



# Metryki w procesie testowym – jak wykorzystać TestLink i JIRA jako źródła danych do wyliczania metryk

---

**Autor:** Piotr Kijewski

**O autorze:**



*Piotr Kijewski Jest absolwentem Wyższej Szkoły Informatyki Stosowanej i Zarządzania na wydziale informatyki o profilu bazy danych. Od ponad 6 lat związany zawodowo z obszarem zapewnienia jakości oprogramowania. Karierę w branży rozpoczął jako tester, był kierownikiem testów a potem menedżerem zapewnienia jakości. Pracował również jako administrator systemów unixowych, a także ekspert w zakresie relacyjnych baz danych. Obecnie zatrudniony w PZU S.A. i PZU Życie S.A. na stanowisku menedżera zapewnienia jakości w dziale rozwoju aplikacji biznesowych. Zajmuje się monitorowaniem i optymalizacją procesu testowego a także wdrażaniem nowych rozwiązań i standardów w zakresie kontroli i zapewnienia jakości.*

*Członek Stowarzyszenia Jakości Systemów Informatycznych. Jako przewodniczący komisji egzaminacyjnej SJSI, zorganizował i uruchomił w 2007 roku proces egzaminowania i certyfikacji programu ISTQB Certified Tester w języku polskim. Posiada certyfikaty ISTQB Advanced - Test Manager.*

## Intermediate

Level

4

Magazine Number

## Testowanie oprogramowania

Section in the magazine

## Wprowadzenie

Metryki mogą ułatwić i przyspieszyć ocenę stanu procesu testowego zarówno na etapie projektowania jak i wykonania testów. Jeśli ponadto metryki te mogą być cyklicznie generowane automatycznie na podstawie danych i tak rejestrowanych już w procesie testowym, unikamy dodatkowego narzutu pracy na ten monitoring

## 1. Miary a metryki

### 1.1 Kilka wybranych definicji

Cytat z „Certified Tester Advanced Syllabus”:

*A variety of metrics (numbers) and measures (trends, graphs, etc) should be applied throughout the software development life cycle (e.g. planning, coverage, workload, etc). In each case a baseline must be defined, and then progress tracked with relation to this baseline. (...)*

*Metryka oprogramowania – miara pewnej własności oprogramowania lub jego specyfikacji. Termin ten nie ma precyzyjnej definicji i może oznaczać właściwie dowolną wartość liczbową charakteryzującą oprogramowanie. Standard IEEE 1061-1998 określa metrykę jako funkcję odwzorowującą jednostkę oprogramowania w wartość liczbową. Ta wyliczona wartość jest interpretowalna jako stopień spełnienia pewnej własności jakości jednostki oprogramowania. (...)*

źródło: [http://pl.wikipedia.org/wiki/Metryka\\_oprogramowania](http://pl.wikipedia.org/wiki/Metryka_oprogramowania)

*Metric - A quantitative measure of the degree to which a system, component, or process possesses a given attribute [2]. A calculated or composite indicator based upon two or more measures. A quantified measure of the degree to which a system, component, or process possesses a given attribute [3]*

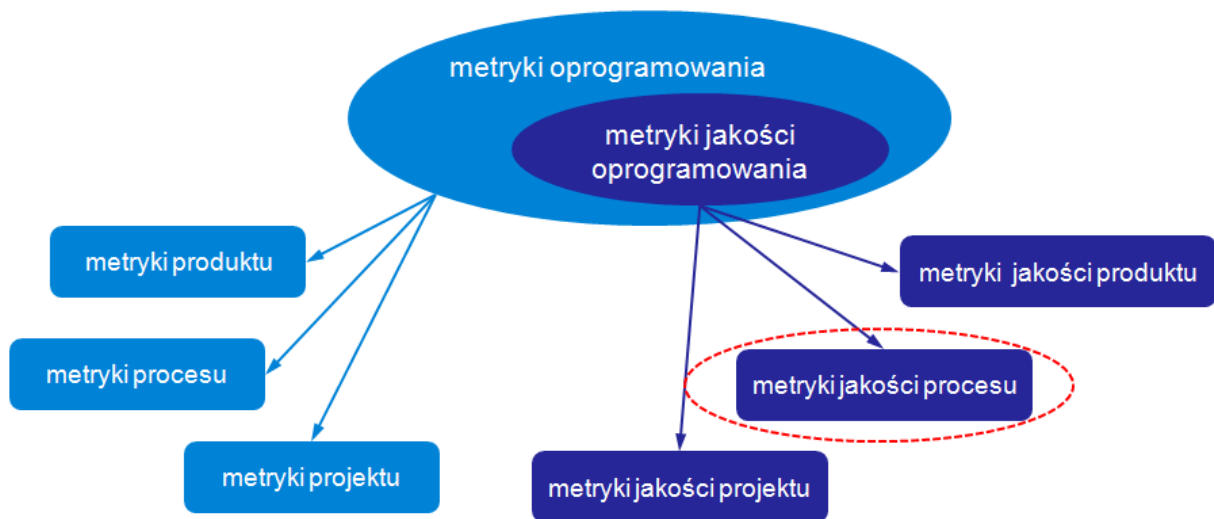
źródło: <http://www.stsc.hill.af.mil/crosstalk/1995/03/Measure.asp>

### 1.2 Podział metryk

Źródło: książka autorstwa Stephena H. Kan'a: „Metryki i modele w inżynierii jakości oprogramowania” (2006 rok):

Według autora, podstawowe miary to na przykład:

- stosunek,
- proporcja,
- odsetek.



Rysunek 1 Podział metryk oprogramowania

### 1.3 Dlaczego stosować metryki

**“You can’t control what you can’t measure”** – Tom DeMarco

Za pomocą metryk można śledzić status procesu testowego i jego zmiany w czasie, a także oceniać jego efektywność i planować jego optymalizację.

W przypadku metryk procesu testowego, za pomocą metryk możemy kontrolować przykładowo:

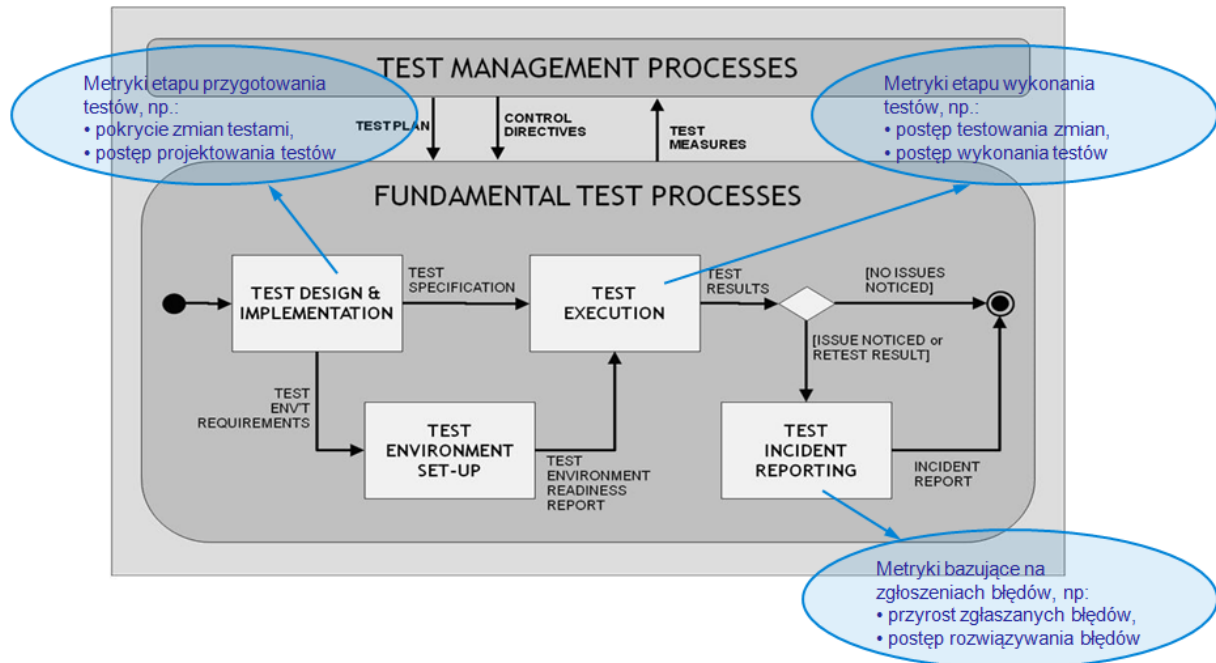
- postęp przygotowania testów,
- stopień pokrycia zmian testami,
- postęp wykonania testów,
- postęp testowania zmian,
- przyrost liczby błędów w podziale na fazy/iteracje,
- szybkość rozwiązywania błędów,
- itp.

Ponadto:

1. Publikacja i odpowiednia prezentacja metryk ułatwia komunikację z udziałowcami projektu a przede wszystkim z:
  - programistami,
  - kierownikami projektów,
  - zamawiającym (klientem)
 i pozwala im na szybką ocenę postępu/stanu testowania i aktualnej jakości SUT.
2. Posługując się analizą zmian wartości metryk w czasie (ich szeregów czasowych) kierownik testów jest w stanie szybko zidentyfikować potencjalne problemy w przebiegu procesu, np.: wykonanie testów niezgodnie z ich priorytetami, zbyt małe tempo rozwiązywania błędów przez programistów w stosunku do tempa ich zgłaszania, itp.
3. Porównanie wartości metryk z kilku projektów umożliwi ocenę czy jakość procesu wzrasta czy nie.

## 1.4 Metryki w procesie testowym

Poniższy diagram obrazuje na jakich etapach procesu testowego można stosować jakie typy metryk. Przedstawiony na diagramie proces testowy pochodzi z będącej wciąż w przygotowaniu normy „ISO/IEC 29119 Fundamental Test Process”.



Rysunek 2 Metryki w procesie testowym

## 2. Jak wybierać metryki

### 2.1 Źródła danych do metryk

Dane do metryk mogą być:

- kolekcjonowane z ankiet, wypełnianych przez uczestników procesu,
- pozyskiwane automatycznie z narzędzi używanych w procesie

Należy dążyć do minimalizacji czasu poświęcanego na gromadzenie danych do metryk, aby nie obciążać uczestników procesu (testerów, programistów, itp.) dodatkową pracą związaną z monitorowaniem.

W przypadku procesu testowego, dane do metryk można pozyskiwać z baz danych narzędzi służących do:

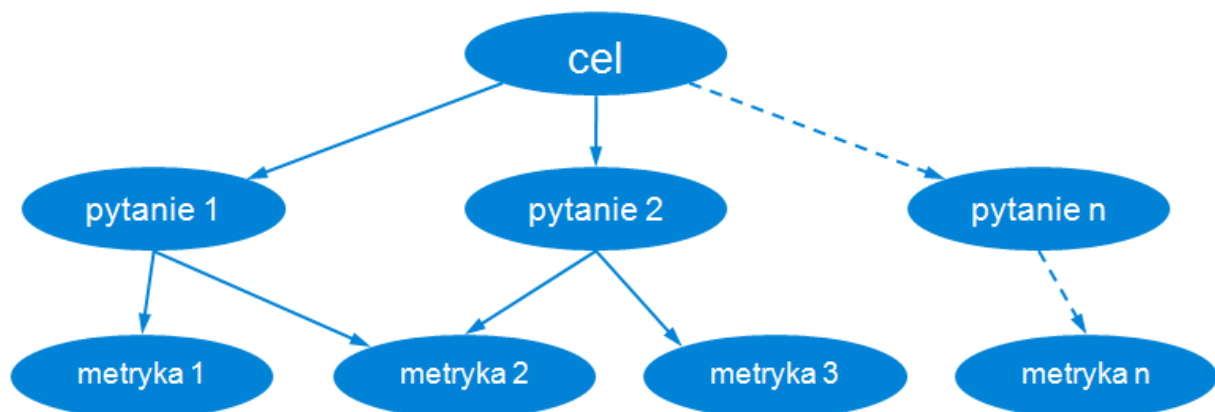
- zarządzania testami (np.: TestLink, SpiraTest, Testopia, QATraq, itp.: <http://www.opensourcetesting.org/testmgt.php>)
- zarządzania wymaganiami,
- zarządzanie zgłoszeniami błędów (np.: JIRA, Mantis, Bugzilla, itp.: <http://www.opensourcetesting.org/bugdb.php>)

### 2.2 Metoda GQM

**GQM** = Goal Question Metric – jest to zaproponowane przez Victora Basil'a i Davida Weiss'a systematyczne podejście do wyboru, projektowania i stosowania metryk oprogramowania. Obszerne informacje na ten temat można znaleźć np. pod adresem:

<http://www.gqm.nl>

Podstawową zasadą GQM jest to, że pomiary powinny być zorientowane na konkretne cele. Chcąc zoptymalizować proces, należy w pierwszej kolejności zdefiniować cele pomiarów, bazujące na celach organizacji/projektu. Następnie należy przełożyć te cele na czynności, które mogą podlegać pomiarom podczas realizacji procesu/projektu.



Rysunek 3 Metoda GQM

Przykład zastosowania:

**Cel:** Poprawienie jakości oprogramowania trafiającego na testy akceptacyjne

**Pytanie 1:** Czy testy systemowe wykrywają wystarczającą liczbę błędów?

**Pytanie 2:** Czy pokrycie zmian testami jest wystarczające?

**Pytanie 3:** W jakim zakresie realizowane są testy regresji?

**Metryka 1:** Pokrycie zmian testami

**Metryka 2:** Skuteczność wykrywania błędów (DDE)

$$DDE = \frac{w_{lzts}}{w_{lzts} + w_{lzuat} + w_{lzwdr}} * 100\%$$

gdzie:

**w<sub>lzts</sub>** - ważona liczba zgłoszeń z testów systemowych

**w<sub>lzuat</sub>** - ważona liczba zgłoszeń z testów akceptacyjnych

**w<sub>lzwdr</sub>** - ważona liczba zgłoszeń po wdrożeniu produkcyjnym

**Ważona** oznacza, że liczba zgłoszeń o danym priorytecie (np. niski, normalny, krytyczny, itp) jest mnożona przez wagę przypisaną temu priorytetowi. Ważona liczba zgłoszeń to suma takich iloczynów dla wszystkich priorytetów w jakich są zarejestrowane zgłoszenia.

### 3. Definiowanie metryk

Każda metryka powinna posiadać swoją definicję. W niniejszym rozdziale zostaną opisane elementy definicji metryki wraz z przykładami.

#### 3.1 Nazwa i kod

Znacząca nazwa oraz w miarę krótki, unikalny kod, ułatwiający identyfikację metryki, np.:

*PPT – Postęp Przygotowania Testów*

#### 3.2 Przeznaczenie

Krótki opis przeznaczenia metryki, co ona oznacza, do czego służy, czym się charakteryzuje i inne ważne informacje dotyczące metryki, np.:

*Metryka pozwala śledzić postęp projektowania przypadków testowych w stosunku do planu. Jej wartość wyraża procent liczby testów gotowych do wykonania.*

#### 3.3 Algorytm wyliczania

Matematyczne równanie lub też algorytm zastosowany do wyliczania wartości metryki. Ważne jest dodawanie legendy, zawierającej objaśnienia symboli zastosowanych w algorytmie/równaniu, np.:

$$PPT = \frac{^{1}tpz}{slt} * 100\%$$

gdzie:

**ltpz** liczba testów pokrywających zmiany (wszystkie testy, które są powiązane przynajmniej z jedną zmianą i są oznaczone słowem kluczowym "GOTOWY")

**slt** szacowana liczba wszystkich testów do zaprojektowania (liczba oszacowana na etapie planowania testów na podstawie analizy zakresu zmian przeznaczonych do realizacji w wydaniu)

#### 3.4 Źródła danych

Skąd pobierane są dane do algorytmu wyliczania wartości metryki? Czy zasilamy z baz danych narzędzi i jakich, czy też z wypełnianych manualnie ankiet i jakich. Jeśli z baz danych, to jakimi zapytaniami SQL są wybierane te dane, np.:

*Dane do wyliczenia metryki są pozyskiwane bezpośrednio z bazy danych TestLinka za pomocą specjalnie w tym celu przygotowanych zapytań SQL.*

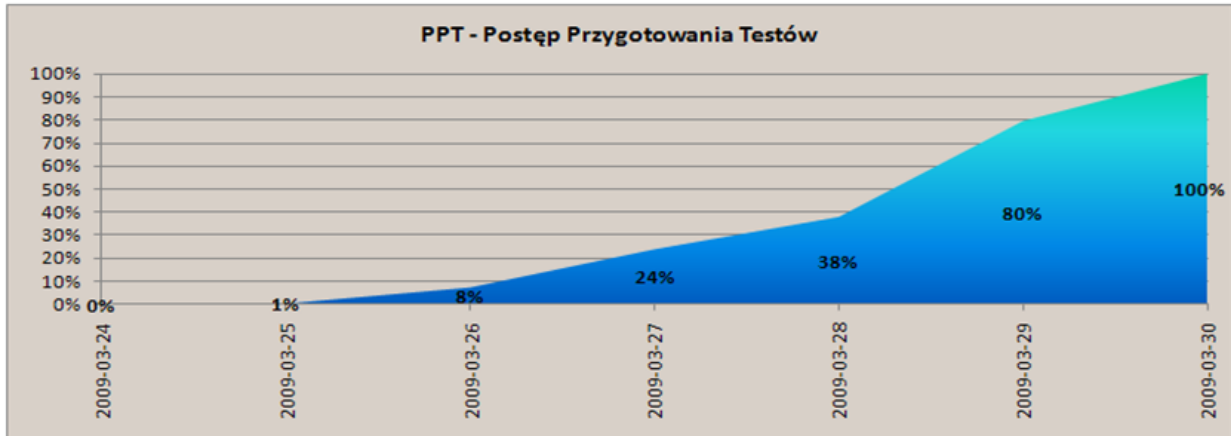
*PPT\_run\_once\_v1.sql Plik zawierający zapytania inicjalnie tworzące procedurę składowaną wybierającą dane do metryki oraz tabele z parametrami i na dane. Należy go uruchomić tylko raz bezpośrednio na bazie danych TestLinka.*

*PPT\_run\_daily\_v1.sql Plik zawierający zapytania uruchamiające procedurę zbierania danych do metryki oraz wyładowujące dane z tablicy wynikowej do pliku. Zapytania zawarte w tym pliku należy cyklicznie uruchamiać bezpośrednio na bazie danych TestLinka.*

### 3.5 Sposób prezentacji

W jaki sposób będą prezentowane wartości metryki: jako wartości liczbowe, wykres (kołowy, słupkowy, itp.), szereg czasowy, itp. Jaka będzie częstotliwość ich aktualizacji, np.:

*Wartości metryki są aktualizowane raz na dzień o godzinie 18:00 i prezentowane w postaci wykresu szeregu czasowego*



Rysunek 4 Szereg czasowy zmian wartości metryki PPT - przykład

### 3.6 Poziomy kontrolne

Poziomy kontrolne są konieczne aby:

- Kontrolować czy wartości metryk nie przekraczają ustalonych poziomów minimalnych / maksymalnych (np. pokrycie zmian testami, średni czas rozwiązywania błędu, itp.).
- Monitorować zaawansowanie realizacji celów projektu (etapu projektu) poprzez osiągnięcie przez metryki poziomów nominalnych (np.: 100%). Przykładami mogą być: postęp projektowania testów, postęp wykonania testów, itp.
- Optymalizować proces. Docelowy poziom kontrolny ustalany jest w wyniku analizy wartości uzyskiwanych w poprzednich projektach i na tej podstawie podnoszona jest poprzeczka (np. DDE).
- Benchmarkować proces testowy/wytwórczy, czyli porównywać osiągnięte wartości metryk ze średnimi uzyskiwanymi przez innych (np. DDE).

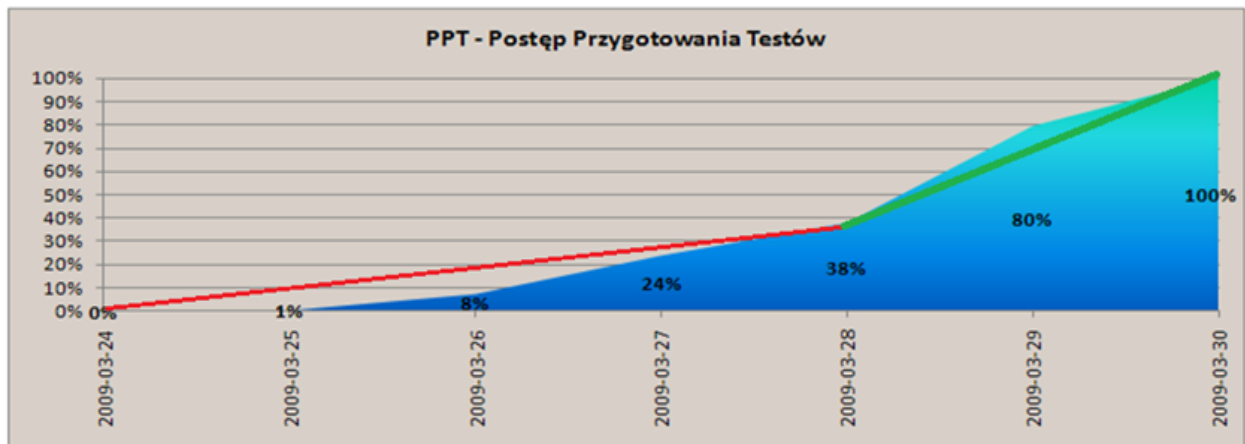
Przykład dla metryki PPT:

*poziom kontrolny = 100% (oznacza ukończenie projektowania testów)*

### 3.7 Interpretacja

Oprócz opisu przeznaczenia, każda metryka powinna mieć opisany sposób interpretacji jej wartości. Interpretować można pojedyncze wartości metryki jak również ich zmienność w czasie (trendy na wykresach szeregów czasowych). Dla niektórych metryk określone kształty trendów mogą być kluczowe do oceny stanu procesu testowego i identyfikacji potencjalnych problemów. Opis sposobu interpretacji metryki pomaga w wyciąganiu właściwych wniosków z analizy jej wartości, np.:

*Dla metryki PPT, kąt nachylenia linii trendu na wykresie szeregu czasowego jest uzależniony od tempa projektowania testów: im większy kąt nachylenia (kolor zielony), tym tempo szybsze.*



Rysunek 5 Szereg czasowy zmian wartości metryki PPT wraz z liniami trendu - przykład

### 3.8 Warunki stosowania

Często zdarza się, że aby rozpocząć stosowanie danej metryki, trzeba spełnić określone warunki, tak organizacyjne jak i techniczne. Warunki organizacyjne staramy się minimalizować, ale mogą nimi być:

- konieczność zmiany sposobu rejestrowania danych w narzędziach używanych w procesie,
- konieczność cyklicznego wypełniania ankiet.

Warunkami technicznymi może być np. specyficzna konfiguracja aplikacji narzędziowych lub też pozyskanie narzędzi do prezentacji wartości metryki. Przykład dla metryki PPT:

*Aby móc monitorować postępowanie projektowania testów, konieczne jest rejestrowanie przez testerów w TestLinku dodatkowo następujących danych:*

- liczby testów, jaka jest zaplanowana do zaprojektowania,
- słowa kluczowego „GOTOWY” przy każdym przypadku testowym, którego projektowanie zostało zakończone

## 4. Generowanie i prezentacja wartości metryk

### 4.1 Mechanizm generowania metryk

Aby zminimalizować nakład pracy na monitorowanie procesu, wyliczanie i prezentacja metryk powinny być jak najbardziej zautomatyzowane. Mechanizm generowania metryk powinien automatyzować jak najwięcej czynności związanych z ich wyliczaniem oraz okresową aktualizacją. Powinien:

- pobierać dane z narzędzi do zarządzania testami, wymaganiami, zgłoszeniami błędów,
- obliczać wartości metryk zgodnie ze zdefiniowanymi algorytmami,
- generować warstwę prezentacyjną metryk (wykresy, szeregi czasowe),
- okresowo odświeżać wartości metryk.

#### 4.1.1 Kolekcjonowanie danych

Dane do kalkulacji wartości metryk mogą być pobierane:

- za pomocą API narzędzi używanych w procesie testowym,
- bezpośrednio z baz danych tych narzędzi,
- z ankiet wypełnianych przez uczestników procesu,

Zarówno TestLink jak i JIRA posiadają API pozwalające na wymianę danych w ograniczonym zakresie. Dlatego w PZU pobieramy dane bezpośrednio z baz danych.

#### 4.1.2 Kalkulacja wartości metryk

Zarówno pobieranie danych jak i kalkulacja wartości metryk realizowane są przez procedury składowane SQL utworzone w bazach danych TestLink i JIRA. Uruchomienie procedury powoduje: wybranie odpowiednich danych, wyliczenie na ich podstawie wartości metryk oraz zachowanie wyników w strukturze bazy danych (w dodatkowych tablicach).

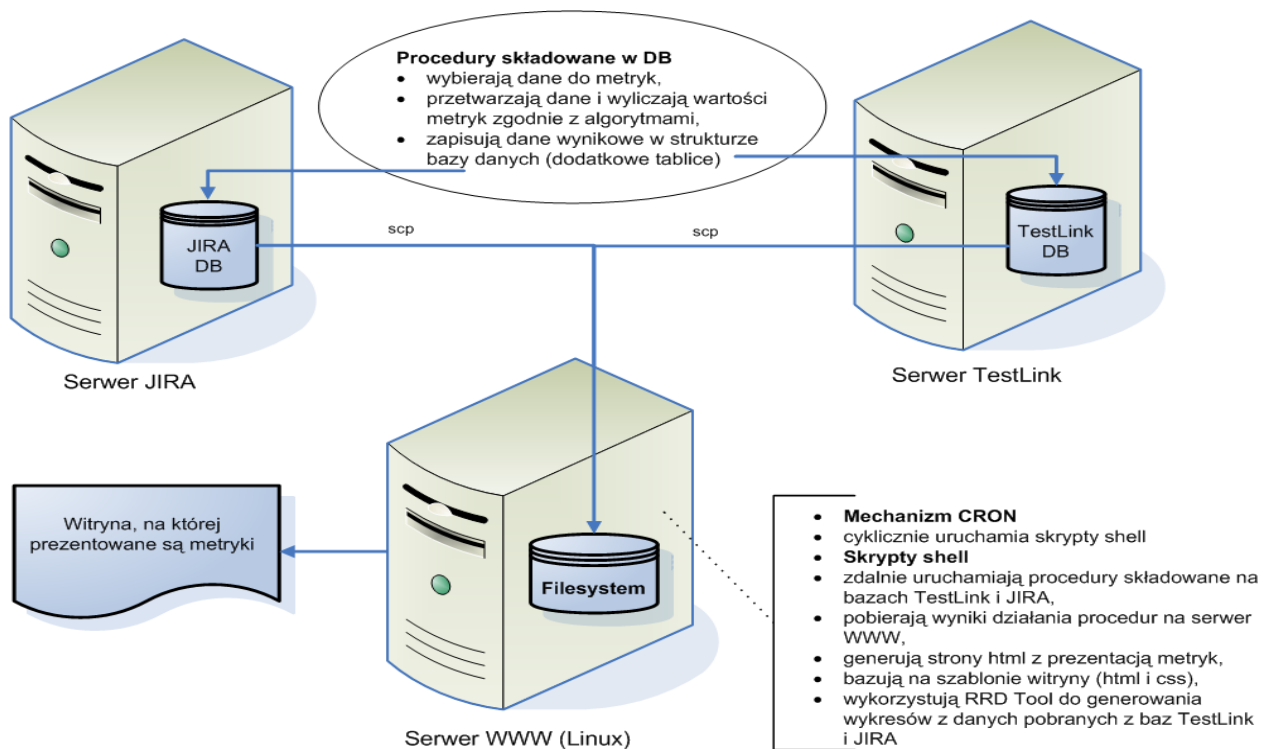
#### 4.1.3 Okresowe odświeżanie wartości metryk

Procedury składowane uruchamiane są cyklicznie za pomocą skryptów shell z częstotliwością zapisaną w CRONie. Skrypty shell wyładowują również z baz wyniki działania procedur, przesyłają na serwer WWW i generują warstwę prezentacyjną.

#### 4.1.4 Warstwa prezentacyjna

Wartości metryk publikowane są na witrynie www. Strony html generowane są automatycznie przez skrypt shell. Wykresy metryk generowane są również przez skrypt shell, z wykorzystaniem open-source'owego narzędzia RRD Tool, autorstwa Tobiego Oetikera, źródło: <http://oss.oetiker.ch/rrdtool/>

## 4.1.5 Mechanizm stosowany w PZU



Rysunek 6 Schemat stosowanego w PZU mechanizmu generowania metryk procesu testowego

## 4.2 Sposób prezentacji

Sposób prezentacji metryk stosowanym w PZU to:

- prosta strona HTML,
- na podstawie przygotowanego szablonu,
- formatowana CSS,
- z zawartością zależną od listy aplikacji objętych monitorowaniem,
- z wykresami generowanymi za pomocą RRD Tool.

RRD = Round Robin Database, to opensource'owe narzędzie autorstwa Tobiego Oetikera:

*RRDtool is the OpenSource industry standard, high performance data logging and graphing system for time series data. Use it to write your custom monitoring shell scripts or create whole applications using its Perl, Python, Ruby, TCL or PHP bindings.*

źródło: <http://oss.oetiker.ch/rrdtool/>

Wygląd szablonów stron html, na których prezentowane są metryki zostały zaprezentowane na zrzutach ekranów:

Ponieważ metryki procesu testowego generowane są w oparciu o bazy danych dwóch różnych narzędzi: TestLink i JIRA, prezentowane są one oddzielnie ze względu na brak możliwości połączenia nazwy projektu testowego w TestLink z nazwą projektu (lub systemu) w JIRA.

### Wybierz projekt z TestLink lub z JIRA:

Projekty testowe z TestLink	Systemy obsługiwane z JIRA
<a href="#">BOM</a> <a href="#">CBA</a> <a href="#">CBS</a> <a href="#">Compass PZUZ</a> <a href="#">eRenty</a> <a href="#">RPT</a> <a href="#">SLS_Systemowe</a>	<a href="#">cba</a> <a href="#">cbki</a> <a href="#">cbs</a> <a href="#">comw2</a> <a href="#">crsz</a> <a href="#">ecash</a> <a href="#">erenty</a> <a href="#">erudo</a> <a href="#">eska</a> <a href="#">irenty</a> <a href="#">rea</a> <a href="#">rpiu</a> <a href="#">rpt</a> <a href="#">selektor2</a> <a href="#">sls</a> <a href="#">symulator</a>

dostępne archiwa z dni: [2009-06-01](#) | [2009-06-02](#) | [2009-06-03](#) | [2009-06-04](#) | [2009-06-05](#) | [2009-06-08](#) | [2009-06-09](#) | [2009-06-10](#) | [2009-06-11](#) | [2009-06-12](#) | [2009-06-15](#) | [2009-06-16](#) | [2009-06-17](#) | [2009-06-18](#) | [2009-06-19](#) | [2009-06-22](#) | [2009-06-23](#) | [2009-06-24](#) | [2009-06-25](#) | [2009-06-26](#) | [2009-06-27](#) | [2009-06-28](#) | [2009-06-29](#) | [2009-06-30](#) | [2009-07-01](#) | [2009-07-02](#) | [2009-07-03](#) | [2009-07-06](#) | [2009-07-06](#) | [2009-07-07](#) | [2009-07-07](#) | [2009-07-08](#) | [2009-07-09](#) | [2009-07-10](#) | [2009-07-13](#) | [2009-07-14](#) | [2009-07-15](#) | [2009-07-16](#) | [2009-07-17](#) | [2009-07-20](#)

Rysunek 7 Witryna z prezentacją metryk – strona główna

### Projekt testowy: SLS\_Systemowe

#### Metryki na podstawie danych z TestLink

[PZT - Pokrycie Zmian Testami](#)  
[PPT - Postęp Przygotowania Testow](#)  
[PWT - Postęp Wykonania Testow](#)  
[PTZ - Postęp Testowania Zmian](#)  
[PZB - Przyrost Zgłaszanych Bledow](#)

Rysunek 8 Witryna z prezentacją metryk – metryki z TestLink

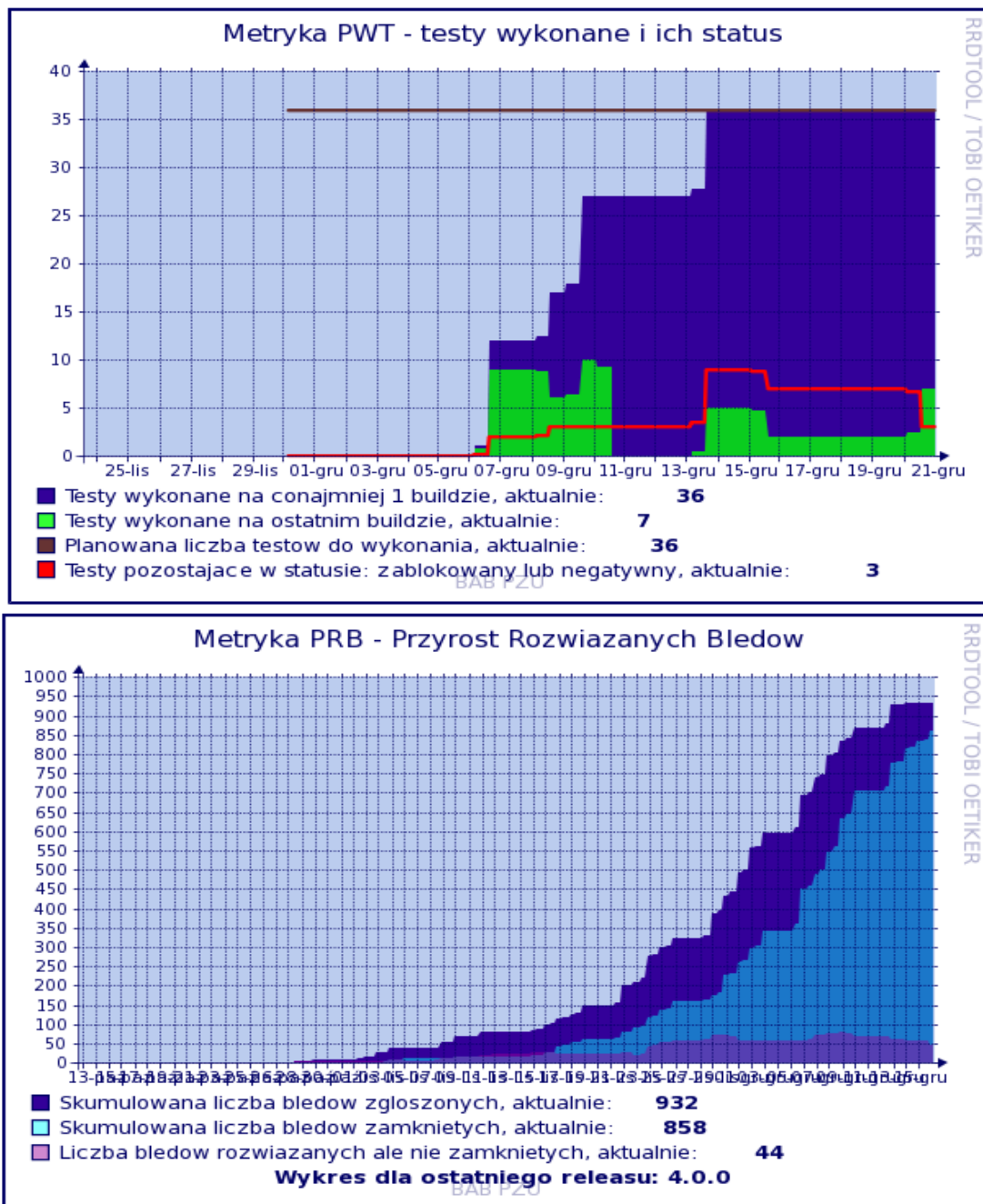
### Projekt testowy: sls

#### Metryki na podstawie danych z JIRA

[PRB - Przyrost Rozwiązanych Bledow](#)  
[DDE - Efektywnosc Wykrywania Bledow](#)

Rysunek 9 Witryna z prezentacją metryk – metryki z JIRA

Przykładowy wygląd wykresów generowanych za pomocą narzędzia RRD Tool, został zaprezentowany na poniższych zrzutach ekranów:



Rysunek 10 Przykładowe wykresy metryk

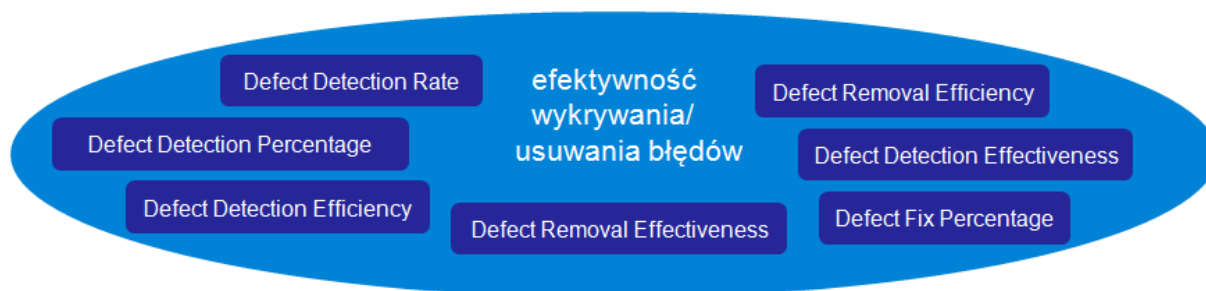
## 5. Propozycje metryk z JIRA

### 5.1 DDE – Skuteczność Wykrywania Błędów

#### 5.1.1 Nazwa i kod

DDE – Efektywność Wykrywania Błędów

Metryka jest szeroko stosowana w wielu firmach i znana pod różnymi nazwami, np.:



Rysunek 11 Różne nazwy metryki dotyczącej skuteczności wykrywania błędów

Z książki Capers'a Jones'a „*Software Engineering Best Practices*” (2009) dowiadujemy się, że pomimo jej niezaprzeczalne wartości, metryka ta nie jest często stosowana:

*In spite of the fact that defect removal efficiency is a critical topic for successful software projects, measuring defect removal efficiency or software quality in general are seldom done. From visiting over 300 companies in the United States, Europe, and Asia, the author found the following distribution of the frequency of various kinds of quality measures:*

No quality measures at all	44%
Measuring only customer-reported defects	30%
Measuring test and customer-reported defects	18%
Measuring inspection, static analysis, test, and customer-reported defects	7%
Using volunteers for measuring personal defect removal	1%

Tabela 1 Statystyki wykorzystania różnych wariantów metryki DDE

#### 5.1.2 Przeznaczenie

Metryka służy głównie jako miernik skuteczności testów systemowych (wykonywanych przez Dostawcę oprogramowania): im mniej błędów wykryto na testach akceptacyjnych i po wdrożeniu, tym skuteczniejsze były testy systemowe. Skuteczność wykrywania błędów przekłada się również bezpośrednio na jakość produktu (wdrożonej wersji systemu).

### 5.1.3 Algorytm wyliczania

$$DDE = \frac{wz_{ts}}{wz_{ts} + wz_{uat} + wz_{wdr}} * 100\%$$

gdzie:

**wz<sub>ts</sub>** - ważona liczba zgłoszeń z testów systemowych

**wz<sub>uat</sub>** - ważona liczba zgłoszeń z testów akceptacyjnych

**wz<sub>wdr</sub>** - ważona liczba zgłoszeń po wdrożeniu produkcyjnym

**Ważona** oznacza, że liczba zgłoszeń o danym priorytecie (np. niski, normalny, krytyczny, itp) jest mnożona przez wagę przypisaną temu priorytetowi. Ważona liczba zgłoszeń to suma takich iloczynów dla wszystkich priorytetów w jakich są zarejestrowane zgłoszenia.

### 5.1.4 Źródła danych

Dane do metryki są pobierane bezpośrednio z bazy danych JIRA, używanej w PZU jako podstawowe narzędzie do śledzenia zgłoszeń błędów. Dla każdej rozwijanej aplikacji w JIRA istnieją 3 projekty:

- aplikacja-testy-systemowe,
- aplikacja-testy-akceptacyjne,
- aplikacja-zarządzanie-problemami,

w których rejestrowane są błędy odpowiednio: z testów systemowych i akceptacyjnych oraz błędy zgłoszone z produkcji, po wdrożeniu wersji aplikacji (problem = wynik zarządzania incydentami zgodnie z ITIL).

### 5.1.5 Warunki stosowania

Aby metryka była wiarygodna, zastosowano wagi dla priorytetów zgłoszonych błędów:

Niski	1
Normalny	2
Wysoki	3
Krytyczny	4
Blokada systemu	5

Tabela 2 Wagi priorytetów błędów

W analizie nie są uwzględniane zgłoszenia posiadające przypisane poniższe sposoby rozwiązania:

- nienaprawialny,
- duplikat,
- nie do powtórzenia,
- to nie jest błąd,

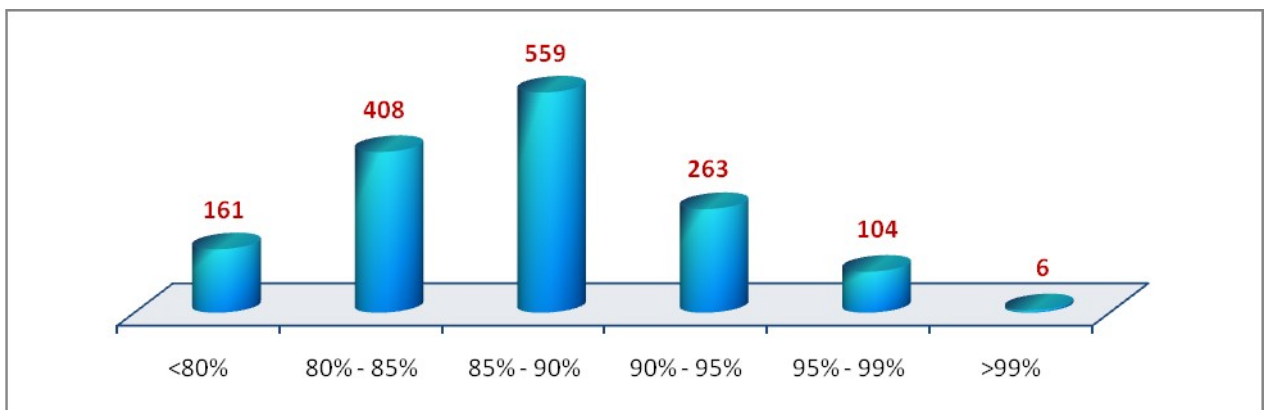
- odrzucone,

gdyż takich zgłoszeń nie można traktować jako błędy.

### 5.1.6 Poziomy kontrolne

Pożądaną wartością metryki jest 100% co oznacza, że w przekazanej do wdrożenia wersji systemu nie znaleziono żadnych błędów, gdyż wszystkie błędy wykryte zostały na etapie testów systemowych. W rzeczywistości mało prawdopodobne jest osiągnięcie tej wartości.

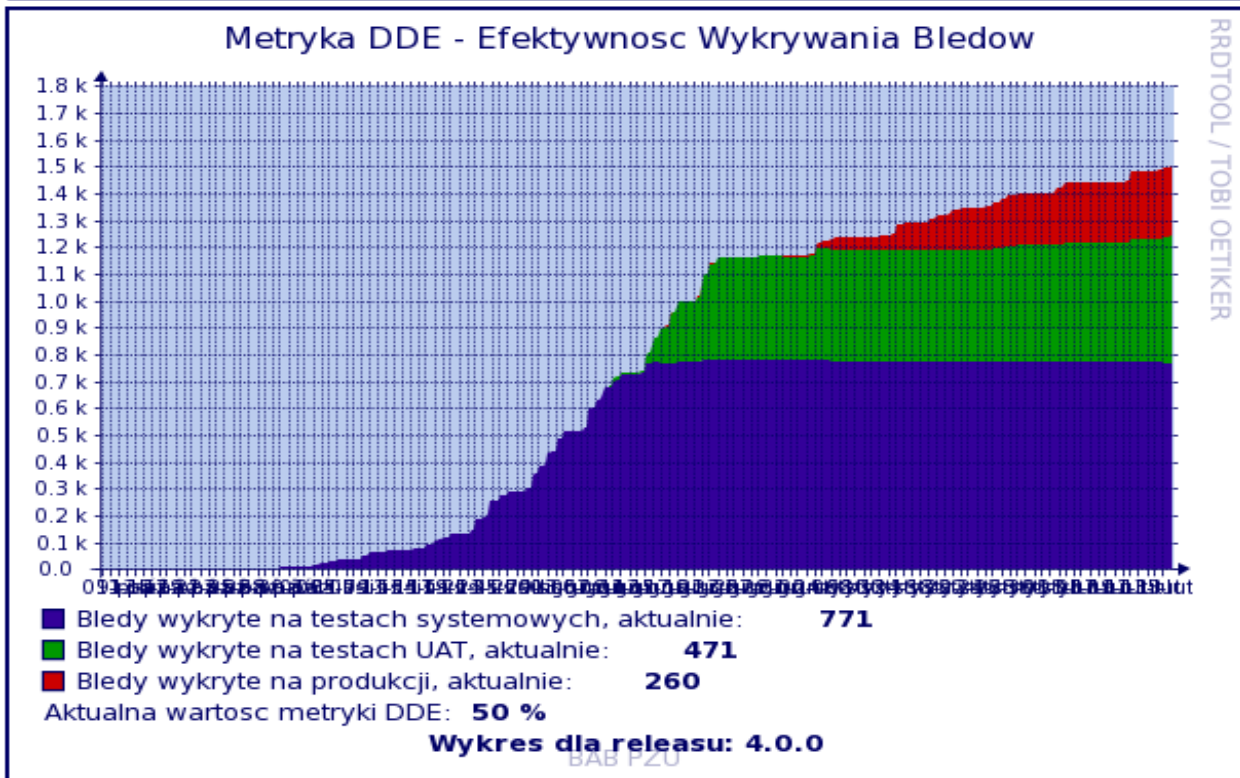
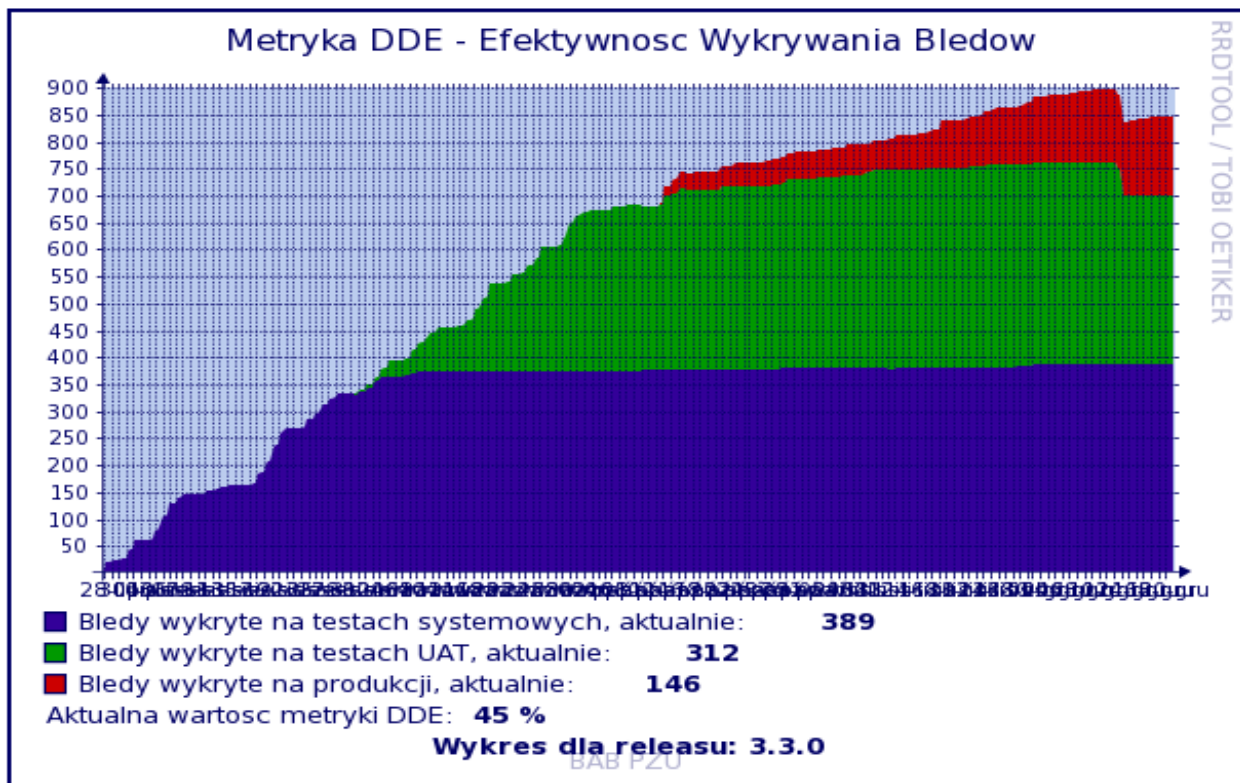
Na podstawie badań 1500 projektów w 2008 roku, Capers Jones wyznaczył rozkład wartości metryki DDE osiągniętych przez te projekty (źródło: *SPR Capers Jones, 2008* <http://www.scribd.com/doc/7758538/Capers-Jones-Software-Quality-in-2008>), przedstawia go poniższy wykres.



Rysunek 12 Rozkład wartości metryki DDE

### 5.1.7 Sposób prezentacji

Wartości metryki prezentowane są na wykresie w funkcji czasu. Metryka jest generowana oddzielnie dla 3 ostatnich wydań każdej aplikacji, aby możliwe było porównanie i ocena czy skuteczność testów systemowych poprawia się.



- Jej wartość będzie spadać w miarę jak będą zgłaszane błędy na testach akceptacyjnych, a potem jeszcze po wdrożeniu wersji na środowisku produkcyjnym.
- Im mniej spadnie wartość metryki po wdrożeniu wersji, tym lepsza jakość tej wersji ergo tym bardziej efektywny proces testów systemowych.
- Ostateczna wartość metryki dla release'u ustalana jest po wdrożeniu produkcyjnym kolejnej wersji, dla której również wyliczana jest wartość tej metryki (wtedy przestają już służyć zgłoszenia produkcyjne do poprzedniej wersji).
- Alternatywnie można wyznaczyć określony okres obserwacji release'u od jego wdrożenia, np.: 1 miesiąc. Po upływie tego okresu wartość metryki jest zamrażana dla tego release'u.

## 5.2 PRB – Przyrost Rozwiązanych Błędów

### 5.2.1 Przeznaczenie

Metryka pozwala na monitorowanie tempa rozwiązywania i zamykania błędów oraz porównanie z tempem zgłaszania błędów dla danego wydania. Opiera się na śledzeniu skumulowanej liczby błędów zgłoszonych podczas testów wydania, w podziale na ich stany:

- **rozwiązany** - poprawiony lub rozwiązany w inny sposób (np: uznany za nowe wymaganie) ale jeszcze nie zweryfikowany przez potwierdzenie lub retesty
- **zamknięty** - poprawiony lub rozwiązany w inny sposób oraz zweryfikowany w retestach lub przez potwierdzenie (np. że faktycznie jest to nowe wymaganie a nie błąd)

Dodatkowo równolegle obserwuje się skumulowaną liczbę wszystkich błędów zgłoszonych podczas testów wydania, niezależnie od ich stanu.

### 5.2.2 Algorytm wyliczania

Metryka opiera się na liczbie błędów zgłoszonych w podziale na ich stany - te dane są dostępne bezpośrednio w JIRA, dlatego nie jest stosowany żaden algorytm transformujący.

### 5.2.3 Źródła danych:

Metryka opiera się na danych zgłoszeń błędów, które są rejestrowane w JIRA. Ponieważ jednak JIRA nie udostępnia odpowiednich raportów, dane do metryki pobierane są bezpośrednio z jej bazy danych za pomocą odpowiednich zapytań SQL.

Do każdej z kategorii (rozwiązane, zamknięte) zgłoszenia kwalifikowane są po ich statusie:

- **rozwiązane** - zgłoszenia posiadające w JIRA status: "Do weryfikacji"
- **zamknięte** - zgłoszenia posiadające w JIRA status: "Zamknięte,,

### 5.2.4 Poziomy kontrolne

- **skumulowana liczba błędów rozwiązanych** – powinna mieć niewielkie wartości a na zakończenie testów powinna osiągnąć wartość 0 (gdy rozwiązanie błędu zostanie potwierdzone, np. w retestach, błąd zmienia status na „zamknięty”)

- **skumulowana liczba błędów zamkniętych i zgłoszonych** – moduł różnicy ich wartości nie powinien być zbyt duży w trakcie trwania testów, a pod ich koniec testów wartości te powinny się powoli zrównać ze sobą

### 5.2.5 Interpretacja

Metryka składa się z trzech wartości obserwowanych w funkcji czasu parametrów:

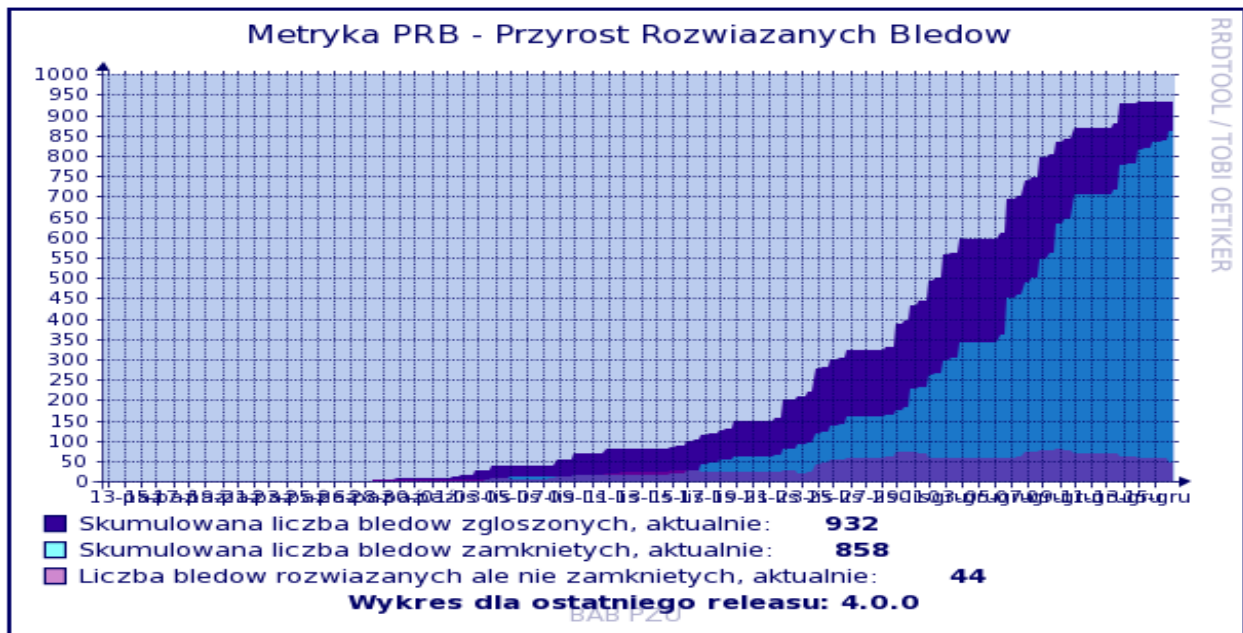
- **lbw** - liczby wszystkich błędów zgłoszonych od rozpoczęcia testów wydania ,
- **lbr** - liczby błędów rozwiązanych,
- **lbz** - liczby błędów zamkniętych,

Trendy zmian tych wartości w czasie i ich wzajemna korelacja pozwalają na interpretację stanu testów i wyciągania wniosków, np.:

1. Trend **lbr** zbliżony do linii poziomej a trendy **lbw** i **lbz** strome i do siebie podobne – błędy są rozwiązywane na bieżąco a także na bieżąco potwierdzana jest skuteczność rozwiązań przez retesty. Jest to sytuacja pożądana, pod warunkiem, że wartości lbr są relatywnie niskie, a trendy lbw i lbz stosunkowo niewiele od siebie oddalone.
2. Stromość trendu **lbz** mała, zbliżona do linii poziomej a trendy **lbw** i **lbr** strome i do siebie podobne – błędy są rozwiązywane w miarę ich zgłaszania na bieżąco, natomiast dużo wolniej potwierdzana jest skuteczność ich rozwiązania (retesty).
3. Stromość trendów **lbr** i **lbz** mniejsza niż stromość **lbw** – oznacza to, że tempo zgłaszania błędów jest większe od ich rozwiązywania i zamykania – jest to sygnał, że programiści nie nadążają z rozwiązywaniem błędów.
4. Początkowa odległość między trendami **lbr** a **lbw** jest mała - oznacza to, że rozwiązywanie błędów podjęto niezwłocznie po rozpoczęciu ich zgłaszania - jest to sytuacja pożądana.
5. Początkowa odległość między trendami **lbr** a **lbw** jest duża - oznacza to, że rozwiązywanie błędów podjęto późno, gdy w kolejce czekała już spora liczba zgłoszeń - jest to sytuacja niepokojąca. W szczególności jeśli trend **lbr** ma małą stromość, istnieje ryzyko, że nie uda się rozwiązać wszystkich zgłoszeń przed zakończeniem testów.
6. Odległość między trendami **lbz** a **lbr** jest mała - oznacza to, że weryfikację (w szczególności retesty) zgłoszeń rozwiązanych podjęto niezwłocznie po rozpoczęciu ich rozwiązywania - jest to sytuacja pożądana.
7. Odległość między trendami **lbz** a **lbr** jest duża - mogą być 2 przyczyny takiego stanu:
  - weryfikację (w szczególności retesty) zgłoszeń rozwiązanych podjęto późno
  - późno wydano wersję poprawkową testowanej aplikacji, zawierającą poprawki błędów do weryfikacji

### 5.2.6 Sposób prezentacji:

Szereg czasowy skumulowanej liczby błędów w każdym ze stanów z dziennym okresem obserwacji



Rysunek 14 Przykładowy wykres metryki PRB

## 6. Propozycje metryk z TestLink

### 6.1 Warunki stosowania metryk TestLink

Aby móc stosować metryki na podstawie danych z TestLink, w PZU wprowadziliśmy następujące zasady rejestracji danych w tym narzędziu:

- muszą być rejestrowane dokumenty specyfikacji wymagań, zawierające wszystkie zmiany (nowe wymagania i poprawki błędów) planowane do przetestowania w danym wydaniu,
- dla każdego dokumentu specyfikacji wymagań musi być uzupełnione pole użytkownika o etykiecie "Szacowana liczba testów" wartością oszacowaną na etapie planowania testów,
- testy muszą być powiązane ze zmianą/zmianami, których poprawność mają weryfikować,
- muszą być rejestrowane plany testów, do których dowiązywane są wszystkie testy planowane do wykonania w danym wydaniu.
- testy, których projektowanie zakończono i które są gotowe do wykonania, muszą być oznaczane słowem kluczowym "GOTOWE",
- każdemu testowi powinien być nadawany jeden z trzech poziomów ważności: wysoka, średnia lub niska, który koresponduje z priorytetami,
- aktywne plany testów oraz dokumenty specyfikacji wymagań muszą być oznaczane prefixem „[a]” w ich nazwach. Gdy stają się nieaktywne, należy usuwać ten prefix z ich nazw.

### 6.2 PPT – Postęp Przygotowania Testów

#### 6.2.1 Przeznaczenie

Metryka pozwala śledzić postęp projektowania przypadków testowych w stosunku do planu. Jej wartość wyraża procent liczby testów gotowych do wykonania.

## 6.2.2 Algorytm

$$PPT = \frac{itpz}{slt} * 100\%$$

- gdzie:

gdzie:

- itpz** liczba testów pokrywających zmiany (wszystkie testy, które są powiązane przynajmniej z jedną zmianą i są oznaczone słowem kluczowym "GOTOWY")
- slt** szacowana liczba wszystkich testów do zaprojektowania (liczba oszacowana na etapie planowania testów na podstawie analizy zakresu zmian przeznaczonych do realizacji w wydaniu)

## 6.2.3 Źródła danych

Dane do wyliczenia metryki są pozyskiwane bezpośrednio z bazy danych TestLinka. Metryki są wyliczane dla każdej aktywnej specyfikacji wymagań zarejestrowanej w TestLink. Jako liczba zaplanowanych testów do pokrycia wymagań, brana jest wartość wpisana do pola użytkownika o nazwie „Szacowana liczba testów”, które zostało przypisane do specyfikacji wymagań. Test pokrywający wymagania ze specyfikacji to taki, który został powiązany z wymaganiem z tej specyfikacji, a także został oznaczony słowem kluczowym „GOTOWY”.

## 6.2.4 Warunki stosowania:

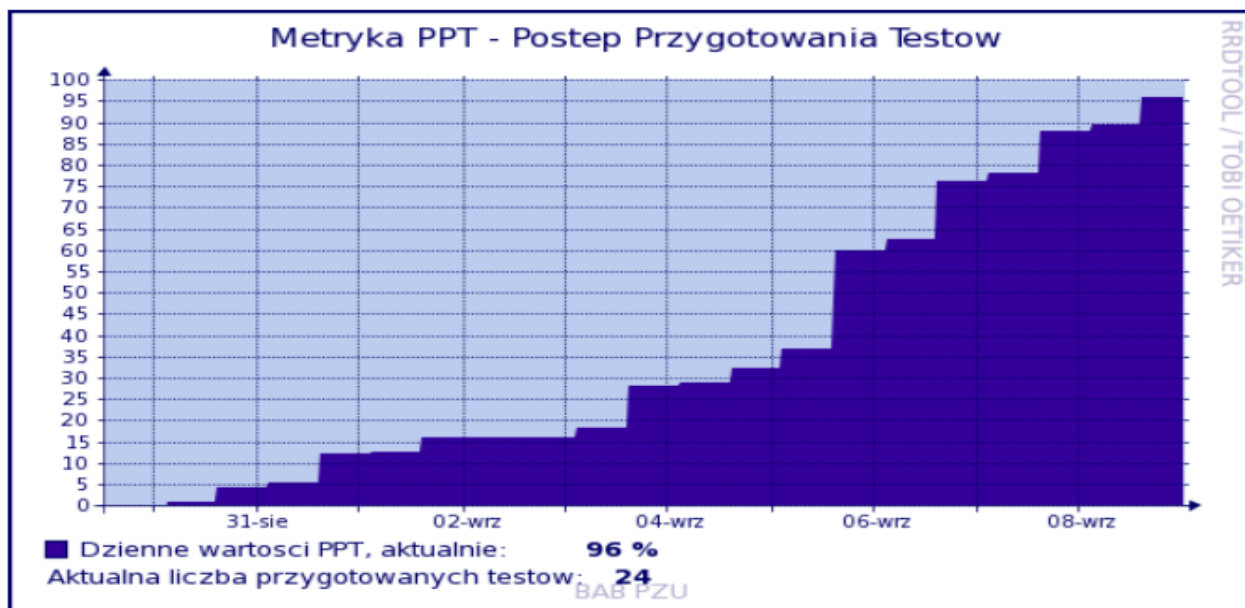
- w TestLink muszą być rejestrowane specyfikacje wymagań, zawierające wszystkie zmiany planowane do przetestowania w danym wydaniu,
- dla każdej specyfikacji wymagań musi być uzupełnione pole użytkownika o etykiecie "Szacowana liczba testów" wartością oszacowaną na etapie planowania testów
- testy, których projektowanie zakończono i które są gotowe do wykonania, muszą być oznaczane słowem kluczowym "GOTOWE"
- testy muszą być powiązane ze zmianą/zmianami, których poprawność mają weryfikować

## 6.2.5 Poziomy kontrolne

Wartość metryki powinna osiągnąć wartość = 100%, co będzie oznaczać, że zostały zaprojektowane wszystkie z zaplanowanych testów.

## 6.2.6 Sposób prezentacji

Szereg czasowy wartości procentowej metryki z dziennym okresem obserwacji.



Rysunek 15 Przykładowy wykres metryki PPT

## 6.2.7 Interpretacja:

W trakcie etapu przygotowania testów mogą zdarzyć się następujące sytuacje:

- **PPT = 100%** - jest to sytuacja pożądana pod koniec terminu zakończenia etapu projektowania testów.
- **PPT > 100%** - oznacza to, że wartość slt została niedoszacowana i nie dokonano jej aktualizacji. W takiej sytuacji należy poprosić Testera o uaktualnienie szacunku dla slt i o dokonanie aktualizacji w TestLinku.
- **PPT << 100% mimo zbliżającego się terminu zakończenia etapu przygotowania testów** - taka sytuacja jest niepokojąco i wymaga interwencji. Oznaczać może, że Testerzy nie wyrabiają się z projektowaniem testów i istnieje zagrożenie, że nie zdążą przygotować zaplanowanej liczby testów przed końcem ustalonego terminu.
- **wartość PPT gwałtownie wzrasta lub spada** - oznaczać to może najczęściej, że dokonano przeszacowania wartości slt w dół lub w górę: w wyniku rozszerzenia zakresu zmian do przetestowania lub też korekty poprzedniego, niedokładnego oszacowania.

## 6.3 PTZ – Postęp Testowania Zmian

### 6.3.1 Przeznaczenie:

Metryka pozwala śledzić postęp testowania zmian (wymagań lub zaplanowanych do implementacji poprawek znanych błędów): liczbę zmian już przetestowanych oraz liczbę zmian, dla których wyniki testów są negatywne.

### 6.3.2 Algorytm:

Metryka opiera się na liczbie wymagań, które zostały przetestowane - te dane są dostępne bezpośrednio w TestLink, dlatego nie jest stosowany żaden algorytm transformujący.

### 6.3.3 Źródła danych:

Dane są pozyskiwane bezpośrednio z bazy danych TestLinka dla każdego aktywnego planu testów. Jako liczba zmian do przetestowania brana jest liczba wszystkich zmian, z którymi powiązane są przypadki testowe należące do planu testów.

### 6.3.4 Warunki stosowania:

- w TestLink muszą być rejestrowane specyfikacje wymagań,
- wszystkie testy dołączone do planu testów muszą być powiązane z wymaganiami, których poprawność mają weryfikować.

### 6.3.5 Poziomy kontrolne:

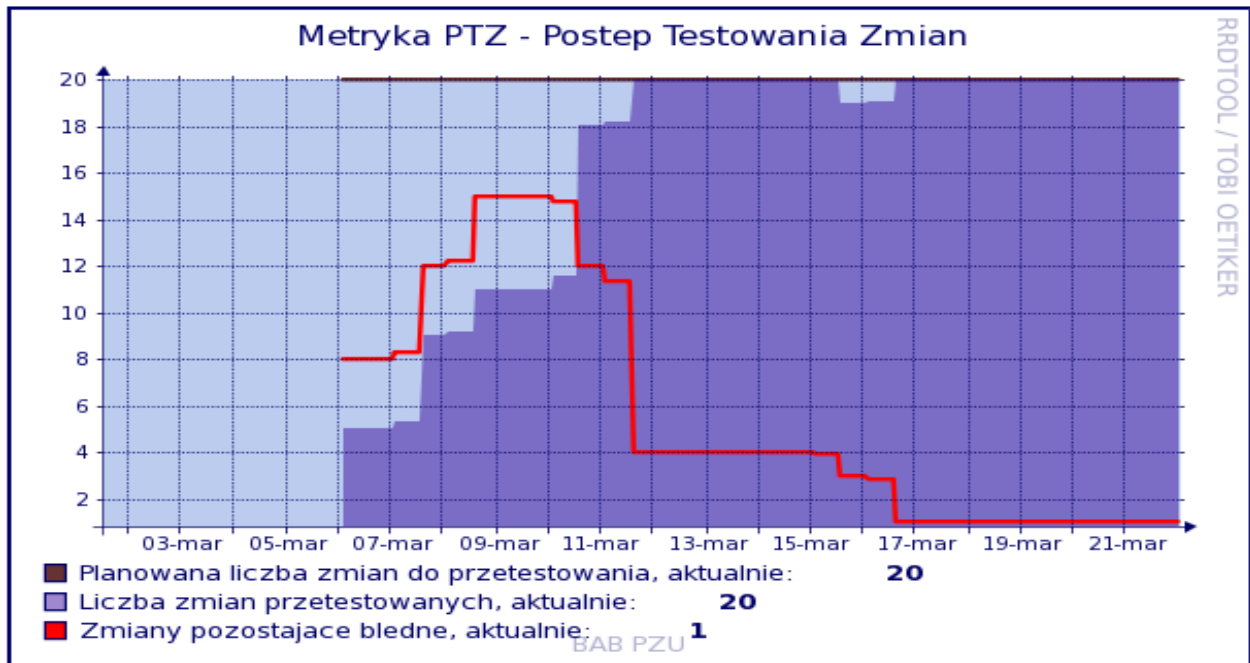
- liczba zmian przetestowanych powinna się zrównać z liczbą zmian zaplanowanych do przetestowania,
- liczba zmian pozostających w statusie: błędna, powinna osiągnąć 0.

### 6.3.6 Interpretacja:

- zmianę uważa się za **przetestowaną**, jeśli zostaną wykonane wszystkie testy z nią powiązane (może być ich wiele), niezależnie od ich wyniku; jeśli choć jeden test powiązany ze zmianą pozostanie niewykonany, zmiana wciąż będzie miała status nieprzetestowanej,
- zmianę uważa się za **pozostającą w statusie: błędna**, jeśli choć jeden test powiązany ze zmianą pozostaje w stanie: negatywny lub zablokowany.
- obserwując zmiany wartości metryki PTZ w czasie, można śledzić ogólne tempo postępu testowania zmian (cykliczny przyrost zmian przetestowanych).

### 6.3.7 Sposób prezentacji

Szereg czasowy liczby zmian przetestowanych oraz pozostających w statusie: błędna z dziennym okresem aktualizacji:



Rysunek 16 Przykładowy wykres metryki PTZ

## 6.4 PTN – Przyrost Testów Negatywnych

### 6.4.1 Przeznaczenie

Metryka pozwala śledzić dzienny przyrost liczby testów negatywnych. Wykres jej wartości w czasie obrazuje trend liczby błędów wykrywanych w ciągu dnia w czasie trwania testów, a także skumulowaną liczbę wszystkich testów negatywnych (niezależnie od ich późniejszej zmiany statusu na pozytywny w wyniku retestów).

### 6.4.2 Algorytm

Metryka opiera się na liczbie testów, dla których zarejestrowano wynik negatywny w TestLink w ramach aktywnego planu testów. Dane te są dostępne bezpośrednio w TestLink, dlatego nie jest stosowany żaden algorytm transformujący.

### 6.4.3 Źródła danych

Dane są pozyskiwane bezpośrednio z bazy danych TestLinka dla każdego aktywnego planu testów. Jako liczba testów negatywnych w danym dniu, traktowana jest liczba wszystkich wyników negatywnych zarejestrowanych w tym dniu dla danego planu testów.

### 6.4.4 Warunki stosowania

- W TestLinku muszą być rejestrowane plany testów, do których dowiązywane są wszystkie testy planowane do wykonania w danym wydaniu,

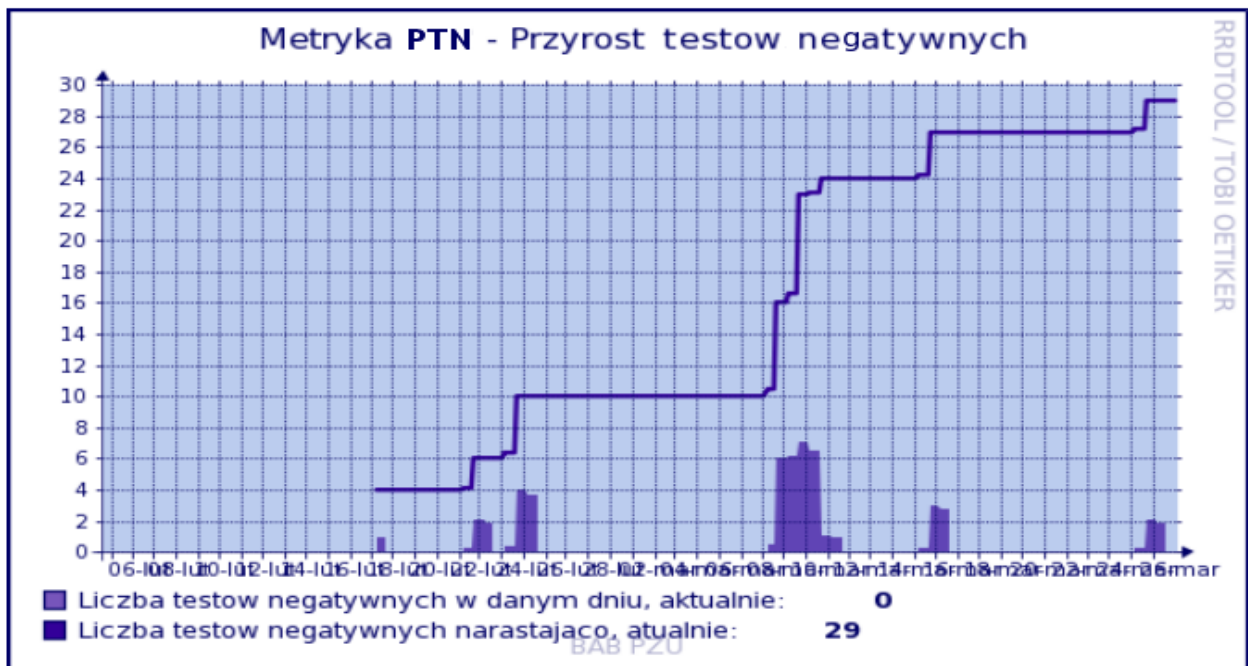
- niezwłocznie po wykonaniu każdego testu, jego wynik rejestrowany jest w TestLinku.

#### 6.4.5 Poziomy kontrolne

- Dzienna liczba testów negatywnych powinna zmierzać do 0 i osiągnąć tę wartość na zakończenie testów,
- skumulowana liczba testów negatywnych powinna stabilizować się na pewnej wartości, co oznacza że nie są znajdowane już żadne nowe błędy.

#### 6.4.6 Sposób prezentacji:

Szereg czasowy skumulowanej liczby wyników negatywnych testów oraz liczby testów z zarejestrowanym statusem negatywnym w danym dniu.



Rysunek 17 Przykładowy wykres metryki PTZ

#### 6.4.7 Interpretacja:

- Poszczególne wartości metryki oznaczają liczbę testów z wynikiem negatywnym, zarejestrowanych w ciągu dnia, natomiast wartość skumulowana oznacza sumaryczną liczbę zgłoszonych wyników negatywnych, bez względu na aktualny status testu.
- Najbardziej typowym kształtem (co do trendu) wykresu liczby testów negatywnych w ciągu dnia jest krzywa przypominająca spłaszczoną krzywą gęstości rozkładu normalnego Gaussa.
- Pożądane jest aby kształt wykresu liczby testów negatywnych w ciągu dnia był lewoskojny lub symetryczny, gdyż oznacza to, że najwięcej błędów wykrywanych jest na początku i w środkowej fazie testów.
- Jeśli kształt wykresu liczby testów negatywnych w ciągu dnia jest wyraźnie prawoskojny - oznaczać to może, że największa intensywność testów (wykrywania błędów) przypada pod koniec sesji testowej.
- Wykres skumulowanej liczby testów negatywnych powinien na początku stromo wzrastać, a im bliżej końca testów, tym powinien stabilizować się na jednej wartości.