KOMPUTEROWE PRZETWARZANIE WIEDZY

Kolekcja prac 2012/2013
pod redakcją Tomasza Kubika

SPIS TREŚCI

ROZDZIAŁ 1

AUDYT W SYSTEMACH INFORMATYCZNYCH

M. Nowak, G. Maj

Jednym z podstawowych problemów profesjonalnego zarządzania firmą w branży informatycznej jest zarządzanie jakością. Jakość można definiować i mierzyć na wielu poziomach. Bardzo dużo uwagi poświęcane jest jakości produktów, choć równie ważnym elementem jest ocena jakości pracy pracowników oraz ich efektywności. Audyt w systemach informatycznych powinien zawierać również elementy oceny jakości pracy i służyć usprawnieniu działania organizacji.

W rozważaniach na temat oceny jakości pracy i wydajności pracowników skupiono się na firmach informatycznych, których przedmiotem działalności jest wytwarzanie oprogramowania. Kluczowym pracownikiem biorącym udział w tworzeniu końcowego produktu, czyli programu jest programista. Ocena pracy programisty jest zadaniem złożonym i wymaga znajomości wielu aspektów procesu wytwarzania oprogramowania. W dzisiejszych czasach zarządzanie jakością oprogramowania jest prawdziwym wyzwaniem dla dużych firm o zasięgu globalnym zatrudniających tysiące programistów z całego świata.

Podstawowym pytaniem, które postawiono w tym rozdziale jest: jak efektywnie mierzyć jakość i efektywność pracy programisty? W niniejszym artykule autorzy postarają się przybliżyć problem, przedstawić dostępne na rynku rozwiązania oraz opisać prosty system kontroli efektywności pracy.

1.1. Dostępne systemy pomiaru pracy

1.1.1. Systemy kontroli wersji

Przykładami systemów kontroli wersji są najbardziej popularne darmowe systemy takie jak:

- Git,
- Svn,
- Mercurial.

Systemy kontroli wersji stworzone są do pomocy przy rozwoju oprogramowania, ich głównym zadaniem jest łączenie pracy wielu pracowników jednego pro-

jektu, jednak poprzez tworzenie historii zmian w projekcie daje możliwość kontroli i oceny pracy pracowników. Dają one możliwość śledzenia postępu prac poczynionych przez poszczególnych pracowników. Podstawowe informacje, które można uzyskać to:

- jaki użytkownik i kiedy wprowadzał zmiany,
- możliwość sprawdzenia jakie zmiany zostały wprowadzone w danym commicie:
  - dokładne różnice między zmienionymi plikami,
  - skrócone statystyki(ilość zmian bez poszczególnych różnic),
  - sumy kontrolne poszczególnych zmian,
  - wyświetlanie tylko informacji przez nas określonych,
- możliwość dostosowania informacji przez nas śledzonej:
  - kontrola jednego pracownika,
  - kontrola danego okresu zmian,
  - kontrola historii tworzenia i łączenia gałęzi.

Git

Git jest rozproszonym systemem kontroli wersji. Jest to wolne oprogramowanie udostępnione na licencji GNU GPL2. Git powstał jako alternatywa dla zamkniętego systemu BitKeeper w kwietniu 2005 roku do rowijania projektu Linux.

Svn

Subversion jest wolnym syetmem kontroli wersji udostępnionym na licencji Apache. Został stworzony do zastąpienia systemu CVS.

Mercurial

Marcurial podobnie jak Git jest rozproszonym

1.1.2. Systemy zarządzania projektem

- Redmine,
- Jira,
- GitHub.

1.1.3. Systemy do statycznej analizy kodu

- Klocwork (C++),
- Pylint (Python),
- CodeSonar (Java).

1.2. Sposoby pomiaru jakości pracy programisty

Na ocenę pracownika programisty można patrzeć z różnych perspektyw. Z jednej strony, efektem jego pracy jest kod źródłowy, więc podstawowym składnikiem oceny powinna być jego jakość. Z drugiej strony jednak, w każdym projekcie obecne są pewne ograniczenia czasowe, więc szybkość pracy pracownika jest również ważnym aspektem oceny. W ogólności można powiedzieć, że jakość pracy programisty można sprowadzić do dwóch składników:

- oceny jakości kodu,
- oceny organizacji pracy.

Normy dotyczące zarządzania jakością (np. ISO 9000:2008, ISO 15504-4:2005) definiują tzw. kluczowe wskaźniki efektywności (ang. KPI - key performance indicator). Ważną cechą tych wskaźników jest to, że opisują one mierzalne procesy w postaci liczbowej. Daje to menedżerowi obiektywny obraz efektywności danego procesu w przedsiębiorstwie i pozwala na cykliczną kontrolę jakości. Poniżej przedstawiono kilka przykładowych wskaźników, które spełniają warunek mierzalności, więc kwalifikują się do grupy KPI.

1.2.1. Ocena jako?ci wygenerowanego kodu

Jakość kodu jest bardzo szerokim zagadnieniem, które po wielu dekadach intensywnego rozwoju branży IT nie zostało jeszcze dostatecznie zbadane. Wynika to stąd, że na przestrzeni lat zmieniały się trendy w tworzeniu oprogramowania – programiści stosowali różne paradygmaty programowania (w kolejności chronologicznej: programowanie obiektowe, programowanie strukturalne, programowanie funkcjonalne). Dziś dominującym paradygmatem jest programowanie obiektowe, lecz wydaje się, że prawdziwa jest teza, że każda z wymienionych filozofii tworzenia oprogramowania jest użyteczna dla pewnych zastosowań.

Stąd obiektywny pomiar jakościowy kody jest trudny. Niektóre elementy języka mogą być uznawane za wartościowe w jednym podejściu, a drugim absolutnie niedopuszczalne. Istnieją jednak pewne fundamentalne cechy kodu, które są pożądane zawsze, np. zwięzłość kodu.

Wartości służące do pomiaru pewnej własności oprogramowania lub jego specyfikacji nazywane są metrykami oprogramowania. Aby metryka była użyteczna powinna być:

- prosta i możliwa do obliczenia przez komputer,
- przekonująca,
- konsekwentna i obiektywna,
- spójna pod względem użytych jednostek,
- niezależna od języka oprogramowania,
- dająca przydatne informacje[**?**].

Metryki można podzielić na statyczne i dynamiczne. Metryki statyczne łączą się ściśle z analizą statyczną kodu, dziedziną inżynierii oprogramowania zajmującą się badaniem struktury kodu źródłowego. Metryki te najbardziej przydatne

są dla samych programistów i innych osób bezpośrednio zaangażowanych w proces powstawania oprogramowania. Pozwalają na bieżące śledzenie jakości kodu i zwrócenie uwagi na miejsca, które wymagają uproszczenia bądź szczególnie uważnego testowania. Niemniej jednak, programista, który dostarcza kod o dobrych statycznych metrykach może być uznawany za dobrego pracownika.

Metryki dynamiczne to wskaźniki abstrahujące od kodu źródłowego. Badają one zachowanie programu po jego uruchomieniu. Są one mocno związane z wymaganiami klienta, stąd dla samego przedsiębiorcy, są to liczby przydatne bardziej do analizy jakości produktu niż pracy jego pracowników.

Linie kodu (LOC)

Najprostrzą metryką rozmiaru oprogramowania jest liczba linii kodu źródłowego. Mimo jej prostoty, wymaga ona pewnego rozróżnienia, bowiem programy pisane w językach wysokiego poziomu mają znacząco mniej linii kody niż programy pisane np. w assemblerze. Do tej metryki z reguły nie są wliczane również linie zawierające tylko komentarze.

Ważnym aspektem jest tutaj również styl pisania kodu. Poniższe dwa przykłady obrazują dwa różne style programowania w języku C++ obrazujące ten niuans:

```
1  // przyklad 1
2  void bubblesort(std::vector<int>& A)
3  {
4    int temp = 0;
5
6    for (int i = 0; i < A.length()-1; i++)
7    {
8      for (int j = 0; j < A.length()- 1 - i; j++)
9      {
10       if (A[j] > A[j+1])
11       {
12         temp = A[j+1];
13         A[j+1] = A[j];
14         A[j] = temp;
15       }
16     }
17   }
18 }
```

```
1  // przyklad 2
2  void bubblesort(std::vector<int>& A) {
3    for (int i = 0; i < A.length()-1; i++)
4      for (int j = 0; j < A.length()- 1 - i; j++)
5        if (A[j] > A[j+1])
6          std::swap(A[j], A[j+1]);
7  }
```

Oba przykłady są identyczną implementacją popularnego slgorytmu sortowania bąbelkowego. W pierwszym przykładzie stosowane są przeniesienia do następnej linii przy otwieraniu nowych bloków kodu (klamr {}). Oprócz tego, każde wyrażenie warunkowe i pętle są akcentowane nowym otwarciem i zamknięciem klamry, choć składnia języka C++ tego nie wymaga. Dodatkowo, w drugim przykładzie, aby bardziej zwiększyć zwięzłość (i czytelność) kodu, do operacji zamiany wartości dwóch elementów wektra, zastosowano funkcję z biblioteki standardowej, która ma dokładnie takie samo działanie jak kod z przykładu pierwszego.

Linie kodu na plik źródłowy

1.2.2. Ocena efektywności pracy

Problem oceny efektywności pracy programisty nie jest problemem łatwym. Dosyć prosto można zaproponować podstawowe kryteria jednak nie zawsze oddają one faktyczny nakład pracy.

- czas pracy pracownika,
- ilość rozwiązanych zadań,
- efektywność mierzona stosunkiem predyktywności do wykonanych zadań,
- ocena wygenerowanego kodu:
  - ilość wygenerowanego kodu,
  - rozkład kodu w plikach,
  - przejrzystość kodu,
  - jakość testów,
  - jakość dokumentacji,
  - nazewnictwo funkcji i klas,
  - analiza statyczna kodu,

1.3. Proponowane rozwiązanie

Literatura

[1] B. Alberts, D. Bray, A. Johnson, J. Lewis, M. Raff, K. Roberts, P. Walter. *Podstawy biologii komórki. Wprowadzenie do biologii molekularnej.* Wydawnictwo Naukowe PWN, 1999.