Jakoś czy jakość?

Zimowisko Linuksowe 2011

Marcin Stępnicki

Definicja jakości

- Inżynieria oprogramowania – niedoceniana rola
- Inżynier czy artysta? Jakie wykształcenie i/lub inklinacje ma programista?
- Definicja jakości - propozycje
- McDonald – czy produkuje żywność wysokiej jakości?

Definicja jakości

- Za Wikipedią:
  - jakość jako zgodność ze specyfikacją
  - Zespół cech i charakterystyk wyrobu lub usługi, które noszą w sobie zdolność zaspokojenia określonej potrzeby
- Jeżeli weźmiemy pod uwagę te definicje, żywność z McDonalda spełnia je w 100%
- Czy stać nas na jakość?

Jakość na przestrzeni wieków

- Obecnie (EULA):

Urządzenie to jest dostarczane bez żadnej gwarancji dotyczącej jego niezawodności, dokładności, istnienia lub nie oraz przydatności do dowolnego konkretnego celu, a *Produkty Bioalchemiczne* w szczególności nie gwarantują, poręczają, sugerują ani nie wyrażają opinii na temat użytkowania swego produktu do jakichkolwiek zastosowań jak również nie ponoszą żadnej odpowiedzialności prawnej ani materialnej wobec ciebie ani dowolnej innej osoby, istoty czy bóstwa za jakiegokolwiek rodzaju straty czy zniszczenia spowodowane przez to urządzenie lub obiekt albo przez jakiekolwiek próby zniszczenia go metodą uderzania o ścianę, zrzucania do głębokiej studni lub dowolnymi innymi środkami, a ponadto stwierdzają, że zgodę na ninjejszą umowę albo dowolną inną umowę którą ją zastąpi , wyrażasz, zbliżając się na odległość do pięciu mil od produktu, obserwując go przez potężne teleskopy lub dowolną inną metodą, ponieważ jesteś kretynem, który tak łatwo daje się zagadać, że przy kawałku złomu z wysoką ceną z radością akceptuje aroganckie i jednostronne warunki, których nie przyjąłby nawet przy torbie psich chrupek i używany jest na wyłączne ryzyko nabywcy.

T. Pratchett „Prawda”. s. 91, Warszawa 1999, przekład Piotr W. Cholewa, Prószyński i S-ka, ISBN 9788374695039

Jakość na przestrzeni wieków

- Kodeks Hammurabiego (1750 r. p.n.e)

Jeżeli budowniczy wybuduje dom dla mieszkańca i wykona jego konstrukcję tak słabą, że dom się zawali i spowoduje śmierć właściciela, to budowniczy powinien zostać zabity.

Jeżeli zawalenie się domu spowoduje śmierć syna właściciela, to należy zabić syna budowniczego.

Jeśli przy zawaleniu domu zniszczone zostanie dobro, to budowniczy ma wykonać to wszystko, co zostało zniszczone, ponieważ nie wybudował domu wystarczająco mocno, powinien więc odbudować go na własny koszt.

Jakość na przestrzeni wieków

- Ukaz 1723, Piotr I (1723 r.)

Rozkazuję wychłostać i zesłać na wygnanie do pracy w klasztorze właściciela fabryki w Tule Korniła Biełogłazowa, ponieważ on podlec, ośmielił się państwowemu wojsku sprzedać złe pistolety i fuzje. Starszego kontrolera Fomu Mitwy wychłostać i wysłać do Azowa, żeby nie stawiał pieczęci na źle wykonanej broni.

(...)

Nowemu właścicielowi fabryki Demidowowi nakazuję zbudować starszemu kontrolerowi i kontrolerowi pokoje nie gorsze od pokoi właściciela; jeżeli będą gorsze, niech się Demidow nie obraża, każę ściąć mu głowę.

Zmiana podejścia

- Jakość “wystarczająco dobra”
- Największy orędownik?

Zmiana podejścia

Microsoft

- Windows vs OS2 – 1990
- Ówczesne metodyki: CMM, ISO9000-3 zbyt kosztowne

Słynne porażki

- Arianne 5 lot 501 – zbyt duża wartość *64-bit floating point* nie rzutuje się na 16-bit sign*ed integer,* software wzięty z Arianne 4.
- Mariner I zbacza z kursu, zniszczony zdalnie przez kontrolę lotów – złe przełożenie wzoru matematycznego na kod,
- Therac-25 – radioterapia nowotworów, zbyt wysokie dawki promieniowania – race condition,
- Multidata Systems – trzeba narysować metalowe ekrany chroniące zdrową tkankę – soft umożliwiał cztery, technicy chcieli pięć. Rysowali więc jeden duży z dziurami. W zależności od kierunku rysowania „dziury” oprogramowanie rekomendowało normalną, lub podwojoną dawkę
- Intel Pentium 1993 – 475 mln $ - błędy rzędu 0.006% w niektórych operacjach zmiennoprzecinkowych

Jakość jest za darmo

- *Koszt nieprzestrzegania jakości jest zawsze wyższy niż koszt zapewnienia jakości*

Phill B. Crosby, „Quality is Free – The Art of Making Quality Certain”

Metodyki

- Jest ich mnóstwo

Capability Mature Model – ocena praktyk stosowanych podczas produkcji:

Poziomy:

1) działania ad hoc, proces niepowtarzalny
2) powtarzalność, zdefiniowanie podstawowych procedur, śledzenie kosztów, harmonogramu i funkcjonalności
3) definiowalność, spójny zbiór definicji i standardów na poziomie projektu i organizacji
4) zarządzalność – mierzalność rezultatów na podstawie zdefiniowanych metryk
5) optymalizacja – monitorowanie i wpływanie na poprawę procesów

Metodyki

- ISO 90003
- ogólnie – co zrobić żeby wytworzyć oprogramowanie dobrej jakości, dobre krótkie wprowadzenie - http://www.praxiom.com/iso-90003.htm ,
- http://www.sjsi.org/webgears/files/sjsi/doc/2BSzomanski.pdf

Metodyki

- ISO/IEC_15504 (SPICE) - Software Process Improvement and Capability Determination
- ISO_12207 - Software Lifecycle Processes – mniej dotyczy samego sposobu wytwarzania oprogramowania, ale raczej procesu od momentu podjęcia decyzji o potrzebie wprowadzenia nowego software'u przez decyzję o zakupie/napisaniu samodzielnie nowego i jego późniejsze utrzymanie i rozwój. Jego wymuszenie może działać na Zarząd wykazujący symptomy Zespołu Gansera (ICD-10 F44.8, http://pl.wikipedia.org/wiki/Zesp%C3%B3%C5%82_Gansera)

Metodyki

Zespół Gansera (otępienie rzekome, zespół przybliżonych odpowiedzi) – zaburzenie reaktywne, zaliczane do grupy zespołów sytuacyjnych na pograniczu psychozy i symulacji. Pacjenci w rozmowie wykazują się rażącą niewiedzą na temat powszechnie znanych faktów i zjawisk. Udzielają błędnych, ale przybliżonych odpowiedzi na pytania, np. pytani ile wynosi tuzin, odpowiadają, że trzynaście. Poza tym wykonują bezsensowne czynności, takie jak próba pisania odwrotną stroną długopisu lub włożenia klucza do zamka odwrotną stroną.

Metodyki

- TickIT(plus) – zestaw przewodników dla wytwórców, dostawców i klientów
- Nie ma uniwersalnej metody wytwarzania oprogramowania. Gdyby była, wszyscy by ją stosowali.

Metodyki

- „Ciężkie” - wymagające rozbudowy i utrzymania aparatu nadzorującego:

- ISO 9000-9001, Six Sigma, CMMI, COBIT, ITIL, TickIT, ISO/IEC 122007, Bootstrap, SPICE, ISO/IEC 15504 + TMM, MMAST, TAP, TCMM, TIM, TOM, TSM, TPI

- model „waterfall” .

- Co to jest wielbłąd?

- rezerwat leśnych dziadków.

zarzuty: koszty, ignorowanie mechanizmów psychologicznych i społecznych, zawiłość. Czy jakość projektu przekłada się na jakość produktu?

Metodyki

- „Lekkie” - minimalizujące wszystkie działania niekonstrukcyjne.
- Rezerwat młodych wilków.
- XP
- Agile (SCRUM)
- zarzuty: rezygnacja z doświadczenia, gromadzenia i stosowania dobrych praktyk, systematyzacji i dyscypliny

Metodyki

Automated Defect Prevention – teoretyczne połączenie:

- skuteczne połączenie projektowania i implementacji,

- pomiary, ocena, udoskonalenie procesów i procedur,

- psychologia uczestników projektu (programistów i klientów)

- wpisanie czynnika ludzkiego: zarówno klient musi być zadowolony (XP – klient partnerem w procesie wytwarzania), jak i programiści – promowanie pracy zespołowej. ALE jednocześnie nadrzędnym celem musi pozostać wysoka jakość produktu.

Bardziej formalnie, modelowanie wymagań? Czy tylko opis słowny? Który level CMMI? Nieważne, ważne są trzy rzeczy:

- wiedzieć co robimy (nadzór procesu)

- wiedzieć co poprawić (udoskonalanie procesu)

- dotrzymywać praktyk, które są uznane za najlepsze w danej sytuacji (utrzymanie procesu)

Dodatkowo:

- automatyzacja procedur, tak aby ich przestrzeganie nie było nudne

- przyrostowe wdrażanie metodyki – nie naraz, krok po kroku!

Metodyki

Jakiej metodyki byśmy nie wybrali, składa się ona z:

- zbierania wymagań
- planowania projektu (tak, XP też)
- kontroli wersji
- wytwarzania (czasami: sterowanego testami)
- testów integracyjnych
- usuwania błędów

Zbieranie wymagań

Określanie potrzeb klientów:

- programiści **zawsze** mają niepełną wiedzę nt. rozwijanego oprogramowania

- porozmawiaj z klientem! W trakcie rozmowy uświadom mu istnienie szczegółów, o których nie ma pojęcia – spróbuj odkryć dodatkowe wymagania! „bujanie w obłokach”:

    - odgrywanie scenek z podziałem na role (ty jesteś oprogramowaniem)

- przekształcenie zebranych informacji na OPOWIEŚCI UŻYTKOWNIKA:

    - mają być zrozumiałe dla klienta, zaakceptowane lub nawet napisane przez niego i pisane z jego perspektywy

    - mają opisywać JEDNO zadanie

    - krótkie – do 3 zdań

    - tytuł i opis

    - opowieści mówią CO musisz zrobić, teraz pozostaje oszacować czas realizacji każdej opowieści

    - szacunki innych osób, poker planistyczny – 13+ kart: 0, ½, 1, 2, 3, 4, 5, 8, 13, 20, 40, 100, ?, „kawa”

    - dzielenie się założeniami, założenia zapisujemy na odwrocie

    - iteracje powyżej 1 miesiąca są zbyt długie!

Zbieranie wymagań

- wszystko wyszło świetnie – szacowany czas to 458 dni roboczych... klient umiera na zawał serca
- priorytety opowieści! - ale wybierane przez klienta (wielokrotności 10!)
- klient **zawsze** będzie oczekiwał więcej, niż zespół jest w stanie udostępnić
- usunięcie funkcji, które nie są niezbędne
- co jeżeli nadal nie mogę dostarczyć oprogramowania w wymaganym czasie, mimo usunięcia zbędnych opowieści?
    - ZREZYGNUJ
    - zatrudnianie dodatkowych osób

Przydzielanie zadań

- miesięczne iteracje to TEORETYCZNIE 20 dni roboczych:
  - chorobowe
  - usterki sprzętu
  - święta
  - aktualizacje
- podstawowy szacunek: praca zajmie 70% przewidzianego czasu. Dni pracy/szybkość = dni potrzebne na wykonanie zadania
- programiści to optymiści: „spoko, zajmie mi to dwa dni”
- „mityczny osobomiesiąc”
- dodanie iteracji? Obcięcie planów? ZALEŻY – lepiej obiecać mniej i dotrzymać, niż obiecać więcej i zawieść

Przydzielanie zadań

- przydzielanie zadań (NIE OPOWIEŚCI) programistom – może 2 na opowieść? Może niektóre opowieści najpierw?
- Szacowanie zadań – co jeżeli wyjdzie więcej niż opowieść?

Im wcześniej tym lepiej

- tablica pokazująca które zadania są w toku, a które ukończone
- niezaplanowane zadania „znikąd”
- codzienne spotkania „na stojąco”:
    - nad czym pracowałeś wczoraj
    - jakie napotkałeś problemy
    - co będziesz robił dzisiaj

Projekt i inne czynniki

- przy skutecznym rozwoju oprogramowania wiesz, gdzie się znajdujesz

- projekt – dobry projekt pomaga realizować zadania
- DRY – don't repeat yourself

- kontrola wersji – anulowanie błędów, poprawianie usterek, niezależny rozwój
- zarządzanie konfiguracją (buildy: BuildBot, CruiseControl)
- kod może się kompilować, ale program może przestać działać
- testy są ważne: refaktoryzacja
  - TDD. Kod który nie jest potrzebny do tego aby test przeszedł, jest na razie niepotrzebny
- tester ma inne nastawienie niż programista

Koniec

Dziękuję za uwagę

Bibliografia

- Bereza-Jarociński B., Szomański B., *Inżynieria oprogramowania, Gliwice* 2009
- Pilone D., Russ M., *Software Development,* Gliwice 2009