posiadają satysfakcjonującą ich pracę jako software developer in test w największych korporacjach z branży IT.

Czy automatyczne testowanie to w ogóle jest testowanie?

Paul Gerrard - tegoroczny szef komisji programowej jednej z największych (o ile nie największej) konferencji testerskiej EuroSTAR, niewątpliwie osoba o ogromnej wiedzy z zakresu szeroko pojętego zapewniania jakości, autor książki "Learn Python" - na jednej ze swoich prezentacji stwierdził "All testing is exploratory".

Miałem przyjemność porozmawiać z nim i poruszyć temat tego, jak jego stwierdzenie ma się do automatyzacji, jak automatyzacja ma się do eksploracji? Skoro jesteśmy, nie boję się tego powiedzieć, przywiązani do określonej ścieżki testów jak łańcuchem, to czy to jest eksploracja? Czy to jest testowanie czy może jednak sprawdzenie? Ktoś może powiedzieć: "Możemy parametryzować automatyzację."- czy to nie jest jednak tylko wydłużanie łańcucha? Odpowiedź, którą otrzymałem na moje pytanie wstrząsnęła mną wtedy. „Co jeśli założymy, że testowanie automatyczne to nie testowanie?" - odpowiedział mi pytaniem na pytanie. To mnie skłoniło do kilku rozważań.

Michael Bolton, kolejna osobowość w świecie testowania, powiedział: „Jeśli nie używasz wiedzy zdobytej podczas projektowania i wykonywania testów do udoskonalania i poprawiania jakości swojej pracy, to to co robisz nie jest testowaniem eks-

ploracyjnym." Czy automatyzacja wyciąga wnioski z kolejnych iteracji? Czy "poprawia się" sama z siebie? Odpowiedź na te pytania brzmi Nie, jeszcze nie teraz.

Skoro testowanie jest równoznaczne z testowaniem eksploracyjnym, skoro automatyzacja nie jest testowaniem eksploracyjnym, to jest to równoznaczne z tym, że automatyzacja nie jest testowaniem. Nie ma więc czegoś takiego jak testowanie automatyczne.

Informacje zwrotne

W tym momencie zatrzymajmy się na moment. Chciałbym odnieść się do jeszcze jednego argumentu odnośnie zbyt późnego przekazywania informacji zwrotnej do programistów. Jeśli jakakolwiek informacja, czy to z testów eksploracyjnych, skryptowych czy z automatyzacji, dociera zbyt późno do developerów, to jest to ze wszech miar naganne. Samo wykonanie testów automatycznych nie gwarantuje przekazania informacji do developerów, tak samo jak wykonanie testów manualnych. Głównym zadaniem testerów nie jest, wbrew obiegowej opinii, znajdowanie błędów. Błędy w aplikacji może znaleźć każdy. Każdego dnia miliony ludzi pracujących na komputerach spotykają się z defektami w oprogramowaniu. Czy wszystkich ich nazywamy testerami? Przecież znaleźli błąd. Duża część z nich pewnie wie jak je zreprodukować. Skoro znalezienie błędu nie definiuje testera, to w takim razie co? Co jest największą wartością dodaną testera?

Najważniejszym zadaniem testera jest dostarczenie informacji wszelkim zainteresowanym stronom, od developerów poczynając, przez analityków biznesowych, po menadżera produktu. To tester wie jako pierwszy, jakiej jakości jest testowany produkt i ma obowiązek podzielić się tymi informacjami z innymi. Samo wykonanie testów czy automatyzacji nie wnosi nam żadnej wartości, tak długo, jak nikt nie zostanie poinformowany o rezultatach.

W tym momencie ktoś może zapytać: A co jeśli wyniki są automatycznie dostarczane do developerów?. To pytanie prowadzi do sedna sprawy.

Czym zatem jest automatyzacja?

Automatyzacja jest częścią dostarczanego oprogramowania. Jest pierwszym stopniem zapewniania jakości wytwarzanego produktu. Sam Wiktor przyznaje, że nie da się automatyzować wszystkiego. Automatyzacja jest bardzo podatna na efekt pestycydów, ale rewelacyjnie sprawdza się w regresji czy też testowaniu "happy path scenario". Automatyzacja jest jak podążanie po szlaku górskim z poręczówką. Pozwala nam wpiąć się w nią, by zapewnić bezpieczeństwo i ułatwić pokonanie najtrudniejszych (najnudniejszych) fragmentów trasy.

Testowanie to wyszukiwanie, to bazowanie na wewnętrznej potrzebie do odkrywania, która drzemie w człowieku. Dlaczego himalaiści wdrapują się na ośmiotysięczniki w środku zimy? Bo nikogo tam jeszcze nie było w tym okresie. To samo tyczy się testowania - eksploracji. Bazujemy na naturalnych zachowaniach w stylu "co się stanie jeśli...?" Automatyzacja ma za zadanie sprawdzenie czy wszystko jest na miejscu, a my jako testerzy szukamy tego, co znajduje się za następnym zakrętem.

Testowanie eksploracyjne

Na tym mógłbym skończyć, ale chciałbym poruszyć jeszcze sprawę testów eksploracyjnych. Kwestia testów eksploracyjnych w artykule Wiktora zaczęła się od stwierdzenia: "Oprócz testów automatycznych konieczne jest testowanie manualne, ale nie tak, jak to było robione dawniej - testowanie wszystkiego - tylko raczej testowanie eksploracyjne, które ma na celu odkrycie kolejnych obszarów, które warto by było zautomatyzować, lub wykrycie niedociągnięć których automaty nie są w stanie wykryć."

Podpisałbym się pod tym obiema rękoma, jedyną kwestią, którą chciałbym wyjaśnić to pozycjonowanie testów eksploracyjnych. Testy eksploracyjne jako technika testowania nie są poziomem testów, ale jak inne techniki mogą być stosowane na wszystkich poziomach i niczym chmura owiewają całą piramidę testów.

Epilog

Testowanie to złożony proces. Składa się z testów eksploracyjnych, testów skryptowych i automatyzacji. Połączenie wszystkich powyższych daje nam szansę na poprawne przetestowanie produktu i potwierdzenie jego jakości.

Testy manualne są i jeszcze bardzo długo będą częścią procesu testowania aplikacji. Jest duża presja na automatyzację w trakcie tworzenia aplikacji, pamiętajmy jednak, że nie musimy ślepo podążać za trendami. Jeśli w naszym projekcie mamy osoby, które zajmują się automatyzacją, to świetnie, lecz jeśli nie, to nie załamujmy rąk. Warto porozmawiać z developerami i kierownictwem projektu. Developerzy w naturalny sposób mogą wpleść automatyzację w proces tworzenia oprogramowania. Dzięki dobrej wymianie informacji z nimi możemy sobie zapewnić benefity w postaci świetnego stanowiska pracy, zautomatyzowanego procesu testowego i dodatkowego czasu na najciekawszą część pracy testera - eksplorację. Siadając do aplikacji doświadczony tester jest w stanie wykonać dziesiątki testów eksploracyjnych, zanim w ogóle uda sie uruchomić framework automatyzacji.

W dobie szybko wytwarzanego oprogramowania, gdzie projekty potrafią być wykonywane w 1-2 iteracjach, często może nie być czasu na przygotowanie automatyzacji funkcjonalności (co nie zwalnia nas z przygotowania testów jednostkowych), za to na pewno będziemy w stanie przygotować i udoskonalić kilka sesji testów eksploracyjnych. Jeśli tworzenie skryptów automatycznych jest dla nas odskocznią w pracy pozwalającą na zwiększenie kreatywności w poszukiwaniu przepisu na kolejne testy, to jest to godne pochwały. Jeśli jednak tworzymy automatyzację po to, by "przeszła" to zatrzymajmy się na chwilę i zastanówmy, czy droga, którą obraliśmy jest właściwa. Pamiętajmy - automatyzacja jest programem i jako taki potrzebuje utrzymania oraz sprawdzenia czy sama z siebie nie zawiera błędów.

AUTOR

Tomasz Olszewski

Tomasz Olszewski – test inżynier w firmie Luxoft , absolwent Politechniki Wrocławskiej, miłośnik testowania, metodologii zwinnych i zdobywania nowych doświadczeń.

W wolnym czasie biegacz długodystansowy, wielbiciel gór i fan "Gwiezdnych Wojen".

IV KONFERENCJA **COMPUTERWORLD**

23-24 WRZEŚNIA 2014, WARSZAWA
DOUBLETREE BY HILTON
HOTEL & CONFERENCE CENTER

Agile w biznesie

✓ **INSPIRUJĄCE CASE STUDIES** – m.in. Centralny Ośrodek Informatyki, ING Bank Śląski, Onet, Orange Polska, PayU – poznaj praktyczne doświadczenia z dojrzałych transformacji, ale także nowe wdrożenia z wielu branż

✓ **BIEŻĄCA TEMATYKA, KLUCZOWE PROBLEMY** – czy da się zmierzyć agile, zarządzanie zmianą w metodykach agile, zarządzanie zespołami rozproszonymi, kiedy zwinność na poziomie projektu jest widoczna na poziomie biznesowym, realizacja kontraktów w duchu agile i wiele innych

✓ **SESJE WARSZTATOWE** – weź udział w jednym z równoległych, specjalistycznych, interaktywnych treningów, by zgłębić swoje umiejętności:
- Warsztat I – Fakty i mity o holokracji – krótki kurs wdrożenia w organizacji
- Warsztat II – Przewodnik po metodykach zwinnych – czy agile jest dla mojej organizacji
- Warsztat III – Oszacowanie kosztów i korzyści metod zwinnych
- Warsztat IV – Kanban dla dużych organizacji

✓ **WYZWANIA KOLEJNYCH LAT** – sprawdź nowe techniki wspierające agile w organizacji – przykłady wdrożeń holokracji po raz pierwszy na konferencji w Polsce!

agile.computerworld.pl

PARTNER MERYTORYCZNY: CT PARTNERS FOCUS ON CUSTOMER VALUE

PATRONAT: IPMA POLSKA, PMI POLAND CHAPTER

ORGANIZATOR: COMPUTERWORLD

Bartłomiej Prędki

Testowanie w rytmie Pop

Witam i o opaleniznę pytam! Krzyśku drogi, jak wiesz, lato to czas pięknej pogody, szumu fal, drinków z palemką i roznegliżowanych kob… ekhm… oraz drinków z palemką. Jak powszechnie wiadomo, jednym z produktów, który świetnie się sprzedaje w lecie jest muzyka, a konkretnie jeden z jej rodzajów – nic tak nie pasuje do klimatu lata jak muzyka Pop.

Siedząc więc niedawno na jednej z plaż na drugim końcu Europy, doszedłem do wniosku, że w zasadzie profesja nasza znakomicie wpasowuje się w klimat tejże muzyki. W zasadzie wstając rano i mając w perspektywie odwieczną walkę z programistami, analitykami i managementem, jedyne co można sobie powiedzieć to „Czasami czuję, że się załamię i rozpłaczę" [Freddie Mercury – Living on My Own]. A motywem przewodnim przez cały dzień mógłby być przebój Bee Gees – Staying Alive.

W rzeczywistości nie jest jednak tak źle – zresztą o frajdzie i udręce testowania już kiedyś pisaliśmy. Spora część komunikacji może się odbywać za pomocą muzyki Pop. Ot, chociażby po znalezieniu krytycznego błędu, możemy zadzwonić do szefa programistów i rozpocząć słowami „Dzwonię tylko po to, żeby Ci powiedzieć…" [Stevie Wonder - I just call to say I love you]. Któż bowiem nie zasługuje na naszą miłość tak jak programiści tworzący kiepski kod – dzięki nim mamy pracę! Mimo wszystko lepiej, żeby osoba po drugiej stronie nie spytała, jak bardzo, bo mogłoby się okazać, że musielibyśmy skłamać odpowiadając „Prawdziwie, do szaleństwa, głęboko" [Savage Garden - Truly Madly Deeply].

Gorzej, jeśli trzeba się wybrać do programistów osobiście – gdy wkracza się na ich teren mrucząc pod nosem „Idąc przez śmierci cienistą dolinę" [Coolio – Gangsta's Paradise] świat nie wydaje się już taki różowy. Można jednak bardzo łatwo wytrącić ich z równowagi mówiąc przyciszonym głosem „z każdym fixem, który robisz, będę cię obserwował" [The Police – Every Breath You take]

Jednym z moich ulubionych obszarów jest analiza wymagań – jak wiadomo, w ideal-

nym świecie proces testowania zaczyna się we wczesnej fazie projektu. Ślęcząc nad dokumentacją i czytając jakiś fragment, często mam ochotę zapytać analityka „Powiedz mi, dlaczego…". Chociaż w tej fazie (wierzcie mi, nie tylko w tej) i tak najlepiej potrafią się wykazać kierownicy projektu, którzy w ostatniej chwili zgłaszają swoje poprawki – „wiem, że jest za późno, ale chcę żeby tak było" [BackStreet Boys - I Want It That Way].

Spinając nieco klamrą proces testowy, wiadomo, że na koniec projektu trzeba przygotować raport z testów. Oczami wyobraźni widzę miny kierownika projektu, klienta, członków zarządu, którzy czytają o fatalnej jakości produkcie. Tak, tak, to te same osoby, które skąpiły kasy na testy – a im dalej brną w raport obnażający ich niekompetencje, tym bardziej chcą powiedzieć „Nie mów, bo to boli" [No Doubt – Don't Speak].

Jak oczywiście wiesz, zdarzają się jednak jaśniejsze strony naszej profesji. Szczególnie, gdy bierzemy udział w eventach rekrutacyjnych typu targi pracy, itp. Na przykład rozmawiając z jakąś kandydatką na praktyki, „młodą i słodką, zaledwie 17-letnią" [ABBA – Dancing Queen] świat od razu nabiera innych kolorów. A wieczorem, w poczuciu dobrze spełnionego obowiązku, można wracać do domu tanecznym krokiem niczym John Travolta w „Gorączce sobotniej nocy", a patrząc później w lustro z szelmowskim uśmiechem zaśpiewać sobie niczym Rod Steward - Do Ya Think I'm Sexy?

Jak więc widzisz, Drogi Kolego, muzyka Pop jest niczym testowanie, a testowanie jest jak muzyka Pop. Aczkolwiek jestem głęboko przekonany, że masz na ten temat odmienne zdanie…

P.S. Oczywiście moim ulubionym zespołem muzycznym jest Crash Test Dummies ;)

AUTOR

Bartłomiej Prędki

Swoją karierę rozpocząłem w 2004 roku jako tester aplikacji mobilnych przeznaczonych na rynek masowy. Przez następne lata nabierałem doświadczenia w obszarach Testowania i Jakości Oprogramowania, koncentrując się głównie na telekomunikacji oraz systemach mobilnych. W czasie swojej kariery byłem zaangażowany w testowanie, zarządzanie procesami testowania, szkolenia, wsparcie techniczne , analizę wymagań, rekrutację oraz tworzenie i inspekcje dokumentacji technicznej.

Poza doświadczeniem mobilnym i telekomunikacyjnym, zaangażowany także byłem w obszary systemów bankowych i finansowych. Obecnie pracuję na stanowisku Lidera Zespołu QA.

Posiadam dwa certyfikaty ISTQB poziomu zaawansowanego: Techniczny Analityk Testów oraz Kierownik Testów.

Mieszkam i pracuję we Wrocławiu.

Krzysztof Chytła

Testowanie z Rockowym pazurem

Uszanowanie! … i zatroskanie o zdrowie Pana Redaktora pragnę wyrazić. Rozumiem, że drinki, słoneczko i plaża (z nadmiaru wrażeń palemka lekko bije, pomroczność jasna czy inna choroba filipińska), ale zaraz Pop? A gdzie – do cholery, ja się grzecznie pytam – podział się stary dobry seks, dragi, a przede wszystkim rock'n'roll? Rozczarowanie. De gustibus non est disputandum.

Opalenizna? Pozwolę sobie odpowiedź rozpisać to na kilka przypadków testowych: opalanie z piwkiem w prawej ręce – status zielony, w lewej – zielony, na kajaku (bez piwka oczywista oczywistość) – zielony, na leżaczku – zielony. Rezultat: paradoksalnie czerwony! Czym się strułem, tym się leczę – okładami z zimnego piwka: raz z prawej, raz z lewej, na leżąco i surfując (po sieci) na kanapie.

Testowanie jak Pop? Szok i niedowierzanie. Bartku, szczepienia były? Złożę to niedorzeczne twierdzenie na karb lekkiego udaru słonecznego i spieszę wyprowadzić z błędu.

Nie od dziś wiadomo, że testowanie jest jak Rock. Tam przykręć, tu podkręć, kable, przewody, taśma MacGyvera - środowisko stoi; z podziałem na sekcje funkcjonalną (wokalną) i regresyjna (perkusyjną), a Test Lead pomyka z eksploracyjną solóweczką (gitarową). Nagle elektryczny trzask! Hop - wyskakuje bug w połączeniach między komponentami [Van Halen - Jump]... Zawsze brakuje tej ostatniej śrubki [Queen – I'm going slightly mad]. Wypisz, wymaluj – scena jak przed testami akceptacyjnymi u klienta (publiczności). Pójdę nawet krok dalej ciemną doliną i powiem, że jest jak Heavy Metal! Biznes rzuca wyzwanie testerom, żądając „testów przez najlepszych" [Manowar – Number One], by ocaleć od zapomnienia i niesławy dostarczenia. Czy samo to nie wystarczy za dowód? Spokojna Twoja rozczochrana, jest tego więcej. Niedoświadczony adept Sztuk Testerskich może, ale Ty – okrzepły w ciężkich bojach

lider testów, weteran – z takim tekstem o poranku? Niepodobne. Spodziewałbym się raczej postawy obrońcy błądzących w ciemności niewiedzy (ślepych na koszt niskiej jakości), po którego słowach metryk zapada cisza ostateczna [Blind Guardian – And then there was silence]. Nie zapominaj o „ogniu, który płonie w każdym z nas, usłysz harmonię raportów – przyznaj, to magia jakaś jest" [Queen – A kind of magic].

Uff, teraz mówisz jak człowiek. Pewnie, że nie jest tak źle. Dobrze wypoczęty – zurlopowany – tester zasiada do pracy nad swym ulubionym, dawno niedręczonym systemem z „die, die, die my darling" [The Misfits] na ustach strzelając z kostek (freelancer o elastycznej etyce mógłby do zestawu dołączyć słoik whiskey). To się nazywa duch walki, którego spokojnie starczy do końca dnia, jeśli nie tygodnia.

Nie lękaj się konfrontacji - z góry bowiem wiadomo, komu bije dzwon [Metallica – For whom the bell tolls], gdy moc wyników regresji jest z Tobą.

Analiza wymagań we wczesnej fazie projektu to jawne „igranie z szaleństwem" [Iron Maiden – Can I play with madness] i wróżenie z fusów – one są jak piekło, nigdy nie zamarzają . Ale jak nie my, to kto? Prawdziwy tester ma jasny cel i bierze to wszystko na klatę [Queen – I want it all] nawet, jeżeli to bilet do piekła w jedną stronę [The Darkness – One way ticket to hell].

Testowanie – moja miłość - jest jak skała (ang. rock) pośród morskich fal [Masterplan – Love is a Rock], opierająca się nieobiektywnym oskarżeniom o marną jakość rozwijanego w pocie czoła oprogramowania. Skoro miłość jest jak skała, a testowanie to moja miłość, to testowanie jest jak Rock \m/.

PS.
Byłbym zapomniał. Uważaj z tymi roztańczonymi 17-tkami. Wszystko ładnie pięknie a tu rozsierdzony tatuś przyłapuje Was in flagrante z tekstem: „Hej! Co to ma znaczyć?! Ona ma zaledwie 16 lat!" [Manowar – Warlord].

AUTOR

Krzysztof Chytła

Kierownik, projektant i automatyk testów z wieloletnim doświadczeniem w domenie systemów wbudowanych. Zdobywał doświadczenie w dużych projektach o międzynarodowym zasięgu, zapewniając najwyższą jakość produktu. Tester z krwi i kości z ciekawością i wnikliwością analizujący prężnie rozwijający się świat nowych technologii. Absolwent Wydziału Elektroniki Politechniki Wrocławskiej. Trener i coach, pasjonat zdobywania i dzielenia się wiedzą.

W życiu prywatnym fan literatury fantastycznej, fantastyczno-naukowej i gier planszowych przy szklaneczce zacnej whisky - najlepszej przyjaciółce redaktora.

Tomasz Osojca

Co jest czym czego*? - Jeśli chcesz coś naprawdę zepsuć, użyj do tego komputera

Prelegent na TESTWAREZ

Prawo Murphy'ego przywołane w tytule, jak wszystkie, zawiera w sobie dwa poziomy prawdy - oczywisty i głęboki.

Oczywisty: komputery przetwarzają informacje znacznie szybciej (przynajmniej w warstwie świadomej), nie męczą się, mogą wykonywać zadania poza normalnymi godzinami pracy, nie rozpraszają się przy powtarzalnych czynnościach - więc jeśli zleci się komputerowi czynność psucia to zostanie ona wykonana szybciej i dokładniej, niż zrobiłby to człowiek.

Głęboki: użycie komputera powoduje zwielokrotnienie problemów, jeśli użyjemy komputera do zadania, którego sami nie rozumiemy (w dodatku do rozwiązania którego, jak wskazują inne prawa Murphy'ego, wcale nie jest wymagany komputer), to na końcu nie będziemy rozumieli tego zadania jeszcze bardziej...

***

Nowy programista jest oprowadzany po firmie.
- o, a tam zespół Quaiów - mówi przedstawiający, mając na myśli Testerów,
- a, czyli ludzie od psucia - z uznaniem (chyba, bo przecież nie obrażałby nowo poznanych kolegów) mówi Programista.

Tak, część testerów jest dumna z tego, że umieją psuć. Uważają wręcz, że ta umiejętność jest ich główną kompetencją

(miękką?). „Czego się nie dotknę, to od razu przestaje działać", „potrafię wywalić każdy system w 5 minut" - to tylko część sprawności z dumą wieszanych przez testerów nad biurkami.

***

„Automatyzujący" - najwyższa forma rozwojowa testera - ktoś mi ostatnio tak powiedział uzasadniając tym samym, dlaczego chce się nauczyć automatyzacji testów. I coś w tym chyba jest, patrząc na wysokość zarobków, pewną specyficzną wyższość z jaką „automatyzujący" traktują zwykłych klikających „manualnych", tematykę większości prelekcji konferencyjnych i spotkań testerskich czy chociażby wiarę managerów w to, że „zautomatyzowanie testów w całej firmie" rozwiąże wszystkie problemy z jakością...

***

Automatyczne testy wykonuje się za pomocą komputera - czyli, zgodnie z przytoczonym prawem Murphy'ego, psucie za pomocą automatów będzie skuteczniejsze...

Tylko czy w testowaniu faktycznie chodzi o psucie? I czy w całym testowaniu? Bo przecież "dylemat": manualny czy automatyczny odnosi się tylko i wyłącznie do fazy wykonania testów - a co z resztą procesu testowego?

I ostatecznie: czy traktując automatyzację jako Świętego Grala testów nazbyt często nie ocieramy się o głęboką prawdę przytoczonego na wstępie prawa?

DOPISEK

*Tytuł „Co jest czym czego" zapożyczyłem z „Wstępu do Imagineskopii" Sledzia Otrębusa Podgrobelskiego. Z tegoż dzieł pochodzi również poniższy cytat stanowiący dla mnie nieodmiennie inspirację przy rozwiązywaniu najbardziej nawet złożonych problemów informatycznych:

„twierdzenia (...) o wczuwaniu się jako metodzie poznawczej teoretycznie najsłuszniejszej i metodycznie najbardziej poprawnej wskazują - jeśli uwzględnić niezastąpioną rolę wyobraźni w procesach wczuwania się - na niebezpieczeństwo dysproporcji, rozziewu (hiatus) między wciąż pomnażającą się rzeczywistością a wyobraźnią. Logicznym wnioskiem wypływającym z powyższego musi być zatem pilny postulat powiększenia wyobraźni stosownie do potrzeb poznawczych."

Tomasz Osojca

Funkcja oceny testowania

Prelegent na TESTWAREZ

Wprowadzenie

Funkcja oceny, nazywana również funkcją przystosowania, jest jednym z podstawowych elementów wielu systemów sztucznej inteligencji. Formalnie pojęcie to odnosi się do algorytmów genetycznych [Goldber 1998, Osojca 2003], gdzie służy do podjęcia decyzji, które osobniki z populacji mają przetrwać do następnego pokolenia. Bardziej ogólnie autor stosuje pojęcie „funkcja oceny" dla różnych metod używanych w celu określenia, na ile dobre jest działanie systemu bądź algorytmu sztuczno-inteligentnego. Jest to bardzo ważny element sztucznej inteligencji, ponieważ bez oceny nie może być nauki i adaptacji do zmieniających się warunków. A to właśnie „umiejętność" samodzielnego odkrycia metody działania oraz dynamiczna „zdolność" korygowania własnych błędów odróżnia systemy sztuczno-inteligentne od „zwykłych" programów komputerowych.

Na gruncie sztucznej inteligencji zostało opracowanych wiele metod i podejść do funkcji oceny, tabela 1. zawiera kilka przykładów.

W opinii autora opracowanie „Funkcji Oceny Testowania" jest jednym z głównych wyzwań stojących na drodze do budowy systemu, który w inteligentny sposób testowałby oprogramowanie (lub też realizował w inteligentny sposób którąś z elementarnych czynności procesu testowego).

TABELA 1

Zadanie	Technika AI	Przykład funkcji oceny
Znalezienie optimum funkcji	Algorytm genetyczny	Różnica wartości funkcji badanej, obliczona dla poszczególnych osobników w populacji
Klasyfikacja (rozpoznawanie) wzorców	Sieć neuronowa	Zbiór uczący zawierający poprawnie sklasyfikowane wzorce
Podejmowanie decyzji w złożonym środowisku	System ekspertowy	Ocena efektów decyzji lub porównanie z decyzjami "żywych" ekspertów
Znalezienie najlepszej strategii gry w szachy	Algorytm genetyczny	Turniej szachowy między wypracowanymi dotychczas strategiami
Opanowanie umiejętności rozmowy w języku naturalnym	System hybrydowy	Wirtualne "nagrody" (wzmocnienie) przyznawane przez nauczyciela-rozmówcę

Problem z oceną testowania

Testowanie oprogramowania jest procesem złożonym, realizującym różne cele i różnie rozumianym przez różnych interesariuszy projektów IT. Prawidłowa i miarodajna ocena testowania nastręcza wiele problemów nie tylko w trakcie projektowania systemów sztuczno-inteligentnych, ale również w codziennej praktyce testowania i zarządzania testowaniem. Oto krótka charakterystyka problemów z oceną testowania na różnych poziomach złożoności.

Zleć i zapomnij

Zlecanie testów wyspecjalizowanym firmom staje się coraz bardziej popularne. Mają tutaj zastosowanie wszystkie zalety outsourcingu, takie jak: ekspercka wiedza specjalistów, elastyczne skalowanie zasobów, przeniesienie ryzyka związanego z dostępnością specjalistów itp.. Problem jest tylko z oceną tego, czy zlecona praca testowa została wykonana dobrze. Dlaczego autor uważa, że jest to problem? Prześledźmy poniższe strategie oceny pracy testowej z punktu widzenia zleceniodawcy i zleceniobiorcy (tabela 2.)

Istnieje jeden niezawodny sposób oceny jakości wykonanej pracy testowej - zrobić testy jeszcze raz samemu. Proste i pewne - tylko, że zupełnie nieefektywne czasowo i kosztowo.

TABELA 2

Strategia oceny	Zleceniodawca	Wykonawca	Komentarz
Liczba godzin spędzonych przez kontraktorów na testowaniu	Ale co właściwie zostało w tym czasie zrobione?	Godziny zaraportowane i zapłacone - robota zrobiona.	Każdy, kto testował, zdaje sobie sprawę z tego, że można siedzieć 8 godzin klikając w testowany system i nic sensownego nie zrobić...
Liczba wykrytych defektów (najlepiej z uwzględnieniem wagi)	Super, że zaraportowano x defektów, ale czy to już wszystkie? lub Nie zależy mi na defektach - chce mieć dobry system.	Liczba defektów nie zależy od nas, tylko od tego ile defektów jest w oprogramowaniu, dostępności środowiska...	Liczba defektów jest raczej miarą jakości systemu, niż jakości prac testowych.
Liczba wykonanych przypadków testowych	Czy dane przypadki są wystarczające do zapewnienia jakości systemu?	A co jeśli nie możemy testować, bo nie działa środowisko testowe lub jest blocker na początku procesu?	Jest to dobra strategia, jeśli faktycznie zleca się samo wykonanie przypadków wcześniej zaprojektowanych przez wewnętrznych analityków testowych - choć i tu spostrzegawczość i dokładność w wykonaniu może być bardzo różna. W innej sytuacji jest to tylko drobna modyfikacja strategii z liczeniem godzin - nie wnosząca żadnej nowej jakości w omawiane zagadnienie
Liczba defektów po wdrożeniu na produkcji	To jest świetny sposób oceny pracy i rozliczenia z kontraktorami ... choć może nie do końca, bo jeśli dopiero po wdrożeniu okazuje się, że zlecona praca testowa była wykonana źle to zazwyczaj jest za późno na sensowną reakcję.	Ale my nie mieliśmy wystarczającego wpływu na usuwanie defektów, decyzję o wdrożeniu...	To może być bardzo dobra strategia o ile opera się na bliskiej i długotrwałej współpracy oraz świadomym podziale ryzyka (wtedy też taki outsourcing musi kosztować więcej niż po prostu wynajęcie testerów, ale może być bardzo skuteczny i efektywny)

Czy dany przypadek testowy jest dobry?

Jeśli podchodzimy do testów inżyniersko stosując formalne techniki projektowania testów, możemy próbować określać „siłę" przypadku testowego w oparciu o technikę, na podstawie której został zaprojektowany. Wiemy, że wartości brzegowe są „silniejsze" od klas równoważności, pokrycie decyzji jest „silniejsze" od pokrycia linii kodu itp. Pytanie tylko, o ile silniejsze?

W domenie systemów krytycznych z punktu widzenia bezpieczeństwa, normy (na przykład DO-178B/DO-178C Avionics) podają jasno, który poziom pokrycia ma być zastosowany w testach określonych fragmentów systemu. Dla aplikacji biznesowych brak takiego punktu odniesienia.

Dodatkowym problemem jest brak danych statystycznych. Firmy informatyczne niechętnie dzielą się informacjami o swoich „błędach" oraz o jakości procesu wytwarzania oprogramowania, w tym procesu testowego. Trudno bez takich masowych danych budować miarodajne podstawy oceny „siły" poszczególnych technik projektowania przypadków testowych w konkretnych kontekstach zastosowań.

W codziennej praktyce testowania już samo odniesienie do formalnych metod projektowania testów jest zazwyczaj pomijane na rzecz „wyczucia" testerów. Autor wielokrotnie słyszał „ja bym tu jeszcze dopisał ze dwa przypadki". Pytania zmierzające do sformalizowania takiego „wyczucia" typu: „na jakiej podstawie?", lub „co kolejny przypadek wniesie do już istniejącego zbioru" pozostają bez odpowiedzi. Powodem może być brak wiedzy inżynierskiej testerów albo też braku poczucia, iż takie odpowiedzi mogą być przydatne w bieżącej pracy.

Był potrzebny, bo wykrył defekt

Potocznie, za dobry (przydatny) przypadek testowy przyjmuje się taki, za pomocą którego udało się wykryć defekt. Taki sposób oceny jakości przypadku testowego może być dobry. Ma tylko jedną wadę - pozwala ocenić przypadek dopiero po jego wykonaniu (często wielokrotnym w testach regresyjnych).

Ważniejsza, w ocenie autora, jest ocena przypadku testowego w kontekście stopnia, w jakim redukuje on niepewność (ryzyko) związane z jakością systemu. Inaczej mówiąc dobry przypadek testowy maksymalizuje liczbę informacji (z uwzględnieniem obszarów technicznie trudnych i biznesowo istotnych) jednocześnie minimalizując nakład niezbędny na wykonanie testu (liczba kliknięć, pracochłonność przygotowania danych testowych). Oceniając przypadki testowe w taki sposób, bazując na wiedzy o „mocy" formalnych metod projektowania oraz danych statystycznych, moglibyśmy uzyskać umiejętność miarodajnego i wiarygodnego oceniania sensowności wykonywania poszczególnych przypadków. Dodatkowo mogłoby to stanowić podstawę do projektowania uniwersalnych, reużywalnych, „mocnych" przypadków testowych (analogicznie do wzorców projektowych w programowaniu).

To jest tak jak z oceną mostów - można budować mosty na chybił trafił (również wzdłuż rzeki) i po fakcie liczyć ile samochodów na dobę przez nie przejeżdża - tak, aby znaleźć ten najlepszy most. Ale lepiej jest na podstawie analizy ryzyka wybrać najlepszy most już na etapie pro-

jektu i wybudować od razu ten właściwy. Tak samo lepiej oceniać przypadki testowe na podstawie świadomego projektu, niż po efektach ich wykonania.

Dobry, bo zielony

Jaki to jest dobry raport testowy? Dla sporej części menedżerów IT to jest raport, w którym wszystkie przypadki testowe są oznaczone jako pozytywne. Autor spotkał się z takim podejściem nawet w organizacjach, w których ogólna dojrzałość procesu testowania była wysoka. Jeśli połączy się „zielony" raport z ogólnie znanymi prawdami testowymi: „test totalny nie jest możliwy" oraz „każdy nietrywialny system zawiera defekty" - to otrzymujemy jednoznaczną konkluzję: raport, w którym wyniki wszystkich przypadków testowych są pozytywne, jest raportem nieprawdziwym lub, co najmniej, niekompletnym. Zazwyczaj taki stan raportu końcowego jest osiągany poprzez ograniczanie zakresu testów lub przenoszenie testów na kolejne wydanie. Znane są autorowi również sugestie menedżera projektu dla zespołu testów w stylu „jeśli nie da się tego sprawdzić na środowisku testowym to weź to na klatę i oznacz jako wykonane pozytywnie" (na szczęście zespół nie był "zielony" i się nie zgodził...). Niezależnie od sposobu osiągania „zielonego" efektu - jeśli początkowe założenie jest takie, aby na koniec wszystkie przypadki były pozytywne, to mamy gwarantowane przekłamywanie istotnej informacji zarządczej. Informacja o tym:

- co nie zostało przetestowane (i dlaczego),
- co nie działa,
- jakie były problemy w trakcie testów,
- jakie pozostało ryzyko

powinna być kluczowa dla menedżera podejmującego decyzję o rozpoczęciu lub wstrzymaniu wdrożenia.

Idealna byłaby sytuacja, w której zgromadzone w procesie testowania dane można by było przeliczyć na jednoznaczną informację o ryzyku wdrożenia systemu w jego aktualnym stanie. Do takiego przeliczenia bardzo przydatne mogą być techniki sztucznej inteligencji (takie jak np.: logika rozmyta, drążenie danych) . Aby to przyniosło pożądany efekt, konieczne jest zbieranie danych w systematyczny (formalny) sposób oraz posiadanie podstawy statystycznej wzmiankowanej przy okazji omawiania przypadków testowych.

Mierz i rządź!

Bardzo dobrym narzędziem do oceny jakości testów są metryki. Z założenia wiążą się z zaplanowanym (formalnym) zbieraniem danych, które następnie przetwarzane są w informacje na temat jakości naszego postępowania. Ciekawe opracowania na temat metryk znajdziemy w (Kijewski 2010, Ziajka 2013, Kan 2060)).

W kontekście Funkcji Oceny Testowania metryki zdecydowanie są istotne. Autor dostrzega jednak kilka problemów wymagających opracowania przed zastosowaniem metryk w sztuczno-inteligentnym systemie testującym:

- opracowane metryki służą w większości do mierzenia jakości testowanego systemu, a tylko pośrednio do oceny testowania jako takiego - konieczne jest opracowanie nowych metryk lub adaptacja istniejących,
- brak jest szerokiej bazy statystycznej wyników metryk w różnych organiza-

cjach, projektach i kontekstach - taka informacja (oczywiście zanonimizowana) pozwalałaby na wyciąganie bardziej precyzyjnych wniosków. Brak takiej informacji porównawczej powoduje konieczność zebrania historii wyników w ramach danej organizacji, co opóźnia uzyskanie zwrotu z inwestycji w metryki,

- metryki pozwalają na ocenę „po fakcie", dlatego autor raczej upatruje ich zastosowanie w systemie ekspertowym pomagającym podejmować decyzje o kontynuowaniu lub zakończeniu testów niż w systemie, który miałby inteligentnie projektować przypadki testowe lub inteligentnie wykonywać testy.

Czynnik ludzki

Ocena pracowników sama w sobie jest trudna. Ocena jakości wykonywanych prac testowych jest jeszcze trudniejsza, ze względu na charakter pracy testerów. Menedżer zespołu testów napotyka problemy podobne do opisanych przy okazji outsourcingu.

Czasami powstają różne „dziwne" koncepcje - na przykład na jednej z konferencji wiele lat temu prezentujący test menedżer przedstawił pomysł stawiania zadania „nie dopuścić do wdrożenia" jako głównego dla zespołu testów.

Czasami osoby zarządzające testami uciekają się do pozornie miarodajnych „norm" w stylu „30 przypadków i 5 defektów dziennie na testera". Pozornie miarodajnych, ponieważ przypadek przypadkowi nierówny, nie mówiąc już o różnej pracochłonności niezbędnej do prawidłowego opisania znalezionego defektu. Różne jest też znaczenie poszczególnych testów i defektów dla zmniejszania poziomu ryzyka pozostającego w systemie.

Jak różnorodne może być postrzeganie „dobrego testera" i „dobrego testowania" pokazują bardzo ciekawe dyskusje panelowe, które miały miejsce na konferencji TestWarez 2013 (o etyce w testowaniu, czy tester musi umieć programować i czy testowanie jest jednoznaczne z jakością).

Do modelowania niezbędne jest zrozumienie

Sztuczna inteligencji definiowana jest jako dziedzina zajmująca się modelowaniem inteligentnego zachowania. Modelowanie nie jest jednak możliwe bez dobrego zrozumienia procesu, który chce się modelować. Szczególnie istotne jest zrozumienie metod oceny, czy dane postępowanie w ramach modelowanego procesu jest inteligentne („dobre").

Testowanie jest bardzo trudne w ocenie szczególnie ze względu na to, iż większość użytecznych danych otrzymujemy dopiero po wykonaniu pracy (przypadek wykrył defekt, awaria wystąpiła lub nie na produkcji). Można rozważać wykorzystania pomysłu z turniejem szachowym: opracowywane przez sztuczno-inteligentny system strategie testowe musiałyby być zastosowane do testowania tego samego kodu (projektu) i najlepsza - na podstawie metryk - wygrywa. Trudno sobie jednak wyobrazić takie podejście w „pędzącej" rzeczywistości projektowej.

Heurystyki pozwalające na ocenę planowanego podejścia przed wykonaniem testów opierają się głównie na wyczuciu testerów. Formalne metody są zazwyczaj pomijane lub brak dla nich niezbędnego odniesienia statystycznego.

Brak głębokiego zrozumienia, jaki jest cel testowania, co w testowaniu przynosi korzyści w danym kontekście, a w szczególności brak umiejętności definiowania formalnej funkcji oceny, blokuje przeniesienie testów na poziom inżynierii - niezbędny do zrobienia kolejnego kroku w kierunku prawdziwie sztuczno-inteligentnego systemu testującego.

REFERENCJE

1. Goldberg, D.E. (1998): Algorytmy genetyczne i ich zastosowania. Warszawa, Wydawnictwo Naukowo Techniczne. ISBN 83-204-2272-8
2. Osojca, T. (2003): Wpływ funkcji oceny na jakość reguł pozyskiwanych z sieci neuronowej za pomocą algorytmu genetycznego. Wrocław, Praca magisterska. Wydział Informatyki i Zarządzania Politechniki Wrocławskiej.
3. Kijewski, P. (2010): Metryki w procesie testowym – jak wykorzystać TestLink i JIRA jako źródła danych do wyliczania metryk. C0RE nr 4. [online, http://pl.coremag.eu/uploads/media/Metryki_w_procesie_testowym_core_PL_05.pdf, – dostępny 14.08.2014]
4. Ziajka, N. (2013): Dobór metryk w celu zbadania jakości oprogramowania. C0RE nr 14. [online - http://pl.coremag.eu/typo3conf/ext/drow_securefiles/pushFile.php?pid=744&file=uploads%2Fmedia%2FCore_magazine_nr_14.pdf – dostępny 14.08.2014]
5. Kan, S.H. (2006): Metryki i modele w inżynierii jakości oprogramowania. Warszawa, Wydawnictwo Naukowe PWN S.A. ISBN-83-01-14829-2

AUTOR

Tomasz Osojca

Doświadczony konsultant i trener z zakresu inżynierii testów oraz jakości systemów i procesów. Traktuje „Informatykę" jako „Naukę o Informacji" a nie „naukę o komputerach". Podkreśla, że:

„W projektach IT najważniejsze jest zrozumienie źródeł oraz sposobów przetwarzania informacji w organizacji a dopiero w drugiej kolejności dostarczenie środków technicznych czyli kodu źródłowego i serwerów. Innymi słowy, najważniejsza jest wartość użytkowa („Quality in Use" wg ISO/IEC 9126) zbudowana na poprawnej jakości zewnętrznej i wewnętrznej produktu zapewnianej przez efektywność procesów wytwórczych."

W trakcie kariery zawodowej uczestniczył w projektach w roli testera, test managera, automatyzera testów, specjalisty ds. zapewnienia jakości, konsultanta ds. jakości procesów. Zarządzał zespołem specjalistów z zakresu testów jako test manager, kierownik zespołu, dyrektor działu. Realizował projekty zarówno po stronie producenta oprogramowania jak i organizacji zamawiającego rozwiązanie. W projektach lokalnych, zespołach rozproszonych geograficznie oraz w projektach międzynarodowych (między innymi dla NATO). Swoją pasję przekazywania wiedzy realizuje jako trener szkoleń za zakresu testowania i jakości (w tym ISTQB) oraz instruktor ratownictwa górskiego (GOPR).

Prowadzi własną działalność gospodarczą pod nazwą QuICK. Nazwa jest skrótem od Quality by Imagination, Curiosity and Knowledge - co stanowi kwintesencję przymiotów niezbędnych do uzyskiwania odpowienio wysokiego poziomu jakości w branży IT.

Jego idée fixe to wykorzystanie technik sztucznej inteligencji w testach oraz inżynierii jakości systemów informatycznych. Tę ideę realizuje poprzez redagowanie strony www.AI4SE.org - otwartej platformy wymiany myśli o "sztuczno inteligentnej inżynierii oprogramowania".

Radosław Smilgin

Moje testowanie

Prelegent na TESTWAREZ

Podczas przygotowań do bycia prelegentem na konferencji zostałem poproszony o przygotowanie opisu własnej osoby. Kiedy przesłałem standardową notkę, otrzymałem informację zwrotną, że każdy pisząc o swojej pracy wspomina również o swoim hobby. Podsunięto mi nawet kilka propozycji - jedni grają w kręgle, inni lubią koty. Czy również ja mógłbym coś takiego napisać? Po głębszym zastanowieniu się odpowiedź brzmiała – nie. Nie mam większego hobby, niż moja praca, innego niż testowanie.

Ta prosta sytuacja kazała mi się pochylić nad tym, co robię i dlaczego to robię. Czy jestem komputerowym maniakiem, człowiekiem bez życia osobistego lub pracoholikiem? Nie. Po prostu lubię to, co robię i robię to, co lubię. Moja praca jest moim hobby.

Czytam tysiące blogowych wpisów o testowaniu i książki o testowaniu, bo jest to dla mnie przyjemność, a nie praca. Piszę o testowaniu, bo lubię to robić i jest to jedna z niewielu rzeczy, na których znam się naprawę dobrze. Wymyślam projekty związane z testowaniem i jakością oprogramowania takie jak TestArena – aplikacja do zarządzania testowaniem – czy TestingCup – impreza dla testerów. Myślę, że skoro ja mam frajdę przy pracy nad tym, to może ktoś będzie miał frajdę podczas uczestnictwa w imprezie, lub też podczas użycia oprogramowania. O jakości myślę również

wtedy, kiedy przygotowuję aplikacje dla szerszego rynku. Kompero jest aplikacją, która ma pokazać, że istnieje dobre, tanie, a nawet darmowe oprogramowanie dla firm. Nie przynosi mi pieniędzy, ale przynosi mi satysfakcję.

Definiując hobby ludzie myślą o czymś relaksującym, czymś, co robi się, aby zapomnieć o pracy. Zakładają poświęcenie czasu i wysiłku, niejednokrotnie pieniędzy, aby coś uzyskać lub miło spędzić czas. W moim przypadku, gdy hobby jest pracą, nie mam potrzeby o czymś zapominać albo od czegoś uciekać. Poświęcam czas i wysiłek, ale z pełną premedytacją. Dla wielu osób hobby, z definicji, nie przynosi rozczarowań, co niestety nierozłącznie wiąże się z pracą. Jednak układane puzzle mogą na końcu okazać się niemożliwe do ułożenia, a jazda szybkim samochodem może zakończyć się wypadkiem. W pasjo-pracy jest dokładnie tak samo. Czasami coś nam nie wyjdzie, a czasami nawet będziemy mieli sytuacje awaryjne.

Z łączenia pracy z hobby jest jeszcze jeden pożytek. Ja nie wydaję na nie pieniędzy, ja je na nim zarabiam. Można powiedzieć, że należę do tych szczęśliwców, którym płacą za robienie tego, co lubią.
Czy nie wolałbym jednak jeździć na koniu, podróżować po świecie czy też kolekcjonować znaczków? Chciałem kiedyś. Spróbowałem raz, nie znalazłem w tym nic na tyle ciekawego, żeby pasjonować się tym na resztę mojego życia. Wróciłem do testowania.

W rozmowie czasami przeciwstawia się moje poglądy zainteresowaniom ludzi wielkich. Dyskutując na temat balansu między życiem osobistym a pracą słyszałem: „Wiesz, Einstein uwielbiał żeglować,

a Rockefeller kolekcjonował żuki". I co z tego? Po pierwsze, nikt nie ceni Einsteina za jego pasję pływania, ale za jego teorie, które wynikały z jego pracy. Po drugie to, że Rockefeller przybijał szpilką owady do styropianu, nie znaczy, że nie czerpał przyjemności z tego, z czego jest znany, czyli z pomnażania pieniędzy. Sam lubię oglądać obrazy impresjonistów i na wakacjach nie przepuszczę okazji, aby na własne oczy zobaczyć któregoś ze swoich mistrzów, mimo tego będąc w Amsterdamie na EuroSTAR pojechałem tam, by słuchać Boltona, a nie chodzić po muzeach. Po trzecie daleko mi do wielkich tego świata i choć są to wzorce do naśladowania, to są jednak na poziomie lekko mówiąc - nieosiągalnym. Chciałbym mieć głowę Einsteina i środki Rockefellera, ale wystarcza mi to, co mam. A mam testowanie.

Testowanie jest fajne. Jest odkrywcze, ciągle się zmienia, przynosi emocje z każdym nowym projektem. Uczenie o testowaniu daje możliwość spotkania z innymi testerami i czerpanie z ich wiedzy, doświadczeń i projektów. Doradzanie w projektach jest jak pomaganie. Jeśli dodatkowo ktoś chce skorzystać z rady i wdraża Twoje pomysły, to jest to jeszcze wyższy poziom satysfakcji.

Pasjo-praca nie musi być objawem choroby psychicznej czy też dowodem na postępujący konsumpcjonizm. Może być prostym połączeniem przyjemnego z pożytecznym.

AUTOR

Radosław Smilgin

Niezależny konsultant i trener z zakresu testowania i zapewnienia jakości aplikacji.

Tester oprogramowania z zamiłowania. Właściciel serwisu testerzy.pl.

Autor publikacji z obszaru testowania i zapewniania jakości. Prelegent polskich i zagranicznych konferencji testerskich TestWarez, PTI, Free Test, CzechTest i wielu innych. Ambasador największych konferencji: EuroSTAR, Agile Testing Days, BTD.

Fan użyteczności i testowania w Agile.

Pomysłodawca i organizator TestingCup – Mistrzostw Polski w Testowaniu Oprogramowania.

Akredytowany przez Stowarzyszenie Jakości Systemów Informatycznych trener ISTQB oraz tłumacz sylabusa i słownika ISTQB.

Daniel Dec

„BDD u mnie nie działa”

Prelegent na TESTWAREZ

Postanowiłem napisać ten artykuł, ponieważ spotykam się z różnymi stwierdzeniami i opiniami na temat BDD (Behaviour-Driven Development). Spróbuję podsumować, skąd moim zdaniem biorą się te złe opinie. Oto garść moich przemyśleń.

Motywacja wprowadzanie BDD

Tutaj wszystko się zaczyna. Z jakiegoś powodu chcemy wprowadzić BDD, tylko czy powód jest wystarczająco dobry?

Presja otoczenia

Widziałem wiele sytuacji, w których wprowadzało się BDD dla samego wprowadzenia BDD, tylko dlatego, że „o tym się mówi”, lub wydaje się lekiem na pewne problemy projektowe. Taka skrajna motywacja jest rzadkością, ale warto sobie uzmysłowić, czy nie ulegamy zewnętrznym czynnikom, tylko po to, aby móc się pochwalić, że my już też mamy BDD w projekcie.

Automatyzacja testów

„Wprowadzamy BDD, bo chcemy automatyzować testy „ - automatyzacja nie potrzebuje BDD, tak samo jak BDD nie wymusza automatyzacji. Jeżeli myślimy tylko o samej automatyzacji, to BDD będzie tylko niepotrzebnym narzutem. Oczywiście prawidłowo napisane scenariusze razem z przykładami (jako kryteriami akceptacji) stanowią idealne przypadki do zautomatyzowania, ale trzeba jasno sobie powiedzieć, że automatyzacja nie jest tutaj głównym celem, bardziej naturalnym następstwem. Skoro wiemy już jak ma działać nasza aplikacja i scenariusze opisują jej zachowanie w zrozumiały sposób, to grzechem byłoby nie skorzystać z auto-

matycznej informacji zwrotnej po każdej wprowadzonej zmianie.

Lepsze TDD

Programiści używają TDD i przeczytali, że BDD to pełniejsze TDD (Test Driven Development). Istotnie, scenariusze powstające w procesie BDD mogą karmić nasz proces TDD. Czasami jednak BDD jest rozumiane tylko jako wyższa warstwa wymagań nad TDD. Trzeba popatrzeć na cele. TDD pozwala tworzyć małe przyrosty kodu spełniające wymagania, ale w bardzo wąskim zakresie, programista skupia się nad pokryciem nisko-poziomego testu, skupia się nad projektowaniem interfejsu i klas. Nie skupia się w tym momencie dokładnie nad tym, czy to jest dokładnie to, czego klient potrzebuje, jak będzie tego używać i czego może brakować w funkcjonalności. Oczywiście po przejściu pełnego cyklu może zweryfikować czy kod spełnia scenariusz BDD, ale to podczas pisania scenariuszy odkrywamy i definiujemy wraz z biznesem zachowanie aplikacji - w tym momencie nie myślimy o kodzie. W cyklu BDD chcemy jak najlepiej zrozumieć potrzebę klienta, chcemy rozmawiać wspólnym językiem i precyzować, poprzez przykłady, zamierzone działanie aplikacji. Zatem stwierdzenie, że BDD jest tylko lepszą wersja TDD jest dużym uproszczeniem. Tym bardziej, jeżeli scenariusze i kryteria akceptacyjne miałyby być pisane tylko przez programistów - dla programistów, to ciężko mówić o BDD.

Więc po co BDD?

BDD jako narzędzie komunikacji:

- Pozwala zdefiniować problem biznesowy, który chcemy rozwiązać - wartość biznesowa
- Pomaga odpowiedzieć na pytanie, w jaki sposób problem będzie rozwiązany - jakich zachowań oczekujemy
- Precyzuje, kiedy uznamy, że program rozwiązuje nasz problem - kryteria akceptacji, które stają się testami akceptacyjnymi
- Odkrywa, co się powinno stać w przypadku, kiedy … - pomaga rozpatrzeć sytuacje graniczne i nietypowe
- Łączy punkty widzenia wszystkich stron: biznesowej, projektowej oraz weryfikującej
- Wprowadza wspólny język zrozumiały dla wszystkich interesantów
- Pomaga odkrywać niezdefiniowane i ukryte obszary wymagań i potrzeb biznesu

Typowe problemy z BDD

Niezrozumiałe scenariusze

Zazwyczaj jest tak, że stawiając pierwsze kroki w BDD, ciężko pisać od początku idealne scenariusze. Jest kilka elementów, które mają negatywny wpływ na ich jakość:

- Scenariusze pisane są przez jedną osobę (programista, tester) bez udziału stron biznesu (właściciel produktu, analityk biznesowy, klient)
- Pisane technicznym językiem lub jego mieszanką z językiem domenowym
- Brak jednoznacznego słownika - inaczej rozumiemy te same wyrażenia np. pojęcie „użytkownika” pojawia się w miejscach gdzie tak naprawdę mamy na myśli konkretną role użytkownika, co sprawia, że w zależności od kontekstu „użytkownik” jest kimś innym

- Zawierają dużą ilość kroków, są przerośnięte i skomplikowane - za mała granulacja
- Przerośnięta asercja „Then" - ciężko określić, co jest celem danego przykładu i jakie warunki są istotne do spełnienia
- Zduplikowane przykłady - nie chcemy wypisywać wszystkich możliwych kombinacji w kryteriach akceptacyjnych - skupiamy się na tych istotnych, które nas czegoś nowego uczą o tworzonym produkcie
- Oczami użytkownika

Opisując scenariusze, klienci czy właściciele produktu mają czasami tendencję do przedstawiania wymagań na poziomie warstwy interfejsu użytkownika zamiast na skupieniu się na problemie biznesowym. Końcowy użytkownik widzi oczami wyobraźni przyciski, formatki, rozwijane menu i inne elementy graficzne. W późniejszej fazie jest to ważne, wygląd i design, ale BDD nie służy do modelowania GUI. Chcemy się dowiedzieć „jak ma działać", „jak się ma zachowywać" aplikacja. Naszym zadaniem jest wydzielenie warstwy abstrakcji ponad szczegóły implementacyjne. Interesuje nas proces i logika.

Tendencję do pisania scenariuszy od GUI mają testerzy, którzy automatyzują właśnie od interfejsu użytkownika. Wtedy przykłady scenariusza logowana wyglądają mniej więcej tak:

Given I enter user_name to „User name" field
And I enter my valid password to „Password" field
When I click on „Subbmit" button
Then I see label status „Logged in"

W tym prostym przykładzie można się domyśleć, że chodzi o funkcjonalność logowania, jednak jeżeli sobie wyobrazimy wielkie przykłady z kliknięciami przycisków i wypełnianiem formatek, może być ciężko aby wyłapać, jaką funkcjonalność przedstawia scenariusz. Mając ogólną postać przykładu:

Given a pair of a valid user name with valid password
When I log to the system
Then I should be properly logged in

Jeszcze nie wiemy nic o tym, czy proces logowania wymaga wypełniania formatek i klikania przycisków czy wysłania zapytania do usługi zajmującej się logowaniem.

Scenariusze stają się niepotrzebnym narzutem

Jeżeli zespół zaczyna czuć ból podczas tworzenia scenariuszy to znaczy, że coś jest nie tak. Czy scenariusz jest wystarczająco ciekawy, aby go zapisywać? A może jest zmodyfikowaną wersją poprzednich kilkunastu? Jak wyglądają kryteria akceptacji, czy mamy interesujące przykłady, czy poprzestaliśmy na pozytywnych?

Trzeba dotrzeć do motywacji zespołu, jeżeli jest jej brak i scenariusze zamiast pomagać, zaczynają przeszkadzać, to zastanowiłbym się czy powinniśmy je pisać. BDD ma nam pomagać zrozumieć, co chcemy stworzyć, co jest ważne w danej iteracji dla klienta oraz jaką wartość wniesie napisany kawałek kodu, po którym nasze kryteria akceptacyjne staną się zielone. Jeżeli jesteśmy w miejscu „oczywistego" obszaru aplikacji i wszyscy wiedzą czego oczekujemy po jego funkcjonalności, nie potrzebujemy niczego odkrywać i dowiadywać się, to wystarczającym narzędziem będą testy

powstałe w procesie TDD, lub inne - ważne, aby gdzieś zapisać te „oczywistości”, by móc do nich wrócić.

BDD - narzędzie testów automatycznych

Wielu ludzi, z którymi rozmawiam o BDD przeważnie mocno skupia się na używanych narzędziach oraz szablonie kryteriów akceptacyjnych lub na tym, jaki framework testów jest zapięty do automatyzacji BDD. Spotykam się też z opiniami, że samo BDD to narzędzie do testów automatycznych. BDD to głównie koncepcja uporządkowania procesu komunikacji pomiędzy klientem a zespołem developerskim. Oczywiście chodzi o proces komunikowania wymagań. Jeżeli zapominamy o tym podstawowym założeniu i zaczynamy myśleć o BDD jako o narzędziu do automatyzacji, to łatwo możemy wpaść w pułapkę. Narzędzia implementujące składnie Gherkina są same w sobie „klejem” pomiędzy specyfikacją (tekstem, scenariuszami) a kodem. To dużo, a jednocześnie mało. Dużo, bo dzięki temu klejowi możemy uruchamiać naszą specyfikację na naszej aplikacji. Mało, bo tak naprawdę nie rozwiązuje to żadnych problemów z samą automatyzacją. Narzędzia BDD wprowadzają warstwę abstrakcji do naszych testów. Dzięki nim testy można czytać jak specyfikację. Jeżeli mamy problem z testami automatycznymi - z architekturą, utrzymaniem, brzydkim kodem - to wprowadzenie w tym momencie BDD może te problemy jeszcze bardziej pogłębić, na pewno jednak niczego nie naprawi. Naprawiajmy problem z testami tam, gdzie on powstał, nie na zewnątrz.

BDD bez BDD

Nie jest czymś nietypowym wprowadzanie narzędzi typu Cucumber do projektu bez wprowadzania pełnego procesu BDD czy ATDD. Nikt nam nie zabroni komunikować się ze sobą w projekcie za pomocą scenariuszy Gherkinowych. Nikt też nie zabroni zastąpić typowych przypadków testowych, kryteriami akceptacyjnymi w postaci przykładów do scenariusza. Nawet jeżeli nie mamy bezpośredniego dostępu do biznesu, albo jeszcze nie udało nam się wypracować wspólnego pisania scenariuszy, jeszcze nie znaczy, że nie możemy użyć narzędzi BDD do komunikacji i dokumentowania wymagań. Kluczem do sukcesu jest odpowiednia hierarchia i zarządzanie scenariuszami na różnych poziomach. U nas w projekcie na poziomie modułu programiści mają bardzo techniczne scenariusze. Mogliby je zapisać w TDD, ale większą logikę biznesową łatwiej rozpisać na scenariusze. Właściciele produktu mają Gherkiny na swoimi poziomie - ogólne, my jako testerzy czerpiemy z nich z kolei do pisania scenariuszy które dokumentują procesy przecinające cały system (testy integracyjne). Jest to szeroki temat, chciałem zaznaczyć tylko, że używanie narzędzi BDD bez typowego BDD nie jest skazane na porażkę.

Często się też zdarza, że za pomocą scenariuszy chcemy udokumentować już to, co mamy, chcemy spłacić dług techniczny i napisać specyfikację do tego, co już działa. W projekcie, w którym pracuję, programiści zaczęli zapisywać testy przy użyciu Gherkina. Któregoś dnia usłyszałem za biurka: „Aaaa, to tak działa ZagadkowaMetodaTransformacjiGąsienicyWmotyla”. Był to dla mnie sygnał, że scenariusze BDD nie są u nas narzutem, są potrzebne i dostarczają wartość.

Tak naprawdę BDD bez BDD niekoniecznie musi oznaczać problem.

Na koniec

Ideologiczne używanie BDD na pewno jest wymagające i ciężkie. Pomimo sceptycyzmu ludzi, którzy narzekają na BDD, uważam, że zawsze warto spróbować i zaadoptować techniki i narzędzia na swoje potrzeby. Jeżeli będziemy rozumieć BDD jako narzędzie do komunikacji jest szansa, że wysiłek włożony w jego wprowadzenie się zwróci z nawiązką, że będziemy robić właściwe oprogramowanie - to które naprawdę jest potrzebne naszemu klientowi. Szczególnie BDD sprawdzi się w projektach Lean startup gdzie najszybciej trzeba poznawać potrzeby klienta, odkrywać i rozumieć biznes oraz weryfikować na bieżąco hipotezy dotyczące potrzeb.

REFERENCJE

[1] http://agilecoach.typepad.com/agile-coaching/2012/03/bdd-in-a-nutshell.html
[2] http://dannorth.net/2012/05/31/bdd-is-like-tdd-if/
[3] http://lizkeogh.com/2012/05/30/showcasing-the-language-of-bdd/
[4] http://itsadeliverything.com/who-is-specification-by-example-for-everybody#more-5679
[5] http://prezi.com/zrldzwlzwxw7/bdd/

AUTOR

Daniel Dec

Pracuje w projektach IT od 7 lat biorąc udział w kilku projektach. Doświadczony tester, inżynier jakości, młodszy programista. Posiada doświadczenie w całościowym zapewnianiu jakości projektów informatycznych, procesie testowania, automatyzacji testów, testach wydajnościowych. Obecnie odnajduje się również w roli konsultanta. Miłośnik paradygmatów XP, oraz pragmatycznego podejścia do testowania.

Pomysłodawca i organizator polskiej konferencji Quality Excites (http://qualityexcites.pl/) oraz spotkań cyklicznych Quality Meetup (http://www.meetup.com/Quality-Meetup/). Współtwórca programu nauczania specjalizacji Uniwersytetu Śląskiego "Inżynieria jakości oprogramowania" (http://zaiio.ii.us.edu.pl/konferencje/wywiad-z-inzynierami-jakosci-oprogramowania-na-temat-nowych-specjalizacji/)

Po godzinach piwowar i etatowy perkusista.

Marta Firlej

Podstawy podstaw, czyli raport defektu

Prelegent na TESTWAREZ

Wprowadzenie

Ilu testerów, tyle teorii na temat formy zgłaszanych defektów. Każda firma, a nawet projekt rządzi się swoimi zasadami. Tworząc ten artykuł moim głównym celem było zebranie informacji w jaki sposób sama zgłaszam defekty. Chcę podkreślić, że jest to tylko moja subiektywna ocena na podstawie projektów i ludzi, z którymi miałam okazję pracować.

Mam nadzieję, że poniższy artykuł zmotywuje wszystkich do dyskusji na tak banalny temat jak zgłoszenie defektu, który jest częścią naszej codziennej pracy, a któremu, mam wrażenie, nie poświęcamy należytej uwagi.

Jakimi zasadami się kierować?

Istnieje kilka zasad ogólnych dotyczących podejścia do zgłaszania defektów, od których moim zdaniem powinno się zacząć:

- Sam fakt znalezienia defektu, nawet krytycznego, nie ma znaczenia, dopóki nie powstanie raport defektu.
- Zgłoszenie błędu ma służyć jego naprawieniu, czyli tworząc raport musimy mieć na uwadze jego poprawne zrozumienie, odtworzenie, naprawienie oraz późniejsze sprawdzenie czy błąd nadal nie występuje.
- Postarajmy się być konkretni. Nazywajmy poprawnie pola, przyciski, funkcjonalności itp.
- Podczas pisania raportu defektu dobą praktyką jest używanie bezokoliczników oraz o ile to możliwe pisanie tekstu

w 3 osobie.

- W przypadku raportów z błędów więcej znaczy lepiej. Jeżeli nie jesteśmy pewni jak dokładnie powinien być opisany defekt postarajmy się dostarczyć możliwie dużo szczegółów. Być może nie wszystkie informacje zostaną użyte, ale mogą one pomóc zrozumieć na czym polega nieprawidłowość.
- Nie oceniajmy pracy innych. Defekty opisujmy obiektywnie, jako działania i obiekty lub ich brak. Przy tworzeniu opisu postarajmy się możliwie najwierniej oddawać fakty z działania aplikacji, a własnymi emocjami i zdaniem na jej temat możemy podzielić się na spotkaniu integracyjnym bądź też wykrzyczeć w pokoju.
- Jeżeli jest to możliwe postarajmy się zreprodukować defekt przed utworzeniem raportu. Nie służy to udowadnianiu swoich umiejętności, a jedynie upewnieniu się, że nie jest to specyficzne zachowanie aplikacji w unikalnych warunkach. Pozwoli nam to dodatkowo na dodanie informacji o powtarzalności występowania defektu.
- Na koniec coś, z czym mamy największy problem podczas intensywnej pracy. Postarajmy się przeczytać swój raport przed wysłaniem do programisty. Pozwoli to na wstępną weryfikację spójności, prawdziwości, a także jakości zgłoszenia.

Kilka słów o najbardziej kłopotliwych pozycjach w opisie defektu:

Tytuł/podsumowanie (Title) - Tytuł powinien być unikalny.

Mam taki sposób na podsumowanie: zawsze próbuję odpowiedzieć na trzy pytania: gdzie?, co? i jak?

Podana kolejność pozwala na łatwe wyszukiwanie i uzyskiwanie informacji, czego dotyczy defekt. Przykładowe odpowiedzi na pytania to:

- gdzie? - projekt/moduł/pole - może zawierać całą ścieżkę dojścia do danego miejsca
- co? - na przykład przycisk, funkcja, itd.
- jak? - na przykład wyświetla się błąd, jaka funkcjonalność nie jest zablokowana itp.

Opis (Description) - powinien zawierać zwięzłe informacje o:

- danych wejściowych
- danych wyjściowych
- zachowaniu oprogramowania

Jeżeli jest to możliwe warto dodać informację o możliwej przyczynie powstania defektu.

Opis najlepiej wykonywać pod postacią:

- kroków potrzebnych do reprodukcji (Steps to reproduce) - opis krok po kroku jak doprowadzić do wystąpienia defektu, kolejne czynności powinny być ponumerowane
- otrzymanym rezultacie (Actual Result) - wynik wykonanego testu, który nie jest prawidłowy
- oczekiwanym rezultacie (Expected result) - oczekiwany wynik testu

Status (Status) - im mniej statusów w danym procesie testowym, tym lepiej. Mają one określać dokładny stan, w jakim znajduje sie defekt. Nazwy statusów zależne są od programu używanego do zarządzania defektami.

CO MUSI ZAWIERAĆRAPORT DEFEKTU

Nazwa pola	Opis
Numer/identyfikator (Number)	Unikalny ciąg znaków. Możemy za jego pomocą identyfikować defekt.
Tytuł/podsumowanie (Title/ Subject/Summary)	Jasno sformułowany Jednoznacznie identyfikuje miejsce oraz rodzaj defektu.
Priorytet (Priority)	Określenie pilności naprawienia defektu.
Status (Status)	Jednoznaczne określenie stanu, w jakim znajduje się defekt.
Opis (Description)	Dokładny opis defektu. Opis powinien być możliwie najprostszy i zrozumiały dla każdego. Powinien zawierać: • Dane wejściowe • Kroki do reprodukcji • Otrzymany rezultat • Oczekiwany rezultat
Załączniki (Attachments)	Wszystkie informacje, które mogą pomóc w zlokalizowaniu i zrozumieniu zaraportowanego defektu, m.in. zrzuty ekranu, logi, nagrania wideo, maile, wyciągi z dokumentacji, itp.
Autor defektu (Author)	Osoba, która zgłosiła występowanie defektu i jest w stanie dostarczyć więcej szczegółów.
Osoba przypisana do defektu (Assigned to)	W zależności od procesu testowego może to być na przykład koordynator defektów, deweloper odpowiedzialny za jego rozwiązanie, a także tester przeprowadzający retesty.

CO MOŻE DODATKOWO ZAWIERAĆ RAPORT DEFEKTU?

Nazwa pola	Opis
Projekt (Project)	Nazwa projektu
Wdrożona wersja produktu (Build version)	Wersja, na której wykonane zostały testy oprogramowania, które doprowadziły do wykrycia defektu.
Moduł (Modul)	Określony moduł produktu, gdzie został wykryty defekt.
Środowisko (Environment)	W zależności od oprogramowania wersja przeglądarki, systemu operacyjnego, parametry procesora, itp.
Ważność (Severity)	Wpływ defektu na działanie produktu.
Wersja, w której został naprawiony defekt (Fixed build version)	Wersja produktu, w której został naprawiony defekt.

Najczęściej stosuje się statusy:

- otwarty (Open, Re-open)
- w trakcie naprawiania (In progress)
- naprawiony (Resolved)
- w trakcie testowania(In testing)
- zamknięty (Closed)
- duplikat (Duplicated)

Priorytet (Priority) - określa jak ważne jest szybkie poprawienie defektu. Przy nadawaniu priorytetu tester dokonuje subiektywnej decyzji jak ważne jest szybkie poprawienie kodu lub grafiki. Podejmując decyzje powinno się brać pod uwagę różne czynniki, takie jak: dostępny czas, zasoby, prawdopodobieństwo wykrycia defektu przez użytkownika oraz potrzeby biznesowej.

Najczęściej stosowane priorytety to:

- Bloker/krytyczny (Blocker/Critical) - musi zostać naprawiony podczas najbliższego wdrożenia, bądź wdrożenie zostanie zainicjowane przez potrzebę wgrania poprawionego oprogramowania bądź elementów grafiki
- Wysoki (High) - może zostać naprawiony podczas najbliższego wdrożenia, jednak musi zostać naprawiony przed wypuszczeniem nowej wersji produktu
- Średni (Medium) - może zostać naprawiony w kolejnym wypuszczeniu nowej wersji produktu
- Niski (Low) - może, ale nie musi być rozwiązany

Ważność (Severity) - wpływ defektu na działanie funkcjonalności, modułu lub całego systemu.

- Bloker/krytyczny (Blocker/Critical) – Defekt dotyczy kluczowej funkcjonalności oprogramowania lub krytycznych danych. Nie ma możliwego obejścia. Na przykład: nie można zalogować się do aplikacji podczas testowania poczty internetowej.
- Ważny (Major) – Defekt mniej ważnej funkcjonalności oprogramowania. Istnieją możliwe obejścia jednak nie są

one oczywiste, lub proste do wykrycia. Na przykład: po zakończeniu procesu rezerwacji nie jest wysyłany mail z potwierdzeniem, mimo że powinien, jednak potwierdzenie jest możliwe do pobrania w systemie.

- Lekki (Minor) – Defekt dotyczy mniej ważnych funkcji w oprogramowaniu, bądź danych nie krytycznych. Na przykład: pole tekstowe do wprowadzania numeru pesel umożliwia wprowadzenie adresu email.
- Trywialny (Trivial) – Defekt nie wpływa na całość funkcjonalności czy dane. Na przykład: literówka w mailu potwierdzającym rezerwację.

Załączniki (Attachments) - Każdy załącznik, który będzie wskazywał gdzie lub w jaki sposób został wykryty defekt będzie idealnym rozszerzeniem zgłoszenia. Im więcej dołączonych informacji w tym przypadku tym lepiej. W szczególności warto dodwać zrzuty ekranu z oznaczeniem co jest nieprawidłowe, logi, nagrania czy wyciągi z dokumentacji.

Przykład

Wierzę, że najłatwiej pracuje się na przykładach, dlatego załóżmy, że:

- Jako pierwsi testujemy formularz rejestracyjny zbierający pewne dane.
- Podczas testowania zauważyliśmy nieprawidłowe zachowanie przycisku "Akceptuję".
- Zamiast otrzymać informację o potwierdzeniu zaakceptowania formularza, zawartość wszystkich pól jest usuwana. Podane dane są poprawnie zwalidowane. Dane nie są zapisywane w bazie danych.

PRZYKŁAD RAPORTU DEFEKTU

Nazwa pola	**Opis**
Numer/identyfikator (Number)	0001
Tytuł/podsumowanie (Title)	Formularz rejestracyjny - przycisk "Akceptuję" - dane z formularza są usuwane zamiast zapisania do bazy danych
Priorytet (Priority)	Bloker (Blocker)
Status (Status)	Otwarty (Open)
Opis (Description)	1. Otworz formularz rejestracyjny : www.test.pl 2. Wypełnij formularz prawidłowymi danymi : • imie: test • nazwisko: test • mail: test@test.pl • telefon: 123456789 • kliknij w przycisk "Akceptuję" 3. Otrzymany rezultat: • dane w formularzu zostają wykasowane • dane nie są zapisywane w bazie danych 4. Oczekiwany rezultat: • otrzymujemy informację potwierdzającą zaakceptowanie formularza. • dane zostają zapisane do bazy danych
Załączniki (Attachments)	Zrzut ekranu po wpisaniu wszystkich danych.Z rzut ekranu z efektem po wcisnięciu przycisku "Akceptuje".
Autor defektu (Author)	Marta Firlej
Osoba przypisana do defektu (Assigned to)	Jan Kowalski
Data utworzenia (Date)	01.04.2014

AUTOR

Marta Firlej

Quality Assurance inżynier w zespole Agile'owym. Zaangażowana w aspekty związane z zapewnianiem jakości od 2006 roku. W trakcie pracy skupiona na produkcie i możliwości wsparcia każdego członka zespołu oraz rozwoju zarówno osobistym jak i projektowym.

Zawodowo zaangażowana w tworzenie dobrej atmosfery w zespole oraz zapewnienie jakości ze szczególnym uwzględnieniem testów użyteczności i narzędzi wspomagających całokształt procesu testowego.

Prywatnie fan podróżowania, jazdy na rowerze i rozwoju osobistego.

- Znamy wymagania i wiemy, że jest to defekt
- Zebraliśmy też wszelkie potrzebne informacje.

Moim zdaniem raport defektu powinien wyglądać jak na Rys. 1.

Najczęstsze błędy podczas zgłaszania defektu:

- Tytuł, który nie sugeruje gdzie znaleziono defekt, czego dotyczy i na czym polega.
- Błędy opisane w jednym zdaniu, bez informacji jakie powinno być poprawne działanie, gdzie i w jaki sposób został znaleziony defekt.

Tutaj wspomnieć należy o raporcie z defektu, którego tytuł oraz opis jest zdefiniowany przez "nic nie działa". Tego typu defekt nie zostanie rozwiązany bez dodatkowych informacji.

Używanie słownictwa, bądź skrótów słownych stosowanych w projekcie, przez co defekt jest trudny do zinterpretowania przez nowych członków zespołu lub osób spoza projektu.

Podsumowanie

Wiem, że czytelnicy mogą mieć zupełnie inne zdanie na temat "idealnego" zgłoszenia defektu. Chciałabym jednak, żeby był to przewodnik lub swoisty impuls do zastanowienia się, co możemy zrobić, żeby poprzez tak małe ulepszenie, jaką jest zmiana sposobu raportowania defektu, ułatwić oraz przyspieszyć pracę osobom związanym z ich naprawianiem.

Rafał Stańczak

Praktyki i wspierające je narzędzia w projektach zwinnych

Wprowadzenie

Powiedzenie dzisiaj o organizacji, że jest zwinna (ang. agile) nie jest trudne, ale bycie „naprawdę agile" wymaga z pewnością sporego wysiłku. Wiele organizacji nadal transformuje siebie i swoje procesy w stronę metodyk zwinnych, ostatnio coraz mocniej na znaczeniu zyskuje również eliminowanie strat z koncepcją Lean na czele. Aby nie być gołosłownym, chciałbym przytoczyć ciekawe dane, które usłyszałem podczas zeszłorocznej konferencji Agile Management. Patrząc na rynek europejski, najwięcej projektów zwinnych ok. 80% ogółu robi się na rynku brytyjskim, na drugim biegunie z ok. 25% znajduje się rynek niemiecki. A jak myślicie gdzie w tym zestawieniu znajdujemy się my, otóż Polska plasuje się z wynikiem ok. 50%. Czy to dużo czy mało??? Sami sobie spróbujmy odpowiedzieć na to pytanie.

Chciałbym zainteresować Was kwestią doboru praktyk i narzędzi w projektach zwinnych, aby praca zespołu była wydajna a zarazem dostarczała na co dzień tzw. 'funu'. Należy mieć na uwadze, iż wiele decyzji dotyczących wyboru narzędzi trzeba podjąć we wstępnej fazie trwania projektu, zanim zaczniemy pisać kod, ponieważ determinują one późniejsze zależności lub zwyczajnie na dalszych etapach może okazać się, że na zmiany jest już za późno. Oczywiście chciałbym przy tym zaznaczyć, że nie ma jednego magicznego zestawu narzędzi, który będzie idealnie pasował do wszystkich rodzajów projektów, jednakże postaram się, aby każdy z Was mógł znaleźć tu coś wartościowego dla siebie.

Po pierwsze: Repozytorium Kodu Źródłowego

Narzędzie do kontroli wersji kodu oraz synchronizowania zmian poszczególnych członków zespołu nie jest oczywiście czymś unikalnym w projektach zwinnych, jest tak samo ważne we wszystkich metodykach, ale piszę o nim, ponieważ bez niego zespół nie będzie w stanie normalnie pracować. Mówiąc o zespole mam na myśli developerów implementujących nowe funkcjonalności, testerów badających konkretne zmiany w nowych rewizjach kodu czy osobę odpowiedzialną za przekazanie wersji do klienta, ewentualnie wdrożenie na produkcję. Co warto zaznaczyć, narzędzie niekoniecznie musi służyć wyłącznie do wersjonowania kodu. Z praktyki wiem, iż można wersjonować inne rzeczy takie jak: skrypty testowe, dokumentację, testy automatycznie, itd. W momencie, kiedy decydujemy się na wersjonowanie dodatkowych rzeczy zależnych od wersji kodu, musimy pamiętać, aby je w logiczny sposób skorelować z kodem. Zazwyczaj robi się to poprzez nadawanie tagów. Nadajemy ten sam tag odpowiedniej wersji kodu oraz korespondującym z nią plikom testów automatycznym, dokumentacji, itd.

Najpopularniejszymi narzędzi tego typu są:

- Subversion (SVN) – darmowy, prosty w instalacji system kontroli wersji, zbudowany na podstawie systemu CVS. Dodatkowo łatwo integruje się ze zintegrowanymi środowiskami programistycznymi, wspiera Continous Integration (o tym w dalszej części) oraz ma wiele innych zastosowań, które znajdziecie na stronie produktu.
- Git – darmowy, rozproszony system kontroli wersji. Początkowo zbudowany przez Linusa Torvaldsa do wsparcia rozwoju Linuxa. Aktualnie wspiera wiele systemów operacyjnych.
- Bazaar – darmowy, łatwy w obsłudze system kontroli wersji, zbudowany całkowicie od nowa z uwzględnieniem pewnych ograniczeń SVN.

Po drugie: Zintegrowane środowisko developerskie (ang. IDE)

Wybór środowiska developerskiego często narzuca nam technologia, w której chcemy budować oprogramowanie. W projektach zwinnych jednym z wymagań, o których należy pamiętać jest to, by środowisko developerskie integrowało się z narzędziem do kontroli wersji. Ma to przeciwdziałać problemom wersjonowania oraz integracji kodu tworzonego przez poszczególnych członków zespołu. W większości organizacji wybór IDE jest z góry określany przez technologie. Dobrą praktyką jest utrzymywać jeden rodzaj platformy w całym zespole (w projektach Java może to być np. Eclipse). Wówczas mamy możliwość lepszej współpracy, dzielenia się wiedzą, łatwiej też przyjdzie nam sprawdzenie technik programowania w parach czy przeglądów kodu, jeśli się na nie zdecydujemy. Wiele z takich platform ma rozbudowany system dodawania pluginów (dodatków), które rozszerzają środowisko developerskie funkcjonalnie i technicznie. Równie istotne, co pisanie nowego kodu jest przeglądanie i refactoring starego (do dyspozycji mamy statyczne analizatory kodu, które bez kompilacji zwrócą listę potencjalnych zagrożeń i błędów w kodzie).

Najpopularniejsze środowiska developerskie to:

- Eclipse i NetBeans używane w projektach wykorzystujących technologie i język programowania Java.
- Visual Studio używanie w projektach wykorzystujących technologie Microsoft.
- IntelliJIDEA - komercyjne zintegrowane środowisko programistyczne dla Javy, Androida, Scala i wielu innych języków.

Po trzecie: Narzędzie do tworzenia buildu

W projektach zwinnych każdy z zespołów musi mieć możliwość częstego i szybkiego weryfikowania jakości swojej pracy. Bardzo istotne jest to, aby zespół wiedział, jak szybko przetworzyć dostarczony kod w wykonywalny plik czy gotową do testów aplikację. W zespołach, w których pracowałem bądź aktualnie pracuję, wykorzystuje się do tego narzędzia ANT lub Maven w celu zbudowania wykonywanej wersji projektu. Te narzędzia łatwo współdziałają ze zintegrowanymi środowiskami developerskimi oraz narzędziami do testów automatycznych. Idąc krok dalej docieramy do narzędzi, które pozwalają zautomatyzować proces tworzenia buildów. Continuous Integration (CI), bo o nim mowa, jest jedną z kluczowych praktyk zespołów zwinnych. Z koncepcją CI wiąże się potrzeba szybkiego feedbacku w reakcji na wprowadzoną zmianę. W kilku słowach, polega to na zestawieniu automatycznego procesu budowania produktu, puszczeniu na nim automatycznych testów potwierdzających jakość zastosowanych rozwiązań i zwróceniu wyników w postaci raportu. W pełni zautomatyzowany proces jest kluczem do sukcesu, dlatego dobierając narzędzia ważne jest, aby wszystkie wcześniej opisane komponenty współpracowały z wybranym rozwiązaniem CI. Przykładem darmowych narzędzi tego typu jest Hudson/Jenkins. Sprawnie działający proces CI jest kluczowy dla powodzenia projektu, wobec czego rekomenduję go opracować i ewentualnie zestawić przed rozpoczęciem developmentu.

Definicja z Wikipedii:
Ciągła integracja (ang. Continuous Integration) - praktyka stosowana w trakcie rozwoju oprogramowania, polegająca na częstym, regularnym włączaniu (integracji) bieżących zmian w kodzie do głównego repozytorium. W praktyce, każdy członek zespołu programistycznego powinien przynajmniej raz dziennie umieścić wykonaną przez siebie pracę w repozytorium. Niezbędnym elementem jest także zapewnienie poprawności kompilacji kodu po wykonaniu integracji.

Po czwarte i ostatnie: Testy jednostkowe

W przypadku tego typu praktyki istotne jest to, aby podjąć decyzję na starcie i zacząć tworzyć testy jednostkowe od samego początku trwania developmentu. Podejmując decyzję później, w trakcie realizacji kolejnych iteracji, narażamy się na niepowodzenie i frustrację zespołu, który będzie musiał przejrzeć i pokryć testami napisany już wcześniej kod. Testy jednostkowe wymagają pewnego nakładu pracy,

AUTOR

Rafał Stańczak

Konsultant i coach Quality Assurance i Scrum z bogatym doświadczeniem w kilku dużych międzynarodowych projektach z rożnych branż (telekomunikacja, ubezpieczenia, e-commerce) Certyfikowany Scrum Master, założyciel i autor bloga www.scrumdo.pl w ramach którego dzieli się posiadaną wiedzą ze społecznością polskiego IT.

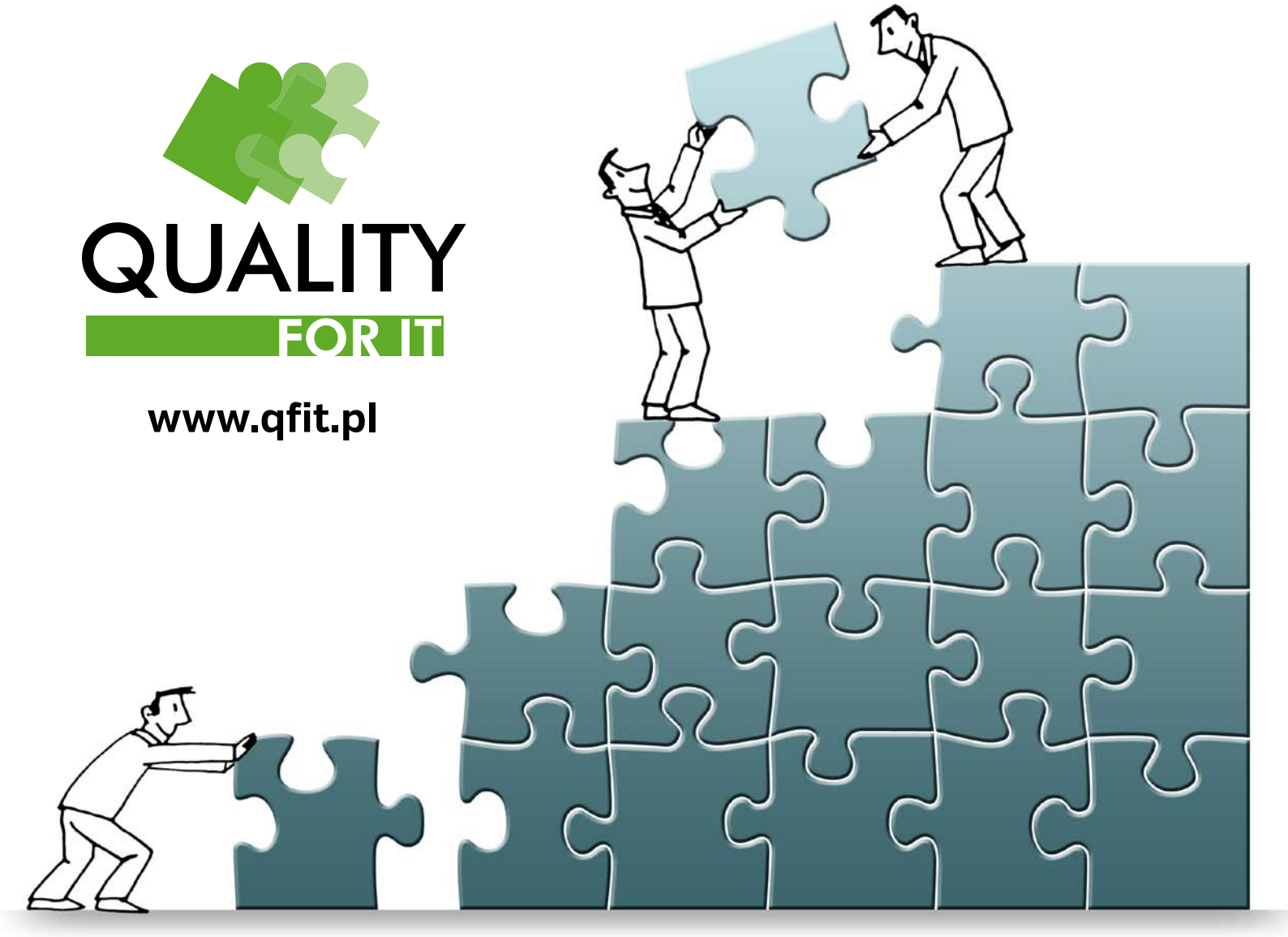
Aktualnie Scrum Master w dwóch zespołach projektowych oraz Koordynator Quality Assurance portfela produktów w szwajcarskiej firmie z branży bankowej. Biorąc udział w projektach zawsze koncentruje się na dostarczaniu wysokiej wartości klientom, usuwa przeszkody stojące przed zespołami w drodze do realizacji celu oraz dba o finalną jakość produktu. Członek organizacji Scrum Alliance i Stowarzyszenia Jakości Systemów Informatycznych. Pasjonat i praktyk lekkich metodyk zarzadzania projektami Agile i Lean.

jednakże pamiętajmy, że z jednej strony dostarczają testowalny kod a z drugiej budują podwaliny pod testy na kolejnych etapach tworzenia oprogramowania. Nie bez znaczenia jest fakt, iż błędy znalezione na tym etapie nie są powielane w późniejszych fazach projektu, przez co kosztują zespół i projekt relatywnie najmniej. Narzędzia do pisania testów jednostkowych zależą od języka programowania, w których tworzymy produkt. Najpopularniejsze to Junit dla Javy i Nunit dla .Net. Również kod reprezentujący GUI powinien być testowany wcześniej, na przykład przy pomocy TestNG lub SWTBot.

Podsumowując, narzędzia wybierajmy świadomie, dobrze wiedząc, czego od nich oczekujemy. Wybór odpowiednich rozwiązań typu Continuous Integration, procesu tworzenia buildów i automatyzacji testów ma wspierać zespół przez dostarczanie informacji zwrotnej tak szybko, jak to możliwe. Zespoły, które nie podejmą przemyślanych decyzji we właściwym momencie, mogą mieć trudności, co może prowadzić do frustracji, złej atmosfery pracy bądź w skrajnych przypadkach przyczynić się do niepowodzenia projektu.

QUALITY FOR IT

www.qfit.pl



KONFERENCJA

QUALITY FOR IT

✓ CELE BIZNESOWE – WSPARTE!

✓ WARTOŚĆ DLA KLIENTA – DOSTARCZONA!

✓ CZAS I BUDŻET – DOTRZYMANE!

5-6 marca 2015, hotel Warszawianka, Jachranka k. Warszawy

Organizatorzy

EVENTION

SJSI STOWARZYSZENIE JAKOŚCI SYSTEMÓW INFORMATYCZNYCH

Wirtualizacja Przewaga konkurencyjna to garść wizytówek do zaufanych partnerów

Polskie firmy niechętnie podejmują współpracę biznesową, ich gotowość do zawierania partnerstw jest niewielka a stopień wykorzystywania nowoczesnych technologii ICT wyraźnie mniejszy niż w krajach rozwiniętych – wynika z badań ekonomistów. Tymczasem praktyka biznesowa pokazuje, że tylko łączenie umiejętności, doświadczeń i możliwości przy wykorzystaniu nowoczesnych technologii umożliwia dziś szybkie i elastyczne reagowanie na zmiany rynkowe i rosnące wymagania klientów. Sukces coraz częściej zależy nie od produktów czy usług, lecz od modelu biznesowego.

klaster.it

Zmienny rynek – zwinny biznes

Ciągła zmienność i nieprzewidywalność rynku powoduje, że firmy coraz częściej nie są w stanie samodzielnie zaspokajać rosnących wymagań klientów - *„Dotyczy to szczególnie małych i średnich przedsiębiorstw, które szybko dostrzegają okazje rynkowe, ale często nie są w stanie ich wykorzystywać, ponieważ brakuje im kompetencji, technologii czy finansów"* – wskazuje dr Karol Kuczera z Wydziału Nauk Ekonomicznych i Zarządzania Uniwersytetu Szczecińskiego.

Jednak ich możliwości biznesowe znacząco się zwiększają, kiedy zaczynają działać w grupie, w ramach sieci zaufanych partnerów. W ten sposób łączą różne kompetencje i zasoby, co pozwala im zwinnie dostosowywać się do aktualnych potrzeb rynku.

„Współpracę ułatwiają nowoczesne technologie ICT, które pozwalają na przezwyciężanie ograniczeń przestrzennych i czasowych. Partnerzy nie muszą codziennie spotykać się w biurze, mogą przebywać w różnych częściach świata i współpracować zdalnie" - wyjaśnia dr Kuczera. Firmy działające w ramach takiego płynnego i elastycznego modelu biznesowego tworzą wspólnie tzw. organizacje wirtualne.

Wirtualny znaczy konkurencyjny

Według ekonomistów polskie firmy niestety nie postrzegają współpracy jako sposobu na zwiększanie swojej konkurencyjności, tymczasem to właśnie wirtualny model biznesowy przynosi rynkową przewagę – *„Dziś firmy nie konkurują produktami i usługami, tylko modelami biznesu. Największą wartością biznesmena stała się garść wizytówek do zaufanych partnerów, z którymi współpracuje, kiedy nadarza się okazja rynkowa. Korzystanie z możliwości produkcyjnych, wiedzy lub kompetencji partnerów jest metodą przyspieszania wzrostu konkurencyjności"* – dodaje dr Karol Kuczera. Wirtualizacja dotyczy wszystkich branż, a jej stopień i zakres zależą od indywidualnych potrzeb firm i konkretnej sytuacji rynkowej.

Wirtualizacja w praktyce

W dobie dynamicznego rozwoju nowoczesnych technologii model biznesowy oparty na wirtualizacji jest coraz częściej wdrażany przez firmy informatyczne. *„Naszym głównym produktem jest system Emigo, wspierający procesy sprzedażowe w firmach zatrudniających mobilnych handlowców"* – mówi Marcin Niczyporuk, Dyrektor Działu IT w Sagra Technology. Do niedawna firma opracowywała samodzielnie wszystkie elementy oprogramowania, ale przy rozwijaniu kolejnych funkcjonalności zdecydowała się wprowadzić nowy model biznesowy. Dlaczego? W zwinnym biznesie bardzo ważna jest szybkość wykonywania projektów, a zgłębianie nowej wiedzy wymaga czasu. *„Technologie zmieniają się bardzo szybko i są coraz bardziej złożone, a w branży IT nie sposób specjalizować się we wszystkim. Dlatego nasi programiści pracują nad technologią, nad elastycznością systemu, umożliwiającą szybkie reagowanie na zmiany rynkowe – to jest wiedza, którą zamykamy w naszym zespole. Natomiast do innych części projektów pozyskujemy partnerów, którzy są specjalistami w swoich dziedzinach, np.*

grafików czy ekspertów tworzących technologie geolokalizacyjne, które chcemy wprowadzić do naszego oprogramowania" – wyjaśnia Marcin Niczyporuk. Taka współpraca pozwala ograniczać koszty związane z zatrudnianiem nowych pracowników i ich przystosowywaniem do pracy. W zależności od potrzeb może zakończyć się na jednym projekcie bądź trwać tak długo, jak długo przynosi korzyści obu stronom.

Dlaczego nie outsourcing?

Istotą wirtualizacji jest partnerstwo. Współpraca w takim modelu biznesowym nie polega jedynie na zlecaniu, nadzorowaniu i rozliczaniu zadania, jak to ma miejsce w przypadku outsourcingu, ale na budowaniu trwałych powiązań biznesowych. Relacji opartych na zaufaniu, które – w przeciwieństwie do produktów czy usług – są trudne do naśladowania. *„Trzeba szukać partnerów, z którymi warto podjąć dłuższą współpracę. Kiedy pracujemy nad projektem z naszymi partnerami, staramy się stworzyć zgrany zespół. Jesteśmy otwarci na ich sugestie, bo jako eksperci mogą wnieść do projektu cenną specjalistyczną wiedzę"* – przekonuje Marcin Niczyporuk z Sagra Technology. Tradycyjny outsourcing przestaje się powoli sprawdzać w firmach tworzących technologie również z powodu ryzyka, jakie niesie za sobą ujawnianie podwykonawcom swojego know-how. *„Całkowite outsourcowanie technologii to tak naprawdę wyjawianie zewnętrznym podmiotom najważniejszej wiedzy, którą się konkuruje na rynku. To duże ryzyko. Lepiej sprawdza się współpraca, w ramach której każdy z partnerów zajmuje się tym, w czym się specjalizuje"* – dodaje Niczyporuk.

Klaster jako organizacja wirtualna

Trwałe biznesowe relacje, których efektem jest współpraca sieciowa, są podstawą funkcjonowania klastrów. W takich organizacjach nie ma hierarchii, dzielącej członków na ważnych i mniej ważnych, nie ma podziału na zleceniodawców i zleceniobiorców, współpraca firm opiera się na partnerstwie – *„Przedsiębiorcy dołączają do klastra dobrowolnie, bo widzą w tym korzyści dla siebie. Są skłonni zaoferować pewne swoje zasoby i kompetencje, ponieważ w zamian mają dostęp do zasobów i kompetencji innych. Liczne spotkania i realizacja wspólnych projektów umożliwiają przedsiębiorcom poznanie się nawzajem, co z kolei buduje zaufanie"* – zaznacza Katarzyna Witkowska, Prezes Klastra.IT. Stowarzyszenie zrzesza ponad 70 firm informatycznych, reprezentujących wszystkie sektory rynku IT – od tworzenia oprogramowania, gier oraz aplikacji mobilnych i internetowych, przez zarządzanie projektami, świadczenie usług i doradztwo IT, po produkcję i obsługę sprzętu informatycznego. Tak szeroka gama zasobów i kompetencji pozwala na budowanie sieci współpracy, która przynosi korzyści każdemu z partnerów.

Klaster ICT Pomorze Zachodnie

Stowarzyszenie Klaster ICT Pomorze Zachodnie ułatwia firmom zdobycie wiedzy i kontaktów biznesowych. Jako najaktywniejsza organizacja IT w województwie działamy w celu rozwoju gospodarczego regionu poprzez rozwój firm sektora IT oraz zwiększenia atrakcyjności Pomorza Zachodniego i miasta Szczecina dla pracowników, inwestorów oraz przedsiębiorców lokalnych oraz studentów.

Więcej na: www.klaster.it

Społeczna odpowiedzialność sprzyja rozwojowi biznesu

Parlament Europejski uchwalił dyrektywę, nakładającą obowiązek raportowania danych pozafinansowych na największe firmy w Unii Europejskiej. Informowanie o działaniach dotyczących m.in. zarządzania firmą, polityki pracowniczej, przestrzegania praw człowieka czy ochrony środowiska ma wzmocnić społeczną odpowiedzialność i przejrzystość dużych przedsiębiorstw. Dyrektywa będzie teraz wdrażana do systemów prawa krajowego przez państwa wspólnoty.

klaster.it

Społeczna odpowiedzialność przedsiębiorstw (ang. corporate social responsibility – CSR) to podejmowanie przez firmy działania na rzecz społeczeństwa i środowiska. Przyjęta przez Parlament Europejski dyrektywa dotyczy raportowania CSR w takich obszarach działania dużych firm jak: kwestie środowiskowe, społeczne, związane z zatrudnieniem i prawami człowieka oraz standardami przeciwdziałania korupcji, założenia biznesowe, wyniki i czynniki ryzyka oraz normy odnoszące się do szeroko rozumianego zarządzania firmą.

„Termin na wdrożenie nowego prawa w państwach członkowskich UE to 2 lata. Jednak raportowanie CSR już od jakiegoś czasu staje się w Polsce dobrą praktyką i to nie tylko wśród największych przedsiębiorstw, ale również wśród firm małych i średnich" – mówi Marzena Zasińska-Igła z Klastra.IT

Firmy w Polsce podejmują różnorodne działania z zakresu CSR, gdyż wiele z nich wpływa na rozwój przedsiębiorstwa – *„Firma Orange realizuje program „Bezpieczny w sieci", w ramach którego uświadamia rodziców, jak dzieci powinny bezpiecznie korzystać z Internetu. Efektem jest nie tylko większe bezpieczeństwo klientów i edukacja rynku, ale także docieranie do nowych konsumentów"* – wskazuje Robert Król z Fundacji Netcamp, wspierającej rozwój branży internetowej.

Inicjatywy prospołeczne sprzyjają także budowaniu pozytywnego wizerunku marki - *„Przynosi to korzyść nie tylko społeczeństwu, ale pozwala także budować wizerunek firmy zaangażowanej społecznie"* – dodaje Król. Przykładem są szczecińscy informatycy z firmy Squiz, którzy organizują cykliczne rajdy rowerowe, mające na celu zebranie pieniędzy na rehabilitację chorych dzieci. Z kolei dostawca Biuletynu Informacji Publicznej nbip.pl współtworzył i wspiera fundację, która prowadzi ogólnopolską akcję „Bezpieczne Wakacje", promującą bezpieczne zachowania podczas letniego wypoczynku – *„We współpracy ze szkołami, przedszkolami, ośrodkami kultury, bibliotekami itp. wspieramy organizację pogadanek z udziałem strażaków i policjantów oraz jeden z największych*

konkursów plastycznych dotyczący bezpieczeństwa, prowadzimy również ogólnopolskie kampanie profilaktyczne. W zeszłym roku w Szczecinie wspieraliśmy pilotażowy Bezpieczny Dzień w szkołach przy wykorzystaniu najnowocześniejszych technologii IT, a w tym integracyjne wyprawy RoweremDoSzwecji.pl dla ponad dwustu rowerzystów" – wyjaśnia Maciej Możejewski z firmy NETStream Poland.

Działania CSR są powszechnie utożsamiane z dużymi firmami i dużymi nakładami finansowymi, jednak podejmowane są również w codziennej działalności małych i średnich przedsiębiorstw, choć często nieświadomie nie klasyfikują ich one jako społecznej odpowiedzialności biznesu. A wśród nich wymienić można choćby dobre relacje z pracownikami – *„W naszej firmie wewnętrzne działania CSR przejawiają się we wspieraniu pasji pracowników poprzez np. opłacanie im kart sportowych, w organizowaniu spotkań, podczas których pracownicy mogą zaprezentować swoje pomysły czy w przejrzystych zasadach awansu i jasnym sposobie naliczania prowizji, aby każdy pracownik wiedział, ile i za co zarobi"* – wylicza Bartosz Kędzior z firmy HEUTHES.

Społeczna odpowiedzialność biznesu przejawia się także w relacjach z klientami i uczciwych praktykach rynkowych – *„Istotna jest przejrzystość umów zawieranych z klientami oraz uświadamianie ich w zakresie ich praw, bo wiele firm wykorzystuje tę niewiedzę i oszukuje konsumentów. Uczciwość biznesowa to również poszanowanie własności intelektualnej innych firm oraz rzetelne i terminowe wykonywanie zleceń"* – zaznacza Damian Barczyk, właściciel firmy SmartNet.

Z kolei działania na rzecz ochrony środowiska to nie tylko duże inwestycje w ekologię, ale również oszczędność prądu i wody czy choćby elektroniczne realizowanie procedur administracyjno-biurowych w firmach, dzięki czemu zmniejsza się ilość zużywanego papieru – *„To korzystanie z ekologicznych materiałów czy choćby mądre wykorzystywanie artykułów biurowych – np. na zwykłe długopisy czy teczki można nakleić logo firmy i wykorzystywać je jako materiały promocyjne"* – wyjaśnia Adam Kurkiewicz z firmy CassperSoft.

Działania z zakresu CSR często podejmowane są również we współpracy z innymi firmami, fundacjami czy instytucjami otoczenia biznesu w ramach wspólnie realizowanych projektów – *„Firmy z Klastra. IT angażują się w projekty mające na celu np. tworzenie aplikacji ułatwiających życie mieszkańców Szczecina dzięki wykorzystaniu danych publicznych czy - we współpracy z Technoparkiem Pomerania - w promowanie obecności kobiet w branży informatycznej i unowocześnianie kształcenia informatycznego w szkołach średnich i na uczelniach"* – wskazuje Marzena Zasińska-Igła.

Choć nowa dyrektywa Parlamentu Europejskiego narzuca obowiązek raportowania działań CSR na duże przedsiębiorstwa, to informowanie społeczeństwa o takich praktykach może przynieść korzyść również małym i średnim firmom. Pozwoli im to bowiem pozyskiwać klientów, którzy w swoich wyborach kierują się etycznymi aspektami działania przedsiębiorstwa, umożliwi inwestorom przyjrzenie się postępowaniu firmy pod kątem polityki zrównoważonego rozwoju czy wpłynie na lepsze relacje pracodawców z pracownikami.

Bogdan Bereza

Pop-psychologia w IT

Prelegent na TESTWAREZ

Oni chcieli dostać gotowe rozwiązania

Oto pewnego dnia prowadzę warsztaty o nazwie „Psychologia zarządzania zespołem IT", takie same, jak te, które kilka tygodni wcześniej, z innymi przykładami i scenkami, pod nazwą „Psychologia zarządzania zespołem Agile Scrum", przedłużyłem – w piątek po południu – aż o dwie godziny, bo uczestnicy tak bardzo się zaangażowali i tak bardzo chcieli dowiedzieć się jeszcze więcej, niż było w agendzie.

Tym razem widzę od razu, że nie będzie równie dobrze. Jeden z uczestników, w garniturze, o ponurym obliczu zawodowego pokerzysty, jest z firmy toksycznej na wskroś: każdy tam każdemu podstawia nogę, podkłada prosiątko, a ulubiony argument prezesa to: „albo zrealizujesz projekt w terminie, albo się pożegnamy". Nie muszę dodawać, że dotyczy to projektu o niejasnych wymaganiach, prawda? Główny problem tego uczestnika: jak sobie poradzić z programistą w zespole scrumowym, który notorycznie nie wypełnia swoich zobowiązań, bowiem zamiast tego wykonuje różne wrzutki, zlecane mu telefonicznie przez samego prezesa czy mniejszych pod-prezesków [1]. Pozostali uczestnicy są z dużych, państwowych przedsiębiorstw i chętnie dowiedzieliby się jak najwięcej o technologiach IT służących do śledzenia pracowników, żeby uniemożliwić im okradanie firmy albo wykonywanie prywatnej fuchy w godzinach pracy.

Przewiduję, że zaplanowane w agendzie na te warsztaty, wykłady i ćwiczenia z technik komunikacji, kwadratu von Thuna, mechanizmów motywacji, asertywności, radzenia sobie ze stresem, kultury organizacji, nie zyskają aprobaty tych uczestników, oczekujących soczystych filmików z

pełnymi napięcia scenkami oraz szybkich, jednoznacznych sposobów, jak przeforsować swoje cele nawet w chorych, z gruntu złych organizacjach. Zrozumienie nie jest konieczne, wręcz przeciwnie, przyda im się natomiast fasada, ersatz zrozumienia, bo zapewni poczucie kontroli i da pewność siebie.

Diabeł, siedzący za kołnierzem u każdego szkoleniowca i kuszący, aby uczyć nie tego, czego uczestnicy naprawdę potrzebują, tylko tego, co zapewni wysokie oceny w ankietach, szepce mi do ucha: daj im NLP!

Tak, NLP jest w tej sytuacji niezastąpione. Z jednej strony oferuje wszelkie pociechy autentycznego szamanizmu: rytuały, niezachwianą pewność, tajemniczych guru i galerię opowieści o sukcesach (oraz skrzętnie przemilczane niepowodzenia), z drugiej strony znakomicie ubiera to wszystko w pseudo-naukową terminologię. Sama nazwa jest tego godnym podziwu przykładem: NLP, czyli „programowanie neurolingwistyczne". W fantasmagoriach NLP nie ma nic z nauki zwanej neurologią, nic z programowania (chyba, że „programowaniem" nazwiemy prostackie odwołania do odruchu warunkowego Pawłowa), nic z lingwistyki – ale jak ta nazwa brzmi!

Nie muszę dodawać, że tezy NLP nie znajdują żadnego poparcia w badaniach i doświadczeniach, ale kogo to obchodzi?

Gwałtowna potrzeba pop-psychologii

Informatycy oraz inni uczestnicy projektów IT nie są wyjątkiem w swoim poszukiwaniu pop-rozwiązań i wierze w skuteczność pop-psychologii. Psychologia naukowa nie ma żadnych szans w starciu z pop-psychologią na wolnym rynku. Ludzie chcą i potrzebują rozwiązań, ale nie chcą stawać się naukowcami, to oczywiste. Każdy lata samolotem, korzysta z komputera i posługuje się telefonem komórkowym, nie czując żadnej potrzeby zrozumienia fizyki, matematyki oraz informatyki, które działanie tych urządzeń umożliwiają.

Mimo tego, nie mamy do czynienia z powszechnym zalewem pop-fizyki i masowej wiary w jej pseudo-prawdy, tak jak w psychologii. Dlaczego? Z dwóch powodów.

Po pierwsze, osiągnięcia naukowe fizyki rzeczywiście zaspokajają wiele ludzkich potrzeb, ba! – nawet tych potrzeb, które nam wcześniej nie przychodziły do głowy. Ledwo sto pięćdziesiąt lat temu nie było ani samochodów, ani telefonów, ale czytając literaturę lub oglądając malarstwo z tego okresu, na próżno szukalibyśmy w nich wyrazu ówczesnej ludzkiej tęsknoty za samochodami i telefonami. Natomiast ludzka potrzeba zmniejszenia cierpienia jest ta sama od tysięcy lat, ogromnie silna, i – tak nam się przynajmniej wydaje – wciąż niezaspokojona. Presja, jaką ludzkie emocje i potrzeby wywierają na psychologię, jest znacznie silniejsza, niż analogiczna presja wywierana na fizykę, a psychologia, choć ma wiele osiągnięć, jest bez wątpienia nauką mniej dojrzałą, niż fizyka.

Po drugie, prawdziwość i skuteczność teorii fizycznych jest znacznie łatwiejsza do zweryfikowania, niż teorii psychologicznych. Ktoś, kto zdalną telepatią chciałby wyprzeć telefonię komórkową, szybko wypadłby z obiegu, bo przez telefon da się rozmawiać, a telepatycznie – to nie bardzo. Natomiast kryteria prawdy i nieprawdy w

psychologii są mniej jednoznaczne, w najlepszym razie – statystyczne, co wyklucza ich zrozumienie przez 99% ludzi, którzy nie rozumieją, co to jest rozkład normalny, korelacja, związek przyczynowo-skutkowy oraz statystyczna istotność. Myślę, że dotyczy to także 99% informatyków.

Dlatego pop-psychologia szerzy się bardziej, niż pop-medycyna (choć i tam pop-lekarzy, typu homeopaci, nie brakuje) i znacznie bardziej, niż pop-fizyka.

Pop-psychologia zdominowała nasz sposób myślenia o ludzkiej psychice. Osiągnięcia psychologii naukowej, choć niemałe, są prawie nieznane, natomiast przedmiotem publicznego dyskursu są rozmaite psycho-religie i szalbiercze pop-terapie, które nie leczą, a w najlepszym razie nie wiadomo, czy leczą. Przykładowo, jedyna forma psychoterapii o naukowo potwierdzonej skuteczności, tak zwana terapia poznawczo-behawioralna, długo czekała na oficjalne uznanie, podczas gdy mistyczno-kłamliwa psychoanaliza wciąż pozwala swoim wyznawcom zgarniać grube pieniądze kosztem autentycznie cierpiących ludzi.

Informatycy kochają pop-psychologię

Ojej, a co to wszystko ma wspólnego z IT? Wszakże my, informatycy, ludzie zimni i racjonalni, takim głupotom nie ulegamy, a przynajmniej nie podczas wykonywania naszego zawodu. Prawda?

Zapewne, w jakimś stopniu zawód informatyka, podobnie jak stolarza czy hydraulika, podlega nieustannej empirycznej weryfikacji, która bezlitośnie wykasza pop-pomysły. Ktokolwiek napisze kod w języku C albo C++ wyrażenie:

```
if (a = 17 || b > 36)
```

mając nadzieję, że wyrażenie to kiedykolwiek będzie fałszywe, wyczytawszy może na jakimś pop-blogu na temat „operatorów intencjonalnych", które jakoby zmieniają swą funkcję zgodnie z intencją programisty :), szybko przekona się o swojej pomyłce.

Jednak informatyka, to nie tylko programowanie, a z kolei IT, to znacznie więcej, niż sama informatyka. W nich jest już ogromne pole do dowolności, pop-teorii lub teorii bardzo wprawdzie zawiłych i zaawansowanych, ale niepotwierdzonych empirycznie. Gołosłowne pseudo-argumenty zmieniają treść, ale nie formę. W roku 1984 przysłuchiwałem się zawziętej dyskusji zwolenniczki języków strukturalnych, w tym „Pascala", z fanem „Cobolu", a równo trzydzieści lat później, w 2014, równie gorącemu sporowi zwolenniczki „Windows" ze wielbicielem „iOS". W latach 80-ych toczyły się gorące walki o wyższość paradygmatu strukturalnego nad innymi, w latach 90-ych religijną niemal czcią cieszyła się obiektowość, a od dziesięciu lat każdy chce być agile, lean startup oraz stosować ciągłą integrację. Czasem się z tymi lub tamtymi poglądami zgadzam, czasem nie. Stosowane argumenty bywają i głupie, i mądre, ale nigdy nie spotykamy się z odwołaniem do wyników empirycznych inaczej, niż hasłowo. Tak, można wprawdzie przeczytać, że „Scrum został osadzony w teorii empirycznej kontroli procesu, lub krócej w teorii empiryzmu. Empiryzm reprezentuje pogląd, iż wiedza wynika z doświadczania"[2], to ładne i

nawet rozsądne słowa, ale brak jest powszechnie znanych, systematycznych (nie anegdotycznych, bo nie są one statystycznie istotne!), empirycznych danych na temat skuteczności, lub jej braku, różnych metodyk realizowania projektów IT, w tym i Scruma [3], [9].

Skoro my, ludzie IT, lubimy wierzyć w cuda nawet na naszym własnym, informatycznym podwórku, to co dopiero, kiedy wchodzimy na cudze podwórko, na tereny psychologii. Czujemy przy tym taki seksowny dreszczyk, i – hulaj dusza! Dowolne gołosłowne ogólniki, albo zdania zupełnie banalne i trywialne, przemyca się pod dumnym szyldem „psychologii". Poznajcie? To chyba z jakiegoś podręcznika „jak być szczęśliwym w małżeństwie", prawda?

Zacznij od współpracy, a nie od wojny.

Komunikuj informacje w sposób neutralny, skoncentrowany na faktach bez krytykowania autora.

Spróbuj zrozumieć, co druga osoba czuje i dlaczego reaguje tak jak reaguje.

Upewnij się, że druga strona zrozumiała, co powiedziałeś i upewnij się, że rozumiesz uwagi drugiej strony.

Metoda naukowa a modne bzdury

Rama programowa do nauczania przedmiotów matematyczno-przyrodniczych w szkołach średnich obejmuje wytłumaczenie i praktyczne zastosowanie metody naukowej (stawianie hipotez, weryfikacja hipotez przez obserwację lub eksperyment, analiza wyników i wyciąganie wniosków), ale efekty tego są najwyraźniej mizerne. Stosowanie metody naukowej nie jest wpisane w nasze geny – dla przeżycia, i przekazania swoich genów dalej, najwyraźniej skuteczniejsze są inne metody, takie na skróty.

Wyobraźmy sobie, że istnieją geny: łatwowierności oraz myślenia naukowego. Ludzie łatwowierni mają skłonność do postępowania zgodnie z obiegową opinią, słuchania plotek, szybkiego wyrabiania sobie poglądów. W zamian, uzyskują w 40% dostęp do informacji prawdziwej, a w 60% - do fałszywej. Jednocześnie gwarantują sobie akceptację grupy i dobre samopoczucie. Ludzie myślący naukowo uzyskują – załóżmy - wiedzę prawdziwą w 80%, a fałszywą – w 20%. Niestety, po pierwsze, tracą mnóstwo energii na swoje obsesyjne dążenie do prawdy, a po drugie, narażają się na niechęć grupy, hejterów, złe samopoczucie. Czyli, cud to prawdziwy, że geny dociekliwości i samodzielnego myślenia nie wyginęły zupełnie, skazując nas na krzepiącą emocjonalnie, ale fałszywą wiarę w różne bóstwa, oraz na stosowanie mającego liczne lajki krzemiennego pięściaka zamiast niepopularnych narzędzi z kłopotliwego w fabrykacji żelaza.

Co lepsze: sucha teoria czy mokre anegdoty?

Wracając na teren IT, możemy obserwować nie od dziś fascynację „praktycznymi przykładami" i niechęć do „abstrakcyjnej teorii". Śledząc w sieci opinie na temat sy-

stemu certyfikacji ISTQB, można często spotkać się z zarzutami, że jest zbyt teoretyczny, że to – cytuję – „sucha teoria". Wynika z tego, że autor tej krytyki jest zdania, że znajomość teorii nie pomoże mu w praktyce, zaś znajomość przykładów, anegdot – pomoże? Nie jestem wprawdzie miłośnikiem ISTQB, gdyż ten system wiedzy jest moim zdaniem w wielu miejscach błędny, niepełny i niespójny, ale cenię w nim, że choć podjął wyzwanie zbudowania takiej teorii, nie poprzestając na anegdotach – nawet znakomitych i trafnych! - w stylu „lessons learned in software testing" [7].

Tę samą tendencję mają członkowie rad programowych wielu konferencji. Każdy banał, byle opakowany w ramki konkretnego doświadczenia z projektu, ma większe szanse dostać się do programu, niż najmądrzejsze nawet uogólnienie [4]. Przykładowo, temat „moje doświadczenia z użyciem Cucumber w projekcie XYZ" cieszy się na pewno większym powodzeniem niż temat „generyczny model klasyfikacji narzędzi testowych".

To samo, oczywiście, dotyka psychologii. Pop-psychoreligia oferuje gotowe, nietrudne do zastosowania, czasem nawet odrobinę skuteczne, przepisy na nasze autentyczne, a przynajmniej modne, kłopoty i swędzenia, zarówno w życiu jak i w biznesie. Nauka zwana psychologią wymaga czasu i wysiłku, nie pomoże poczuć się lepiej tu i teraz.

Badania nad tym, jak ludzie widzą, pozwoliły wykryć, że najłatwiej zauważamy kolor jaskrawożółty i jaskrawo-pomarańczowy. Dzięki temu, w takich kolorach są dziś setki milionów ubrań i kamizelek roboczych; dzięki temu setki tysięcy ludzi uniknęły śmierci w wypadkach [5], ale nie jesteśmy za to psychologom specjalnie wdzięczni, natomiast chętnie popadamy w radosną ekstazę, czytając rozmaite „obudź w sobie olbrzyma" [8], bo wydaje nam się, że to nam robi dobrze. Jeszcze lepiej robi nam kokaina (nielegalna), alkohol (legalny) oraz rozmaite inne uzależnienia i kompulsje, ale ich nie polecam, bo mają krótkotrwały efekt pozytywny, a długotrwały – szkodliwy. Lepiej w życiu znać prawdę, a jeśli dusza boli, szukać pociechy w jedynym według Martina Seligmana skutecznym środku – medytacji [6].

Pop-psychologia i co dalej?

Co dalej? Wydorośleć i nie ulegać popularnym głupotom, nie brać za prawdę tego, co tylko hałaśliwe i modne. Opłaca się sceptycznie patrzeć na świat i uważnie badać to, co nam dają do wierzenia.

Czyżbym proponował chodny racjonalizm w miejsce romantyzmu i uniesień? Nic podobnego. Sam jestem miłośnikiem rozmaitych pop-psychoreligii, na przykład lubię uczestniczyć w tak zwanych ustawieniach Berta Hellingera (podkreślam – LUBIĘ, a nie wierzę w ich prawdziwość!). W trudach życia na tym niełatwym świecie pociechę dają mi górskie wędrówki, cenię sobie rozmaite rytuały, niemające żadnego naukowego uzasadnienia. Ale – nie robię z tego ideologii. Nie udaję, że to, co działa dla mnie, na pewno musi działać dla innych. Nie biorę za to pieniędzy. Nie oszukuję.

Pop-psychologia, to może być znakomita zabawa, czasem autentyczna pomoc i pociecha, ale tylko wtedy, jeśli nie pozwolimy, aby nami zawładnęła.

AUTOR

Bogdan Bereza

Posiada wieloletnie doświadczenie w inżynierii jakości.

Jest skutecznym konsultantem, trenerem i prelegentem, autorem kilku książek i wielu artykułów z dziedziny inżynierii oprogramowania. Jest ceniony za umiejętność szerokiego, kompleksowego traktowania jakości oraz łączenia w całość różnorodnych obszarów, takich jak programowanie, projektowanie systemów, inżynieria wymagań, zarządzanie projektami oraz psychologiczne aspektów projektów IT. Jest inicjatorem i wiceprezesem Stowarzyszenia Inżynierii Wymagań, a także inicjatorem i organizatorem 1-szego Ogólnopolskiego Turnieju Inżynierii Wymagań.

REFERENCJE

[1] Polecam rozmowę z Jackiem Santorskim o folwarcznym stylu zarządzania w IT, w http://itwiz.pl/kiosk/kiosk/magazyn-itwiz-332014/ (oraz w http://blogomocja.blogspot.com/2014/07/brawo-santorski.html).

[2] https://www.scrum.org/Portals/0/Documents/Scrum%20Guides/2013/Scrum-Guide-PL.pdf

[3] http://requirementsjournal.com/ENG/Papers/STAR_1998_paper.pdf

[4] Nie twierdzę, że każde uogólnienie jest mądre – można wśród nich trafić na modne bzdury tak samo, jak wśród konkretów i przykładów. Jednak uogólnienie ma przynajmniej szansę, by stworzyć nową wartość, zaś przykład zawsze pozostanie tylko przykładem, anegdotą, ciekawostką o wątpliwej wartości dla szerzenia wiedzy.

[5] Tomasz Witkowski „Zakazana psychologia"; polecam też jego blog http://tomaszwitkowski.pl.

[6] Martin Seligman „Co możesz zmienić, a czego nie możesz".

[7] Cem Kaner, James Bach, Bret Pettichord „Lessons Learned in Software Testing"

[8] Anthony Robbins „Obudź w sobie olbrzyma"

[9] Dziękuję Redakcji Quale za zwrócenie mi uwagi na bardzo ciekawe pozycje: www.scrumalliance.org/resource_download/1156, www.slideshare.net/melnik/empirical-evidence-of-agile-methods-190997, www.amazon.com/Making-Software-Really-Works-Believe/dp/0596808321. Rzeczywiście, one dotyczą tej tematyki, i są oparte na empirycznych obserwacjach, ale: (1) są głównie jakościowe, nie ilościowe; (2) są obserwacją, nie eksperymentem, więc nie pozwalają na wyciągnięcie wniosków o związku przyczynowo-skutkowym, a wyłącznie o korelacji; (3) nie pozwalają na wykluczenie działania trzeciej zmiennej.

testerzy.pl

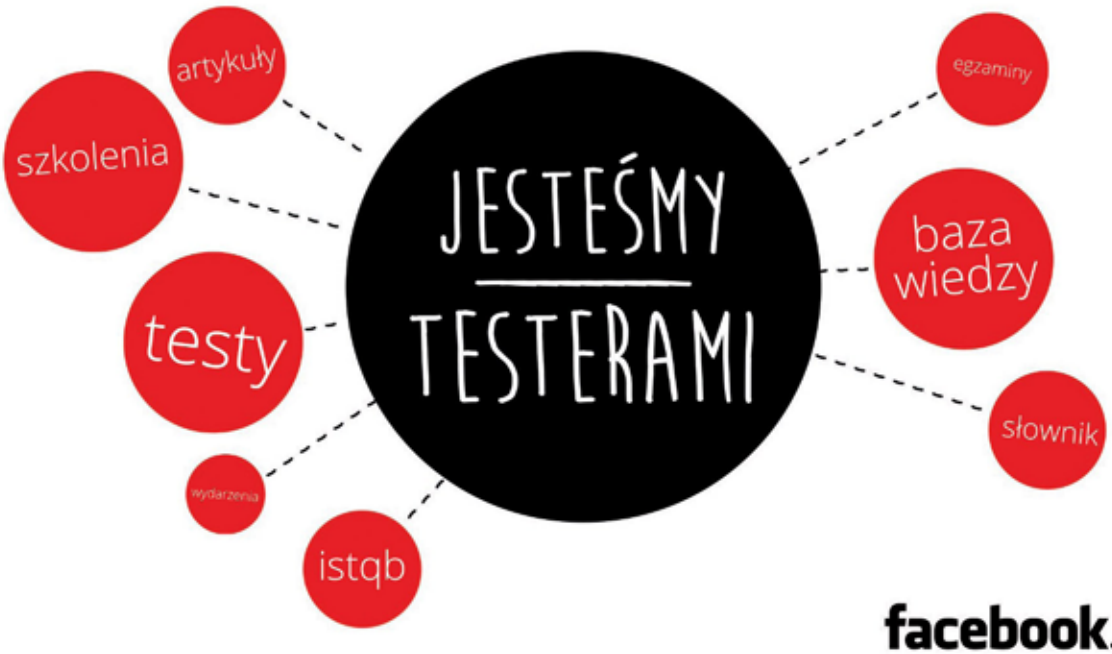
ISTQB Silver Partner
International Software Testing Qualifications Board

Agile Tester

Certified Tester Foundation Level Extension

szkolenie & egzamin

PIERWSZE W POLSCE!



facebook.
facebook.com/testerzy

Justyna Rybarczyk
Agnieszka Skokowska

Jak złapać testera

Prelegent na TESTWAREZ

Wprowadzenie

Żyjemy w czasach dominacji komputerów. Nie ma już chyba dziedziny życia, której nie zdominowałyby komputery. Branża IT rozwija się od wielu lat w sposób imponujący, przyciągając do siebie setki młodych ludzi. Pojawia się też coraz więcej nowych firm, często z zachodu Europy, które otwierają filie u nas i poszukują developerów i testerów.

Jednocześnie zmiany w naszym kraju oraz potrzeby terminowych projektów doprowadziły do pojawienia się na rynku pracy nowych umów takich jak umowy zlecenia, czy tez umowy kontraktorskie. Mają one na pewno jedną wspólną cechę – są ograniczone czasowo.

Wniosek jest taki, iż ruch na rynku pracy w obszarze IT jest ogromny. Prowadzi to często do sytuacji, w której firmy prowadzą stałą rekrutację na wybrane stanowiska lub też rekrutacje otwierane są kilkakrotnie w roku.

Niesie to za sobą duże koszty wszelakiego rodzaju – zarówno finansowe jak i psychologiczne. Ilość czasu, którą trzeba poświęcić na wybranie odpowiedniego pracownika jest ogromna. Do tego czas ten bardzo często muszą poświęcać osoby, najbardziej cenione w firmie, czyli pracownicy techniczni oraz kierownicy działów.

Każda rekrutacja i zatrudnienie nowej osoby niesie ze sobą ryzyko. Ponosi je i firma i dział, do którego osoba jest rekrutowana. Szybkie odejście nowozatrudnionej osoby można uznać za porażkę i dobrze gdyby takich porażek było jak najmniej. Oczywiście każda rozwijająca się firma musi ponieść zarówno koszty jak i ryzyka, chcąc mieć dobrych, wykwalifikowanych pracowników. Jednak ważne jest, aby zarówno jedno jak i drugie minimalizować. Pod koniec chciałybyśmy jeszcze wspomnieć o innej rekrutacji, do projektów, która wydaje się mieć wiele ciekawych aspektów, o których warto dyskutować.

Na początek zróbmy przegląd poszczególnych rekrutacji i przyjrzyjmy się problemom, które im towarzyszą.

Zacznijmy, od tych wydawać by się mogło, najprostszych i najłatwiejszych – praktyki, staże oraz stanowiska juniorskie. Bo przecież znaleźć taka osobę jest niezmiernie prosto. Czy aby na pewno?

Praktykanci

Szukamy młodych ludzi zwykle wśród studentów informatyki bądź pokrewnych dziedzin. A co dalej? Jakie stawiamy wymagania przed taką osobą?

Z naszego doświadczenia wynika, iż szukamy osobowości, ludzi chętnych do pracy, niebojących się wyzwań, zaangażowanych i chętnych do nauki. Szukamy konkretnych cech charakteru – nie stawiamy tu przecież na doświadczenie ani umiejętności nabyte w poprzednich pracach. Niezmiernie ważna też jest komunikatywność. Brak porozumienia między praktykantem a zespołem, do którego trafił, czy choćby pomiędzy opiekunem praktyki, eliminuje taką osobę praktycznie po miesiącu.

Ale praktyki nie trwają wiecznie i przychodzi czas podsumowań i ważnego pytania – co dalej praktykancie? Jeśli nasza ocena jest wysoka i widać potencjał młodej osoby, chętnie ją rozwijamy i proponujemy dalszą współpracę. Ale czy na pewno ma to sens?

Wiele razy zdarza się, że rokująca młoda osoba zdobędzie trochę wiedzy i po prostu ucieka do konkurencji. I wtedy przychodzi refleksja – czy warto było? W końcu poświęciliśmy wiele czasu i energii, aby wykształcić nowego testera.

Ale odpowiedź wydaje się być oczywista - warto! Zawsze warto poświęcić czas młodej osobie pełnej chęci do pracy. Warto namawiać do testów i przedstawiać naszą działkę IT jako pracę ciekawą i z możliwościami. Bo przecież nie każdy praktykant wie, co chce robić po studiach, nawet tych informatycznych. Zachęcenie do testów i dowiedzenie się, że tak „to chcę w życiu robić" jest wielką nagrodą, nawet, jeśli po niedługim czasie ta osoba nas opuści.

Juniorzy

To już trochę inna kategoria, choć w zasadzie cechy, których szukamy u takiej osoby są dość podobne jak w przypadku praktykanta. Różnica jest taka, że możemy postawić tu na pewne nakierowanie na testy, być może otarcie się o nie gdzieś w przeszłości.

Ale czy szukamy tylko wśród ludzi po informatyce? Niestety oczywistym wydaje się fakt, że nie spotkamy się ze 100 osoba-

mi, a niejednokrotnie takie właśnie ilości CVitek spływają na ogłoszenia na juniora testera. A my musimy dokonać wstępnej selekcji i jest ona najczęściej podyktowana właśnie wykształceniem. Czy to słuszna zasada? Jakąś trzeba przyjąć, a ta wydaje się być najprostsza i najlogiczniejsza.

Dobrze jest jednak odpowiedzieć sobie na pytanie, kogo takie postawienie sprawy nam determinuje w przyszłości? Bo przecież osoba techniczna znająca podstawy programowania niekoniecznie będzie dobrym testerem. A może powinniśmy zacząć od pytania, kogo szukamy i potrzebujemy? Jeśli ma to być tester analityk na przykład, niekoniecznie należy skupiać się tylko na kandydatach po informatyce. Doświadczenie pokazuje, że poświęcenie trochę więcej czasu takiej rekrutacji, czyli spotkanie się nie tylko z osobami po uczelniach technicznych, może zaprocentować w przyszłości kimś wyjątkowym.

Specjaliści/ managerowie

W tych przypadkach wymagania wydają się w miarę oczywiste. Obeznanie z rzemiosłem plus cechy osobowości, które cenimy – na przykład komunikatywność, otwartość, bezpośredniość. Do tego ISTQB. Oczywiście wiele dodatkowych rzeczy będzie działało in plus – wiedza biznesowa z danego obszaru, dyspozycyjność, znajomość AGILE tak teraz popularnego, zarządzanie zespołem w przeszłości, doświadczenie.

Jak się dowiedzieć czy mamy do czynienia z testerem czy cwaniakiem, który liznął trochę testów i próbuje nabić nas w butelkę? Taki problem w zasadzie nie istnieje w przypadku, gdy dana firma posiada Test Managera czy Test Leadera. Po półgodzinnej rozmowie wszystko już jest jasne i żadna super CVitka takiej osobie nie pomoże. Rozmowa obnaża w sposób bezlitosny wszystkie jej braki.

A co jeśli takiej osoby nie ma w firmie? Pozostaje wybrać odpowiednią CV, w której ilość opisanych projektów sugeruje doświadczenie, przygotować sobie zestaw pytań i zdać się na intuicję.

Całkiem sporo firm korzysta też z aplikacji internetowych napisanych na własny użytek, które w formie testu mają sprawdzić kandydata. Mamy mieszane uczucia, co do zasadności ich używania szczególnie w przypadku testów dla testerów manualnych. Bo czy ma sens zadawanie pytań typu: „Czym jest black i white box?" lub „Czym są testy funkcjonalne?" w aplikacji internetowej, ograniczonej czasowo na przykład do godziny? W czym pomoże nam taki test? Na jakie pytania odpowie? Może okazać się, że odsiewamy osoby niekoniecznie szybkie w odpowiedziach, ale za to dokładne, które zaczynają rozpisywać się w odpowiedzi i nie starcza im czasu na ich dokończenie. Tester nie musi być szybki, ale musi być przecież dokładny.

Zupełnie inna sprawa ma się w przypadku, gdy szukamy test inżyniera – tu test techniczny, odsieje nam przypadkowych kandydatów, na których szkoda byłoby czasu podczas rekrutacji. Doświadczony inżynier sprawdzający test od razu wychwyci, z kim warto porozmawiać, a na kogo szkoda czasu. Właśnie w takim przypadku test internetowy zdaje się mieć sens i sprawdza się jego zastosowanie w 100%.

Czy na rozmowie są pytania, które są uniwersalne i od razu rozwieją nasze wątpli-

wości? Każdy rekrutujący ma zestaw takich, które go wspomogą, na przykład: „Jak przetestowałbyś bankomat?", „Czy musiałeś kiedyś pracować bez dokumentacji?" „Jakie testy byś przeprowadził w takim przypadku?" „Czy wiesz, czym jest SCRUM?", „Jakie najwcześniej testy przeprowadzałeś w projekcie?" „Jak zarządzałeś błędami w JIRA (inny tool do wpisywania błędów)?", „Czy miałeś do czynienia z procesem testowym?" „Czy wg Ciebie był to odpowiedni proces dla firmy, czy się sprawdzał?", „Czy oceniałeś ryzyka dla projektu?" „Czy szacowałeś liczbę testów na poziomie oferty?" „W przypadku zagrożenia projektu, jakie działania byś podjął lub musiałeś podjąć, aby zminimalizować groźbę fiaska?"

Rekrutacje na stanowiska dla doświadczonych testerów często są utrudnione ze względu na to, iż wiele z tych osób pracuje. Pojawia się wtedy problem z umówieniem na rozmowę. Z pomocą przychodzi technika. Obecnie wielu z nas pracuje w zespołach rozproszonych agilowych, gdzie wszelakiego rodzaju komunikatory zastępują nam codzienne spotkania. Z tego też korzystają obecnie rekruterzy – rozmowa przez Skype czy Hangout nie dziwi już nikogo. Jest tylko jeden warunek – nie może to być połączenie złej jakości. Dopytywanie się, co chwile co Pani/Pan powiedział zdecydowanie bardziej przeszkadza niż pomaga.

Jednocześnie być może już na jedno pytanie otrzymamy odpowiedź – jak dana osoba poradzi sobie w zespole rozproszonym i czy taka komunikacja nie będzie dla niej problemem?

Rekrutacje do projektów

Jesteśmy już po rekrutacjach, dysponujemy zespołem testerów, ludźmi na różnych stanowiskach, o różnych kwalifikacjach i różnych składkach kosztowych.

Co dalej?

Przypisanie testerów do projektów. Wydaje się, że jest to proste zadanie - przypisujemy do projektu najlepszą wolną osobę. Nasuwa się pytanie, co to znaczy najlepszą? Najlepszą z punktu widzenia umiejętności/predyspozycji testerskich, najlepszą z punktu widzenia znajomości biznesu, najlepszą z punktu widzenia stawki kosztowej? Wydawać by się mogło, że to ostatnie nie jest aż tak ważne, przecież rozmawiamy o jakości. Jeżeli jednak oprogramowanie nie jest krytyczne pod względem bezpieczeństwa (safety critical software), to niestety okazuje się, że w starciu z Project Managerem (PM) cena testera jest bardzo ważna, a czasem nawet kluczowa.

Naszym celem, z punktu widzenia Test Managera jest wyselekcjonowanie odpowiedniego pracownika, doświadczonego, efektywnego, który posiada pożądaną wiedzę biznesową. W praktyce wygląda to często inaczej, napotykamy się na mur, czyli pytanie „ile"?

Jak sobie wtedy radzić, jakie mamy możliwości?

Zmniejszamy zespół QA, obniżamy wymagania, co do kompetencji testerów, ograniczamy zarządzanie testami/proces testowy (rezygnacja z Test Managera, rezygnacja z niezależności zespołu QA), wspieramy się studentami (przy metodach formalnych), rezygnujemy z formalnego procesu testo-

wego i stawiamy na testy eksploracyjne. Każde z tych rozwiązań ma zarówno plusy jak i minusy.

Zmniejszając zespół, ryzykujemy, że nie damy rady pokryć wszystkich wymagań, należy się wtedy zastanowić nad analizą ryzyka. Tak zaplanować testy, aby zwrócić szczególną uwagę na te fragmenty systemu, które muszą być naprawione przed wdrożeniem.

W przypadku obniżenia wymagań, co do kompetencji testerów, dobrym rozwiązaniem jest przeanalizowanie, jaki wpływ, na jakość danego projektu będzie miała rezygnacja z poszczególnych kwalifikacji i co będzie najmniejszym złem. Do wyboru mamy: wiedzę biznesową, umiejętności techniczne, predyspozycje testerskie, kompetencji zarządzania.

Kolejną możliwością na obniżenie kosztów i uzyskanie kompromisu z PM jest zredukowanie procesu testowego, począwszy od zlikwidowania roli Test Managera - testerzy przechodzą pod architekta/PM. Należy jednak pamiętać, że tracimy w takiej sytuacji niezależność, a co za tym idzie w większości przypadków tracimy na jakości.

Utrzymanie procesu formalnego przy obniżonych kosztach może mieć zastosowanie przy wsparciu projektu osobami niedoświadczonymi (np. studentami). Taki proces charakteryzuje się szczegółową dokumentacją: planem testów, dobrze zaprojektowaną specyfikacją testową (scenariusze testowe) – na początku możemy wykorzystać doświadczonego testera a następnie do wykonywania scenariuszy zatrudnić znacznie tańsze osoby. Powszechna jest opinia, że „testować może każdy", w rzeczywistości ma to zastosowanie wtedy i tylko wtedy, gdy za przygotowanie/planowanie i nadzór odpowiedzialna jest osoba doświadczona w naszym przypadku Test Manager.

Kolejną możliwością są testy eksploracyjne, w których nie ma planowania prac (planu testów), ani scenariuszy testowych. Rezygnując z dokumentacji, formalnego procesu musimy jednak pamiętać, że do takich testów należy przypisać kompetentnych testerów, a nie przypadkowe osoby. To nie jest kwestia „przeklikania się" wg wcześniej przygotowanych scenariuszy testowych, ważne jest tu analityczne myślenie, zwracanie uwagi na wszystko (na każdą wątpliwość) oraz zdolności do wyszukiwania błędów (stwarzania problemów). Oczywiście, tego typu podejście nie jest równoznaczne z metodami formalnymi, ale po odpowiednim dobraniu testera pozwala zmniejszyć koszty przy sporych korzyściach.

Nie znamy jednoznacznej odpowiedzi, który wariant jest najlepszy. Wybierając którykolwiek z powyższych należy się liczyć, że ryzykujemy, co w praktyce oznacza często, gorszą jakość testowanego systemu. Jak traktować kompetencję posiadania wiedzy biznesowej? Na ile jest ona kluczowa przy doborze testera do projektu?

Ekspert ze znajomością specyfiki danego biznesu jest najbardziej pożądanym kandydatem. Tester, który łączy predyspozycje testerskie wraz z intuicją biznesową potrafi szybciej, sprawniej odnaleźć się w dokumentacji i samym systemie, co sprawia, że jego testy są efektywniejsze.

W sytuacji, gdy mamy do czynienia ze specjalistą bez wiedzy biznesowej istotna

jest dokumentacja. Jeżeli jest ona dobrze, szczegółowo napisana to profesjonalista poradzi sobie z testami bez problemu. Jeśli specyfikacja jest uboga i nie ma możliwości dopytania się, tester musi się domyślać, jak aplikacja ma działać i interpretować według własnego rozumienia, ważne są tu umiejętności/intuicja danej osoby. W takiej sytuacji, w czasie testów dostawcy stawiamy głównie na wykrywanie błędów, a następnie oczekiwane są testy akceptacyjne użytkownika, które potwierdzą, że system poprawnie realizuje procesy biznesowe.

Doświadczenie pokazuje, że gorszym rozwiązaniem z punktu widzenia stabilności systemu jest osoba tylko z wiedzą biznesową. Po pierwsze taki „tester" może mieć problem z odnalezieniem się w systemie, który jest w trakcie tworzenia, po drugie może wystąpić problem z komunikacją miedzy nim a zespołem developerskim (m.in. sposób przekazania informacji o błędzie autorowi kodu), a po trzecie zabraknie myślenia analitycznego i zdolności wykrywania usterek czy awarii.

Co zrobić w sytuacji, gdy projekt tonie? Czy przerzucanie testerów pomiędzy projektami to dobra praktyka? Czy taka reakcja może pomóc w uzyskaniu lepszej, jakości w projekcie? Sytuacje takie zdarzają się z różnych powodów, np. opóźnionego developmentu, niewystarczającej analizy czy złego przypisania osoby do projektu, które mogłoby być spowodowane wcześniej wspomnianą stawką kosztową. Często w takich okolicznościach, zapominamy o kosztach i dostajemy wolną rękę w działaniu. Zaczynamy od weryfikacji procesu testowego użytego w projekcie, sprawdzamy istniejącą dokumentację, faktyczne wykonanie testów, porównujemy to z kompetencjami zaangażowanych w projekt osób. Rozwiązaniem może być zwiększenie zasobów, nie zawsze się to jednak sprawdza. Ma to zastosowanie, gdy mamy problem z pokryciem wymagań (spraw-

dza się w przypadku formalnego procesu i dużej ilości scenariuszy testowych). Jeżeli problem tkwi w zarządzaniu można przypisać do projektu Test Managera, który poprowadzi testy, usprawni komunikację. Czasem dobrym pomysłem jest przypisanie testera z wiedzą biznesową (pomaga to w sytuacjach, gdy np. okazuje się, że nie mam dobrze rozpisanych scenariuszy testowych) lub wiedzą techniczną (np. testy automatyczne czy testy na bazie).

Jeśli analiza wykazała, że potrzebujemy konkretnych kompetencji, należy się zastanowić nad ewentualnym przesunięciem osób z innych projektów. Największą wartość dodaną otrzymujemy w przypadku kwalifikacji zarządzania, umiejętności technicznych czy znajomości wiedzy biznesowej. W takiej sytuacji należy ocenić ryzyko, jakie pojawi się w projekcie, z którego zabierzemy testera i wtedy zadecydować czy warto danego testera przerzucać.

Podsumujmy

Niestety, aby zdobyć dobrych pracowników należy ponieść koszty związane z rekrutacją i nie oszczędzając poświęcać rekrutacjom więcej czasu niż to często jest praktykowane. Nie wystarczy na pewno test godzinny czy też krótka rozmowa przez Skypa. Testerskie rekrutacje okazują się nie mniej poważnymi od developerskich, szczególnie w przypadku tych na stanowiska specjalistyczne.

Nie umniejszajmy też ważności tym praktykancko - juniorskich, bo przecież pomagamy kształtować przyszłe pokolenia testerów, co jest niezmiernie ważne.

W aspekcie pozyskiwania ludzi ciekawym problemem do dyskusji wydaje się być rekrutacja do projektów.

Testowanie, jako proces związany z wytwarzaniem oprogramowania, może być wdrożony w dowolnym momencie, może mieć obcięte koszty – należy jednak pamiętać, że nie powinno się oszczędzać ani na czasie (początek testów) ani na testerach (kompetencjach). Testowanie we wczesnych fazach projektu pozwala wykryć więcej defektów, nie tylko w kodzie, ale również w analizie wymagań. Dużo zależy także od kwalifikacji/predyspozycji testerów, wyniki „droższego" specjalisty/zespołu, zazwyczaj zwracają z nawiązką poniesione koszty. Późne wykrycie defektów, skutkuje, droższymi kosztami naprawy. W niektórych przypadkach, jeśli defekt zostanie znaleziony po wdrożeniu na produkcje, cena jest największa, ponieważ dochodzą koszty pogorszenia wizerunku (narażenia na śmieszność), utraty klientów czy korzyści.

Ale czyż nie brzmi to wszystko znajomo? Obyśmy doczekali jeszcze wdrożenia i trzymania się tych zasad w naszych firmach, czego sobie i wszystkim czytelnikom szczerze
życzymy :).

AUTORKI

Justyna Rybarczyk

Od wielu lat zwiazana z firmą IMPAQ. Kierownik zespolu testerów, obecnie na stanowisku test managera i team leadera.

Przez 17 lat pracy przeszła przez wszystkie stanowiska od juniora począwszy. W sumie brała udział w kilkudziesięciu projektach prowadzonych różną metodyką i z różnych obszarów tematycznych takich jak ubezpieczenia, gambling czy bankowość.

Uczestniczyła w wielu projektach międzynarodowych, miedzy innymi dla firmy amerykańskiej w Atlancie gdzie pracowała jako członek zespołu testerskiego klienta, przez 4 miesiące.

Obecnie jest test leaderem w dwóch projektach Agile'owych.

Posiadaczka certyfikatu ISTQB Advanced Level Test Manager.

Agnieszka Skokowska

Pracuje jako Test Manager w firmie IMPAQ. Absolwentka Matematycznych metod informatyki Politechniki Poznańskiej. Posiada 9-letnie doświadczenie w zapewnieniu jakości i testowaniu oprogramowania.

Brała udział w wielu projektach zarówno polskich jak i międzynarodowych dla różnych sektorów rynku (m.in. ubezpieczenia, gry hazardowe, marketing online). Na codzień zajmuje się zarządzaniem procesem testowym, wsparciem analizy biznessowej (zarządzanie wymaganiami /zmianą) oraz wykonywaniem testów eksploracyjnych.

Posiada certyfikat ISTQB Advanced level Test Manager.

Fundacja Governica

Fundacja Governica to pozarządowa organizacja non profit, która wspiera rozwój gospodarki, promuje innowacyjność i wspiera przedsiębiorczość poprzez upowszechnianie wykorzystywania standardów zarządzania, w tym najlepszych światowych praktyk, norm i modeli zarządzania.

Fundacja została ustanowiona 9 lutego 2012 roku przez Bartosza Górczyńskiego, Tomasza Mazurka, Krzysztofa Tomkiewicza oraz Michała Woszczynę.

Dołącz do nas!

HTTP://FUNDACJA.GOVERNICA.COM/

O Fundacji

Jednym z główny projektów Fundacji jest Governica.com - serwis społecznościowy budowany z myślą o dzieleniu się wiedzą i doświadczeniami, dyskusjach oraz tworzeniu i utrzymywaniu relacji przez ludzi zajmujących się praktycznie i teoretycznie zarządzaniem. Governica.com to wykorzystujący najnowsze technologie użyteczny przewodnik, prezentujący standardy i najlepsze praktyki zarządzania organizacjami. Uzupełnieniem encyklopedycznych artykułów są powiązane z nimi mikroblogi, newsy, przeglądy, analizy i zaproszenia oraz relacje z konferencji, seminariów i spotkań branżowych. To kompendium wiedzy o zarządzaniu, tworzone przez rozwijającą się wokół Governiki społeczność.

Wiedza zgromadzona w serwisie jest powszechnie i bezpłatnie dostępna na licencjach CC-BY-SA 3.0. Może więc być w nieskrępowany sposób wykorzystywana również w celach komercyjnych, bez konieczności wnoszenia opłat i pytania o zgodę właścicieli praw autorskich do tych materiałów. Zainspirowani ideą Web 2.0 udostępniamy serwis Governica.com i zapraszamy do jego współtworzenia wszystkich, którzy chcą się podzielić wiedzą i doświadczeniem z dziedziny zarządzania. Uprawnienia do dodawania i edytowania treści są przyznawane każdemu, kto zdecyduje się zarejestrować.

Społeczna kontrola prezentowanych treści nie jest jedynym mechanizmem gwarantującym, że są one wiarygodnym źródłem wiedzy. Prezentując standardy zarządzania współpracujemy z organizacjami i instytucjami formalnie nadzorującymi dany standard w naszym kraju. Powołujemy równocześnie Ekspertów Governiki nadając im specjalne uprawnienia w zakresie weryfikacji artykułów, zmiany haseł i kategorii. Do grona tego zapraszamy osoby obdarzone największym zaufaniem, a równocześnie będące uznanymi autorytetami w danej dziedzinie.

Współpraca

Jeśli zajmujesz się zarządzaniem i chcesz podzielić się z innymi swoją wiedzą oraz doświadczeniem zarejestruj się i dołącz do społeczności Governautów! Rejestracja zajmuje tylko chwilę, a profitów jest wiele.

TestingCup po raz trzeci

Ruszają przygotowania do kolejnej edycji Mistrzostw Polski w testowaniu oprogramowania

To już pewne – wiosną 2015 roku czeka nas III edycja najbardziej emocjonującego wydarzenia w polskim świecie testerskim. I tym razem nie zabraknie pozytywnych emocji, a na uczestników czekać będzie kilka niespodzianek.

Poprzeczka postawiona jest wysoko, oczekiwania uczestników I i II edycji wobec nas -organizatorów są spore. Tak jak II edycja zmieniła się pod wpływem sugestii i uwag uczestników pierwszego TestingCup, tak i tym razem zebrane opinie nie tylko motywują nasze działania, ale i są pomocne w tworzeniu kolejnych, jeszcze lepszych Mistrzostw.

Nie zapominajmy o tym, że Mistrzostwa Polski w Testowaniu Oprogramowania powstają dzięki zaangażowaniu, entuzjazmowi i pracy Wolontariuszy. Część z nich jest związana z imprezą od pomysłu jej powstania. Część zgłosiła się do nas, bo idea Mistrzostw jest im bliska i chcą zrobić coś dla społeczności testerskiej. Bez nich wszystkich nie byłoby TestingCup.

Jeśli chcesz dołączyć do naszej ekipy, współtworzyć z nami to wyjątkowe wydarzenie, wesprzeć nas swoim czasem, zaangażowaniem i pomysłami - nie zwlekaj, napisz do nas na testingcup@testingcup.pl.

A może Twoja firma chciałaby zostać partnerem Mistrzostw? Skontaktuj się z nami i dowiedz się dlaczego warto zostać sponsorem TestingCup 2015.

Jak dokładnie będzie wyglądać kolejna odsłona TestingCup – tego na razie nie możemy zdradzić. Datę i lokalizację ujawnimy już wkrótce, a kolejne informacje pojawiać się będą na testingcup.pl, naszym profilu na facebooku: https://www.facebook.com/TestingCup, a także na łamach magazynu Quale, który po raz kolejny wspiera nas medialne. Za merytoryczną część TestingCup odpowiadają, jak co roku, testerzy.pl.

Bądźcie czujni, Mr Buggy już nie może się doczekać na kolejne testerskie starcie.

Wydawca
VWT Polska Michał Kruszewski
Przy Lasku 8 lok. 52, 01-424 Warszawa
Numer NIP 5272137158
Numer REGON 142455963

Redaktor naczelny
Karolina Zmitrowicz
karolina.zmitrowicz@quale.pl

Zastępca redaktora naczelnego
Krzysztof Chytła
krzysztof.chytla@quale.pl

Redaktorzy
Bartłomiej Prędki
bartlomiej.predki@quale.pl
Michał Figarski
michal.figarski@quale.pl

Współpraca
Tomasz Olszewski
tomasz.olszewski@quale.pl
Piotr Poznański
piotr.poznanski@quale.pl

WWW
www.qualemagazine.com
www.quale.pl

Facebook
http://www.facebook.com/qualemagazine

Reklama
info@quale.pl

Współpraca
Osoby zainteresowane współpracą w zakresie publikacji prosimy o kontakt:
info@quale.pl

Wszystkie znaki firmowe zawarte w piśmie są własnością odpowiednich firm.

Prawa autorskie
Wszystkie opublikowane artykuły są objęte prawem autorskim autora lub przedsiębiorstwa. Wykorzystywanie w innych publikacjach, kopiowanie lub modyfikowanie zawartości artykułu bez pisemnego upoważnienia autora jest zabronione.

Grafiki użyte w magazynie

Okładka

Ball Landscape River Winter Rays Fractals Fractal
http://pixabay.com/en/ball-landscape-river-winter-rays-67200/

Treść

Man Businessmen Woman Economy Community
http://pixabay.com/en/man-businessmen-woman-economy-162951/
chicago-329249
http://pixabay.com/en/chicago-skyscraper-city-high-rise-329249/
directory-235079
http://pixabay.com/en/directory-signposts-wood-grain-235079/
Table Restaurant Furniture Glass Wine Drink
http://pixabay.com/en/table-restaurant-furniture-glass-71380/
Arrow Efficiency Cost Speed Quality Development
http://pixabay.com/en/arrow-efficiency-cost-speed-394143/
Beautiful Business Computer Female Girl Internet
http://pixabay.com/en/beautiful-business-computer-female-15728/