Kryterium jakości w procesie projektowania i wytwarzania oprogramowania

Arkadiusz Aptewicz

Hipoteza pracy

Jakość oprogramowania jest silnie determinowana jego użytecznością, funkcjonalnością, niezawodnością oraz ryzykiem projektowym.

Zakres pracy

- Atrybuty ilościowo-wartościowe projektów informatycznych
- Metody zapewnienia jakości projektów informatycznych
- Metody analizy jakości projektów informatycznych
- Zarządzanie jakością w projektach informatycznych
- Koncepcja badania oraz pomiaru jakości
- Koncepcja oceny jakości oprogramowania
- Studium przypadku

Charakterystyka dziedziny problemu

Atrybuty ilościowo wartościowe - pozwalają na porównanie projektów informatycznych przy użyciu zdefiniowanych wielkości/parametrów.

Metody zapewnienia jakości:

- Dom jakości
- Planowanie eksperymentów
- Metoda FMEA jako analiza projakościowa

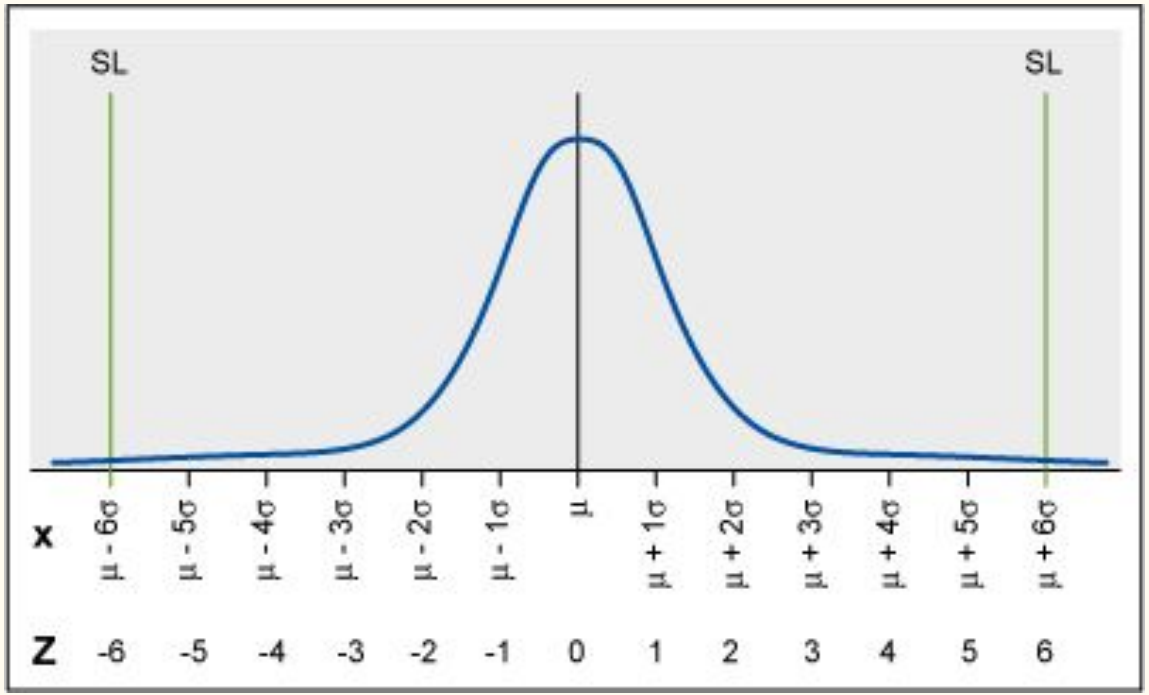
Metody analizy jakości projektów informatycznych:

- Statyczna analiza kodu źródłowego
- Testy jednostkowe i integracyjne

Charakterystyka dziedziny problemu c.d.

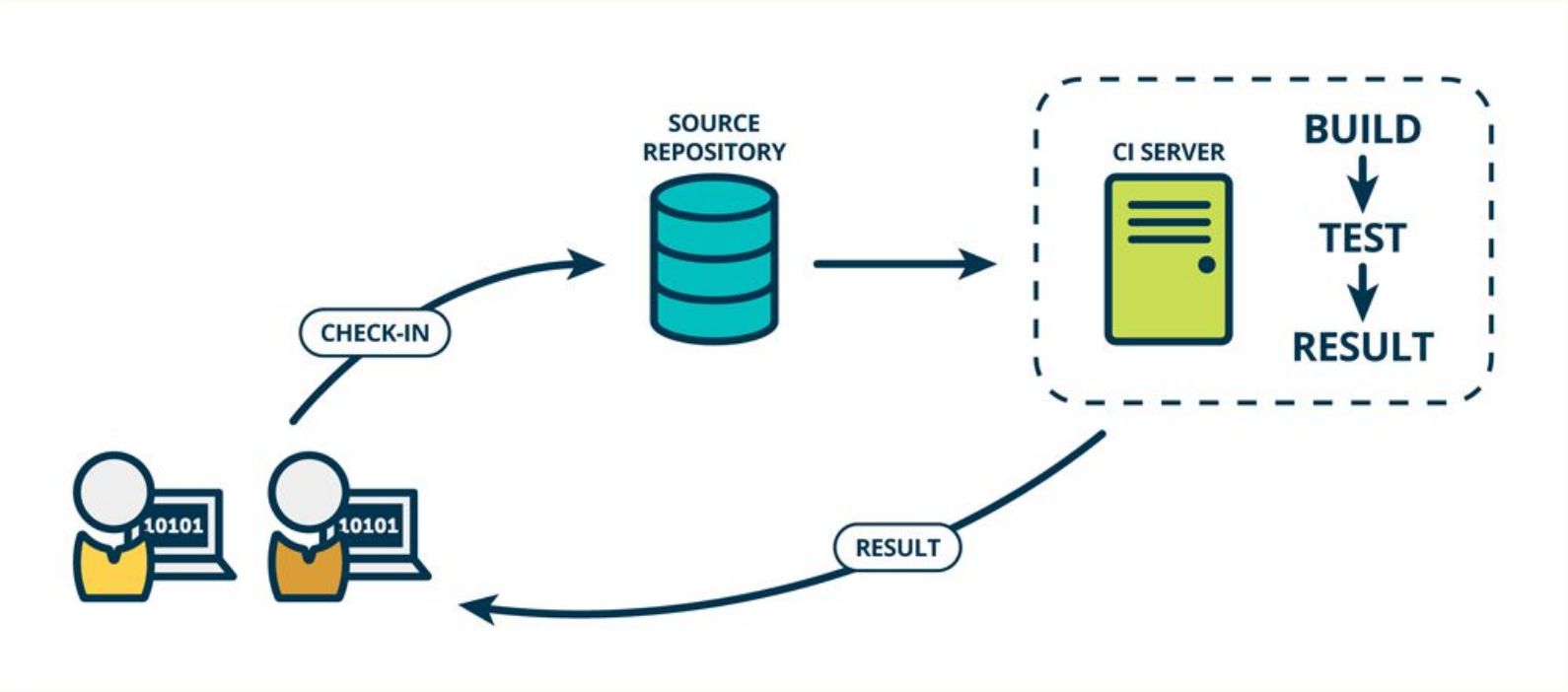
Zarządzanie jakością w projektach informatycznych:

- TQM
- Six Sigma
- Normy ISO 9000



Modele i koncepcje rozwiązań

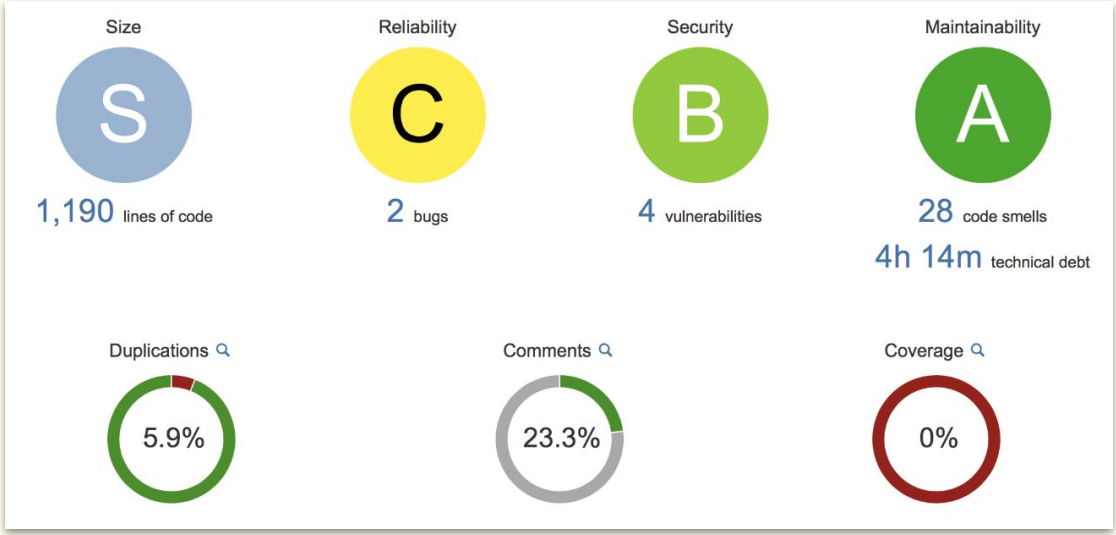
CI (*ang. Continuous Integration*) - ciągła integracja.



Modele i koncepcje rozwiązań c.d.

Koncepcja badania oraz pomiaru jakości:

- wyniki testów jednostkowych
- pokrycie kodu źródłowego testami
- dług technologiczny



Modele i koncepcje rozwiązań c.d.

Koncepcja oceny jakości oprogramowania w oparciu o metodę Six Sigma na podstawie:

- wyników testów jednostkowych;
- pokrycia kodu źródłowego przez testy jednostkowe;
- długu technologicznego;
- rezultatów procesu budowy projektu.

Ocena jakości w oparciu o wyniki testów jednostkowych

Założenia:

- każdy test jednostkowy jest możliwością wykrycia defektu;
- negatywny wynik testu jednostkowego jest interpretowany jako błąd.

Wydajność procesu:

$$(tt — tf)/tt \cdot 100\%$$

gdzie:
tt- ilość wykonanych testów jednostkowych;
tf- ilość testów zakończonych niepowodzeniem.

Ocena jakości na podstawie pokrycia kodu źródłowego przez testy jednostkowe

Założenia:

- każdy proces budowy jest możliwością wprowadzenia defektu;
- defekt jest rozumiany jako zbyt niskie pokrycie kodu źródłowego.

Poziom Sigma procesu:

$$Z = (bt - bf)/bt$$

gdzie:
Z- poziom Sigma procesu;
bt- ilość zakończonych procesów budowy;
bf- ilość procesów budowy charakteryzujących się zbyt niskim pokryciem kodu źródłowego.

Ocena jakości na podstawie długu technologicznego oraz rezultatów procesu budowy

Założenia:

- każdy proces budowy jest możliwością wprowadzenia defektu;
- defekt jest rozumiany jako zbyt wysoka wartość długu technologicznego (w założonych granicach tolerancji);
- każdy proces budowy skutkować może defektem;
- sam proces budowy zakończony niepowodzeniem jest traktowany jako defekt.

Studium przypadku

Charakterystyka projektu - projekt aplikacji mobilnej dla pracowników firmy kurierskiej.

Etapy procesu ciągłej integracji:

- kompilacja kodu źródłowego projektu;
- wykonanie zdefiniowanych testów jednostkowych;
- przeprowadzenie statycznej analizy kodu źródłowego;
- wdrożenie nowej wersji aplikacji (dotyczy tylko gałęzi produkcyjnej kodu źródłowego).

Studium przypadku - wyniki

czas trwania projektu (wyrażony w tygodniach)	ilość zakończonych procesów kompilacji kodu źródłowego	ilość błędnie zakończonych procesów kompilacji	wydajność procesu	poziom Sigma procesu
1	250	1	99,6000%	~4,15
2	500	3	99,4000%	~4,05
4	1000	4	99,6000%	~4,15
8	2000	5	99,7500%	~4,30
12	3000	6	99,8000%	~4,40
16	4000	7	99,8250%	~4,45
20	5000	7	99,8600%	~4,50
24	6000	7	99,8833%	~4,55

Przykład wyznaczania poziomu Sigma procesu na podstawie rezultatów kompilacji kodu źródłowego.

Studium przypadku - wyniki

czas trwania projektu (wyrażony w tygodniach)	ilość zdefiniowanych testów jednostkowych	ilość procesów budowy podczas których zostały wykonane testy jednostkowe	ilość wykonanych testów jednostkowych	ilość negatywnie zakończonych testów jednostkowych	wydajność procesu	poziom Sigma procesu
1	25	249	6225	6	99,9036%	~4,60
2	50	497	24850	30	99,8793%	~4,55
4	100	996	99600	129	99,8705%	~4,50
8	200	1995	399000	528	99,8677%	~4,50
12	300	2994	898200	1426	99,8412%	~4,45
16	400	3993	1597200	3023	99,8107%	~4,40
20	500	4993	2496500	5519	99,7789%	~4,35
24	600	5993	3595800	9114	99,7465%	~4,35

Przykład wyznaczania poziomu Sigma procesu na podstawie wyników testów jednostkowych.

Studium przypadku - wyniki

czas trwania projektu (wyrażony w tygodniach)	ilość procesów budowy podczas których została przeprowadzona statyczna analiza kodu źródłowego	ilość procesów budowy ze zbyt niskim pokryciem kodu źródłowego testami	wydajność procesu	poziom Sigma procesu
1	249	0	100,0000%	6
2	497	1	99,7988%	~4,40
4	996	3	99,6988%	~4,25
8	1995	3	99,8496%	~4,50
12	2994	3	99,8998%	~4,60
16	3993	5	99,8748%	~4,50
20	4993	6	99,8798%	~4,50
24	5993	6	99,8999%	~4,60

Przykład wyznaczania poziomu Sigma procesu na podstawie pokrycia kodu źródłowego testami.

Studium przypadku - wyniki

czas trwania projektu (wyrażony w tygodniach)	ilość przeprowadzonych statycznych analiz kodu źródłowego	ilość błędów/defektów	wydajność procesu	poziom Sigma procesu
1	249	0	100,0000%	6
2	497	1	99,7988%	~4,40
4	996	1	99,8996%	~4,60
8	1995	1	99,9499%	~4,80
12	2994	3	99,8998%	~4,60
16	3993	3	99,9249%	~4,70
20	4993	3	99,9399%	~4,70
24	5993	4	99,9333%	~4,70

Przykład wyznaczania poziomu Sigma procesu wytwarzania oprogramowania na podstawie długu technologicznego.

Studium przypadku - wyniki

czas trwania projektu (wyrażony w tygodniach)	ilość wykonanych procesów budowy	ilość procesów budowy zakończonych niepowodzeniem	wydajność procesu	poziom Sigma procesu
1	250	1	99,6000%	~4,15
2	500	5	99,0000%	~3,85
4	1000	8	99,2000%	~3,95
8	2000	9	99,5500%	~4,15
12	3000	12	99,6000%	~4,15
16	4000	15	99,6250%	~4,15
20	5000	16	99,6800%	~4,20
24	6000	17	99,7167%	~4,25

Przykład wyznaczania poziomu Sigma procesu na podstawie rezultatów procesów budowy aplikacji.

Wnioski z przykładu praktycznego

- Podstawą do zapewnienia jakości projektu informatycznego jest poprawna kompilacja kodu źródłowego;
- spełnienie przez oprogramowanie zdefiniowanego zbioru testów znacząco podnosi jego jakość;
- pokrycie kodu źródłowego przez testy jednostkowe jest dodatkowym parametrem pozwalającym na podnoszenie jakości oprogramowania;
- utrzymywanie długu technologicznego na niskim poziomie wpływa na zwiększenie poziomu jakości;
- analizy i zestawienia mogą być docelowo dobrym rozwinięciem do planowania jakości na etapie budowy domu jakości i specyfikacji wymagań użytkownika oraz parametrów techniczno-technologicznych postrzeganych przez projektanta.

Uogólnione wnioski

- Metoda Six Sigma jest metodą komplementarną do metody domu jakości;
- przedstawiona koncepcja ma zastosowanie dla projektów wytwarzanych w podejściu iteracyjnym;
- stosowanie serwerów ciągłej integracji znacząco ułatwia kontrolę jakości oraz zarządzanie jej poziomem;
- wysoka dostępność rozbudowanych środowisk programistycznych sprawia, że wprowadzenie zaproponowanych metod zapewnienia jakości nie jest szczególnie absorbujące;
- zaproponowana koncepcja może być z powodzeniem stosowana we współcześnie realizowanych projektach informatycznych w różnych obszarach implementacyjnych;
- przedstawiony w pracy zestaw metod, technik i narzędzi zapewnienia jakości oprogramowania jest zbiorem komplementarnych możliwości i rozwiązań.