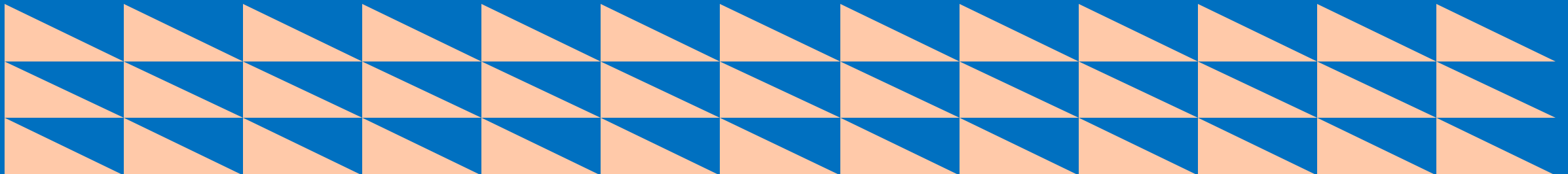
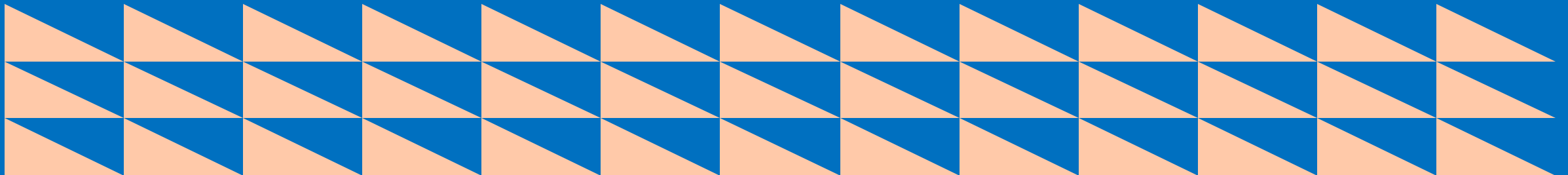


Statyczne metryki kodu. Predykcja defektów oprogramowania.



Statyczne metryki ~~kodu~~.

Predykcja defektów oprogramowania.



Jaka jest (według Ciebie) najistotniejsza cecha stojąca za stosowaniem **wzorców projektowych** i **dobrych praktyk pisania kodu** (w tym używaniem statycznej analizy kodu) w zespole?

Artem Wczoraj o 19:27

bo są to sprawdzone i efektywne rozwiązania dla powszechnych problemów?

F!lip Wczoraj o 19:16

O kurde xd ciekawe pytanie

Imo największy zysk jest z tego że kod jest spójny

I nie robi się tego samego na parę różnych sposobów

zwatotem Wczoraj o 19:37

Wiesz co? Dla mnie to jest pytanie z kategorii

[redacted], bo

osobiście jestem przeciwnikiem wzorców projektowych. Uważam, że wprowadzają dodatkowy poziom komplikacji do kodu (w dodatku niejawniej) i wywołują w zespole zbyt dużą ilość bezproduktywnych dyskusji.

Wojak Wczoraj o 19:26

maintainability kodu (edytowane)

i destrukcja kapitalizmu

klaudia Wczoraj o 19:33

Przejrzystosc?

Ksiądz Wczoraj o 19:21

Programista otrzymuje rozwiązanie na napotkany problem wraz ze znajomością konsekwencji jego zastosowania.

Dawid Wczoraj o 19:30

Bo dzięki nim łatwiej wprowadzać nowe featurey

Oraz kod jest czytelniejszy z reguły

Hitman Wczoraj o 19:33

Spójność

Może coś takiego

Arturex Wczoraj o 20:07

W sumie trudne pytanie wylosowało

Mi się wydaje że nie jest aż tak ważne w kwestii wzorców

W sensie jak nie masz intuicji jak pisać kod który jest utrzymywalny to fajny punkt odniesienia, ale potem to już nie aż tak istotne

Konrados232 Wczoraj o 21:50

Jasna komunikacja w zespole

Artem Wczoraj o 19:27

bo są to sprawdzone i efektywne rozwiązania
dla powszechnych problemów

Filip Wczoraj o 19:16

O kurde xd ciekawe pytanie

o największy zysk jest z tego że kod jest spójny
robi się tego samego na parę różnych sposobów

zwatorem Wczoraj o 19:37

Wiesz co? Dla mnie to jest pytanie z kategorii

[redacted], bo

osobiście jestem przeciwnikiem wzorców projektowych. Uważam, że wprowadzają dodatkowy poziom komplikacji do kodu (w dodatku niejawniej) i wywołują w zespole zbyt dużą ilość bezproduktywnych dyskusji.

Wojak Wczoraj o 19:26

maintainability kodu (edytowane)
i destrukcja kapitalizmu

Ksiądz Wczoraj o 19:21

Programista otrzymuje rozwiązanie na napotkany problem wraz ze znajomością konsekwencji jego zastosowania.

klau Wczoraj o 19:33

Przejść do punktu?

Dawid Wczoraj o 19:30

Bo dzięki nim łatwiej wprowadzać nowe featurey
Oraz kod jest czytelniejszy z reguły

Hitman Wczoraj o 19:33

Spójność

Może coś takiego

Arturex Wczoraj o 20:07

W sumie trudne pytanie w tym kontekście

Mi się wydaje że nie jest aż tak ważne w kwestii wzorców

W sensie jak nie masz innego punktu odniesienia, ale potem to już nie aż tak istotne

Konrados232 Wczoraj o 21:50

Jasna komunikacja w zespole

Analiza
statyczna

Monitoring



Działanie
programu

- LOC (linie kodu)
- Całkowita/średnia złożoność cyklomatryczna
- Liczba zależności
- Ilość plików
- Pokrycie testowe
- Duplikaty kodu
- ...

Projekt

Projekt

The diagram consists of two light blue rounded rectangular containers. The top container contains a dark blue rounded rectangle with the word 'Projekt' in white. A thin vertical line connects the bottom center of this rectangle to the top center of a similar dark blue rounded rectangle in the bottom container, which contains the word 'Pakiet' in white. To the left of the 'Pakiet' rectangle, there is a block of text in black.

Metryki podobne
do tych z projektu,
ale ograniczone do
zasięgu pakietu

Pakiet

Projekt

Pakiet

Klasa

- LOC
- Ilość metod i zmiennych
- Średnia złożoność cyklopatyczna
- Relacje z innymi klasami
- ...

Project
level

Package
level

Class
level

Method
level

- LOC
- Zwracany typ
- Złożoności: cyklometryczna i inne
- Głębokość
- ...

Złożoność cyklomatyczna (McCabe, 1976)

- miara złożoności oparta o punkty decyzyjne programu

G – graf reprezentujący dany fragment logiczny kodu

E – liczba krawędzi grafu (w praktyce: liczba punktów decyzyjnych)

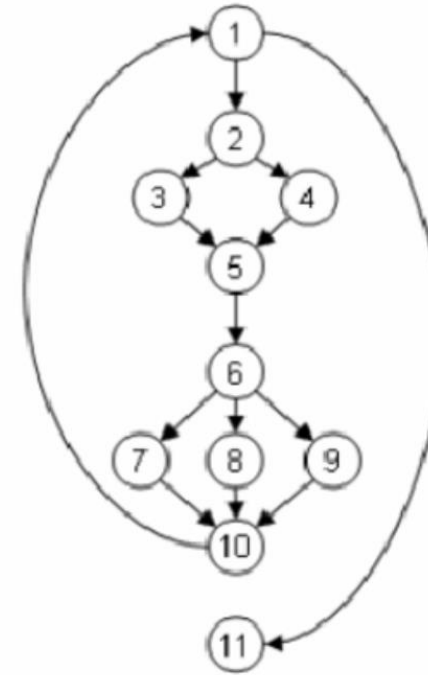
N – liczba wierzchołków grafu (w praktyce: liczba stanów)

P – liczba spójnych składowych grafu (w praktyce często równa 1)

$$\mathbf{MC(G) = E - N + 2P \approx \text{liczba decyzji binarnych} + 1}$$

Złożoność cyklomatyczna (McCabe, 1976)

Node	Statement
(1)	while(x<100){
(2)	if (a[x] % 2 == 0) {
(3)	parity = 0;
	}
(4)	else {
(5)	parity = 1;
(6)	} switch(parity){
(7)	case 0:
	println("a[" + i + "]" is even");
(8)	case 1:
	println("a[" + i + "]" is odd");
	default:
(9)	println("Unexpected error");
	}
(10)	x++;
(11)	}
	p = true;



$$MC(G) = 14 - 11 + 2 = 5$$

```
import java.net.URI;
import java.net.URISyntaxException;
import java.util.concurrent.ExecutorService;

import org.apache.log4j.LogManager;

import net.data.technology.jraft.RpcClient;
import net.data.technology.jraft.RpcClientFactory;

public class RpcTcpClientFactory implements RpcClientFactory {
    private ExecutorService executorService;

    public RpcTcpClientFactory(ExecutorService executorService){
        this.executorService = executorService;
    }

    @Override
    public RpcClient createRpcClient(String endpoint) {
        try {
            URI uri = new URI(endpoint);
            return new RpcTcpClient(new InetSocketAddress(uri.getHost(), uri.getPort()), this.executorService);
        } catch (URISyntaxException e) {
            LogManager.getLogger(getClass()).error(String.format("%s is not a valid uri", endpoint));
            throw new IllegalArgumentException("invalid uri for endpoint");
        }
    }
}
```

```
import java.net.URI;
import java.net.URISyntaxException;
import java.util.concurrent.ExecutorService;


import org.apache.log4j.LogManager;

import net.data.technology.jraft.RpcClient;
import net.data.technology.jraft.RpcClientFactory;

public class RpcTcpClientFactory implements RpcClientFactory {
    private ExecutorService executorService;

    public RpcTcpClientFactory(ExecutorService executorService){
        this.executorService = executorService;
    }

    @Override
    public RpcClient createRpcClient(String endpoint) {
        try {
            URI uri = new URI(endpoint);
            return new RpcTcpClient(new InetSocketAddress(uri.getHost(), uri.getPort()), this.executorService);
        } catch (URISyntaxException e) {
            LogManager.getLogger(getClass()).error(String.format("%s is not a valid uri", endpoint));
            throw new IllegalArgumentException("invalid uri for endpoint");
        }
    }
}
```



```
import java.net.URI;
import java.net.URISyntaxException;
import java.util.concurrent.ExecutorService;

import org.apache.log4j.LogManager;

import net.data.technology.jraft.RpcClient;
import net.data.technology.jraft.RpcClientFactory;

public class RpcTcpClientFactory implements RpcClientFactory {
    private ExecutorService executorService;

    public RpcTcpClientFactory(ExecutorService executorService){
        this.executorService = executorService;
    }

    @Override
    public RpcClient createRpcClient(String endpoint) {
        try {
            URI uri = new URI(endpoint);
            return new RpcTcpClient(new InetSocketAddress(uri.getHost(), uri.getPort()), this.executorService);
        } catch (URISyntaxException e) {
            LogManager.getLogger(getClass()).error(String.format("%s is not a valid uri", endpoint));
            throw new IllegalArgumentException("invalid uri for endpoint");
        }
    }
}
```

```
import java.net.URI;
import java.net.URISyntaxException;
import java.util.concurrent.ExecutorService;

import org.apache.log4j.LogManager;

import net.data.technology.jraft.RpcClient;
import net.data.technology.jraft.RpcClientFactory;

public class RpcTcpClientFactory implements RpcClientFactory {
    private ExecutorService executorService;

    public RpcTcpClientFactory(ExecutorService executorService){
        this.executorService = executorService;
    }

    @Override
    public RpcClient createRpcClient(String endpoint) {
        try {
            URI uri = new URI(endpoint);
            return new RpcTcpClient(new InetSocketAddress(uri.getHost(), uri.getPort()), this.executorService);
        } catch (URISyntaxException e) {
            LogManager.getLogger(getClass()).error(String.format("%s is not a valid uri", endpoint));
            throw new IllegalArgumentException("invalid uri for endpoint");
        }
    }
}
```



```
import java.net.URI;
import java.net.URISyntaxException;
import java.util.concurrent.ExecutorService;

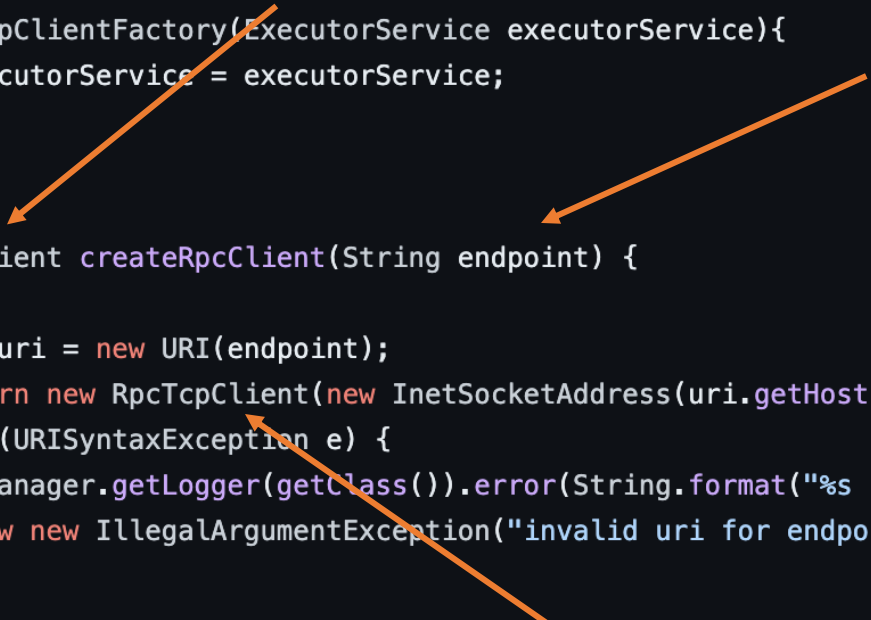
import org.apache.log4j.LogManager;

import net.data.technology.jraft.RpcClient;
import net.data.technology.jraft.RpcClientFactory;

public class RpcTcpClientFactory implements RpcClientFactory {
    private ExecutorService executorService;

    public RpcTcpClientFactory(ExecutorService executorService){
        this.executorService = executorService;
    }

    @Override
    public RpcClient createRpcClient(String endpoint) {
        try {
            URI uri = new URI(endpoint);
            return new RpcTcpClient(new InetSocketAddress(uri.getHost(), uri.getPort()), this.executorService);
        } catch (URISyntaxException e) {
            LogManager.getLogger(getClass()).error(String.format("%s is not a valid uri", endpoint));
            throw new IllegalArgumentException("invalid uri for endpoint");
        }
    }
}
```



```

import java.net.URI;
import java.net.URISyntaxException;
import java.util.concurrent.ExecutorService;

import org.apache.log4j.LogManager;

import net.data.technology.jraft.RpcClient;
import net.data.technology.jraft.RpcClientFactory;

public class RpcTcpClientFactory implements RpcClientFactory {
    private ExecutorService executorService;

    public RpcTcpClientFactory(ExecutorService executorService){
        this.executorService = executorService;
    }

    @Override
    public RpcClient createRpcClient(String endpoint) {
        try {
            URI uri = new URI(endpoint);
            return new RpcTcpClient(new InetSocketAddress(uri.getHost(), uri.getPort()), this.executorService);
        } catch (URISyntaxException e) {
            LogManager.getLogger(getClass()).error(String.format("%s is not a valid uri", endpoint));
            throw new IllegalArgumentException("invalid uri for endpoint");
        }
    }
}

```

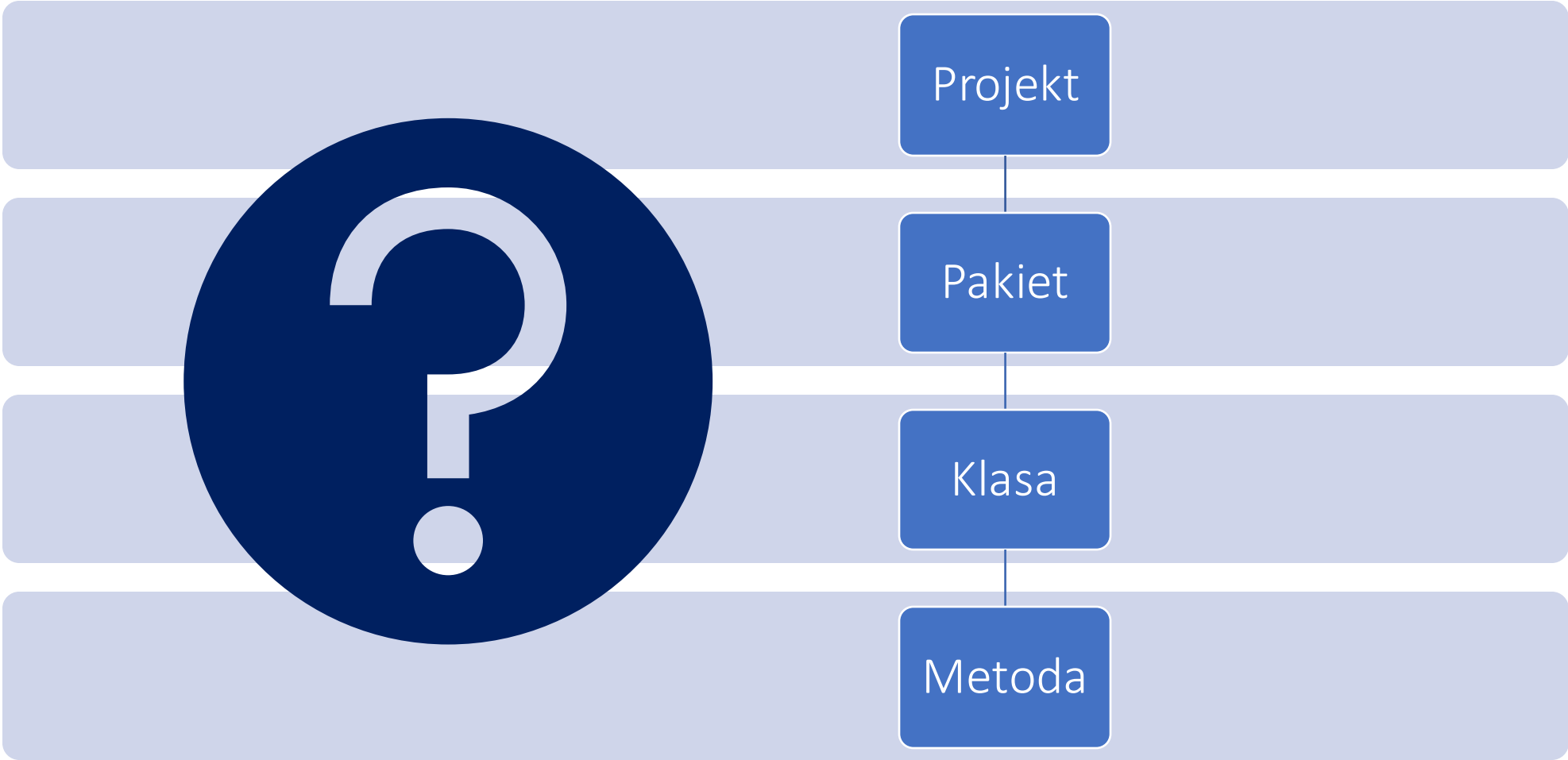
McCabe = 1, CBO = 3, returnQty = 1,
logStatements = 1, methodCalls = 1,
...



Najważniejsze narzędzia dla Javy

- SonarQube
- PMD
- CK

```
className: 'org.markdownj.MarkdownProcessor$Anonymous6',
methodName: 'genericCodeBlock/1[Ljava.lang.String]',
isConstructor: false,
startLine: 414,
cbo: 0,
cboModified: 1,
fanin: 1,
fanout: 0,
wmc: 1,
rfc: 1,
loc: 4,
returnQty: 1,
variablesQty: 1,
parametersQty: 1,
methodInvocationsSize: 1,
methodInvocationsLocalSize: 0,
methodInvocationsIndirectLocalSize: 0,
loopQty: 0,
comparisonsQty: 0,
tryCatchQty: 0,
parenthesizedExpsQty: 0,
stringLiteralsQty: 1,
numbersQty: 0,
assignmentsQty: 1,
mathOperationsQty: 0,
maxNestedBlocks: 0,
anonymousClassesQty: 0,
innerClassesQty: 0,
lambdasQty: 0,
uniqueWordsQty: 7,
modifiers: 1,
logStatementsQty: 0,
hasJavadoc: false
```



Ogół projektu

Programiści: seniority,
dyspozycja dnia, miara
profesjonalizmu,
wydajność

Jakość zarządzania w
firmie i zespole

Satysfakcja z
wykonywanej pracy,
pasja

Wartość biznesowa

Wielkość zespołu/ilość
contributorów

Zmienność wymagań

... i wiele innych

Projekt

Pakiet

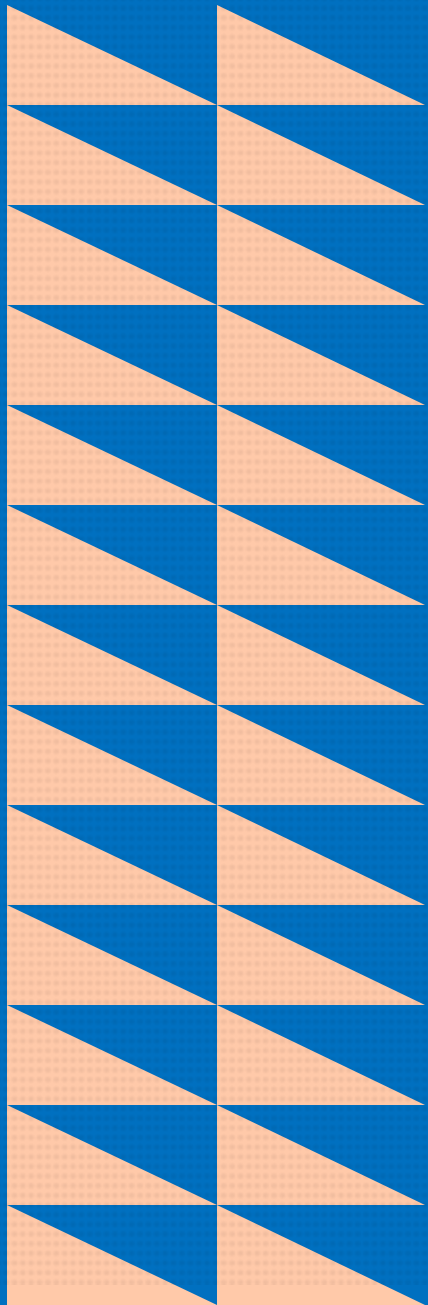
Klasa

Metoda

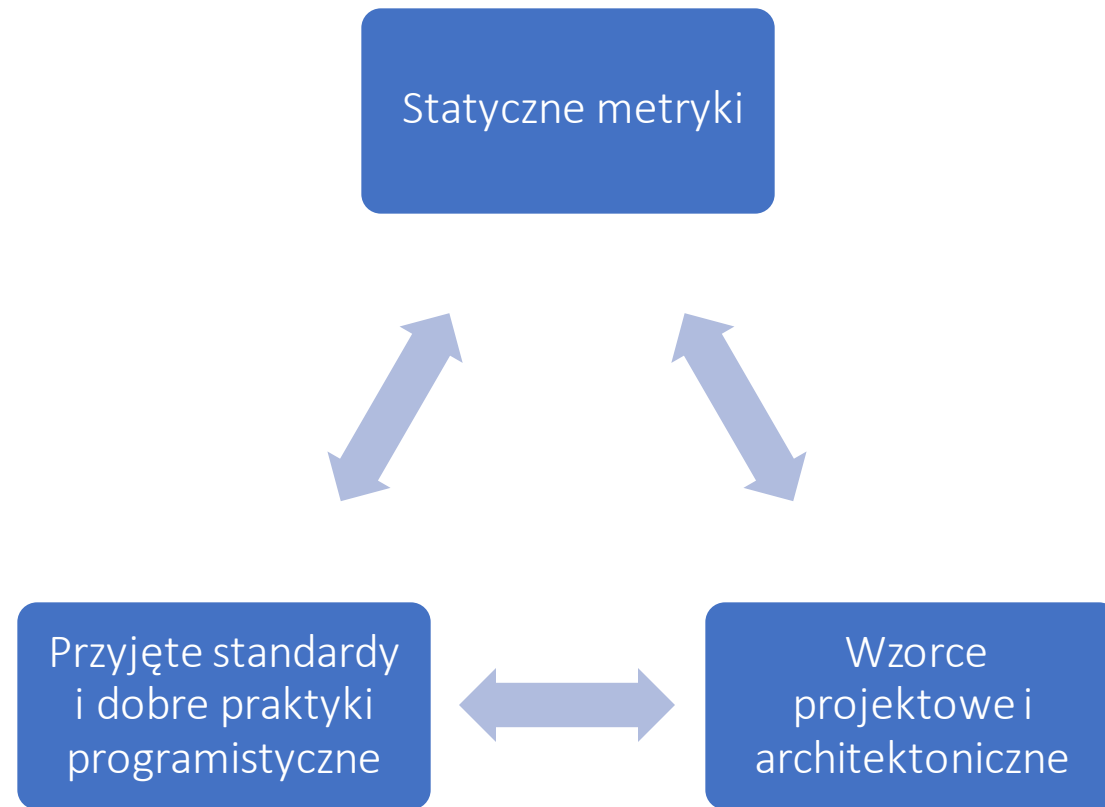
Metryką może być praktycznie **wszystko**.

Co nie jest równoznaczne z tym, że każda miara będzie miała (duże) znaczenie przy ewentualnym wnioskowaniu.

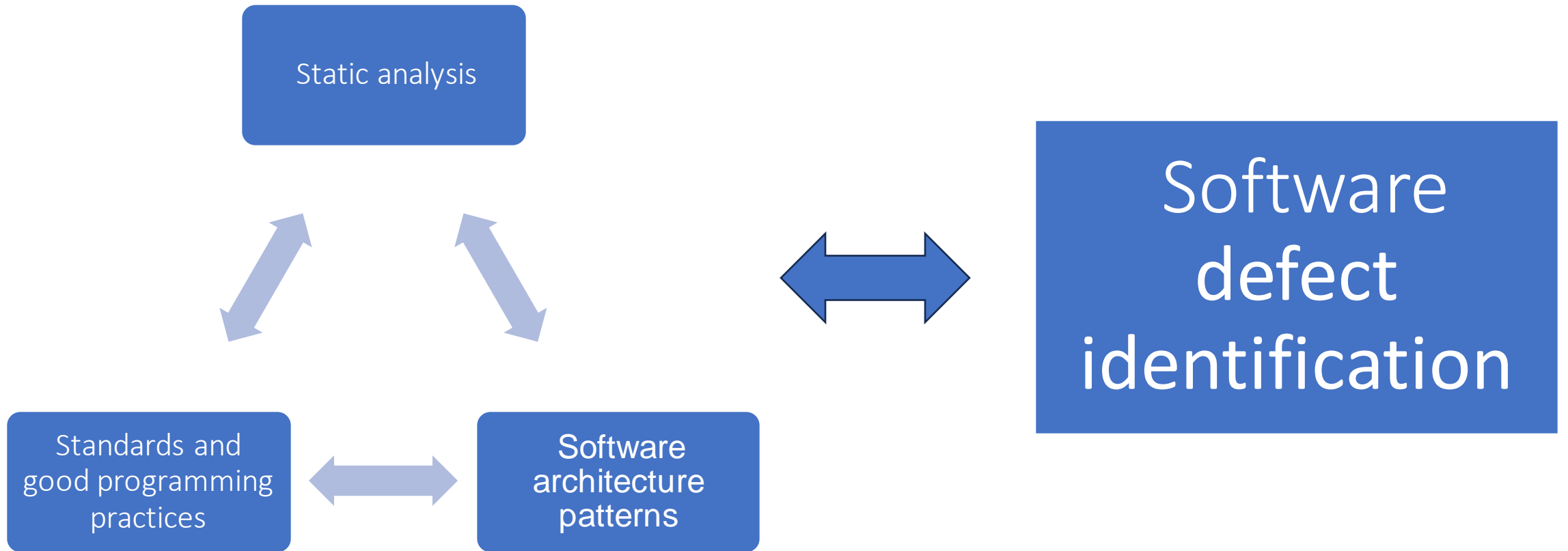
- G. Weinberg, *"The Psychology of Computer Programming"* - 1971
- P. Louridas, *"Static code analysis"* - 2006
- <https://github.com/mauricioaniche/ck/blob/master/README.md>
- J. Zhang, *"Predictive Mutation Testing"* - 2019



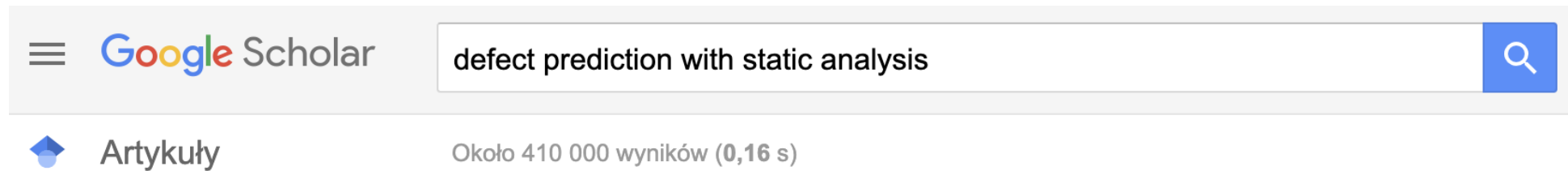
**A czy Ty
piszesz swój kod
przed śniadaniem?**



- S. Uchiyama, *"Design Pattern Detection using Software Metrics and Machine Learning"* - 2011
- F. Jaafar, *"Analysing Anti-patterns Static Relationships with Design Patterns"* - 2013



Predykcja występowania defektów



https://scholar.google.com/scholar?hl=pl&as_sdt=0%2C5&q=defect+prediction+with+static+analysis

Predykcja występowania defektów

X_1, \dots, X_n – statyczne metryki projektowe (głównie kodu) wskazane dla metody

Y – prawdopodobieństwo defektogenności (podatności) metody

X_1	X_2	...	X_n	Y
5.0	21.214	...	72419.42312	0.37
68.0	331.10		21.452	0.29
2145	50.213		5123.127	0.88
220.0	51.255	...	99124.33344	0.48

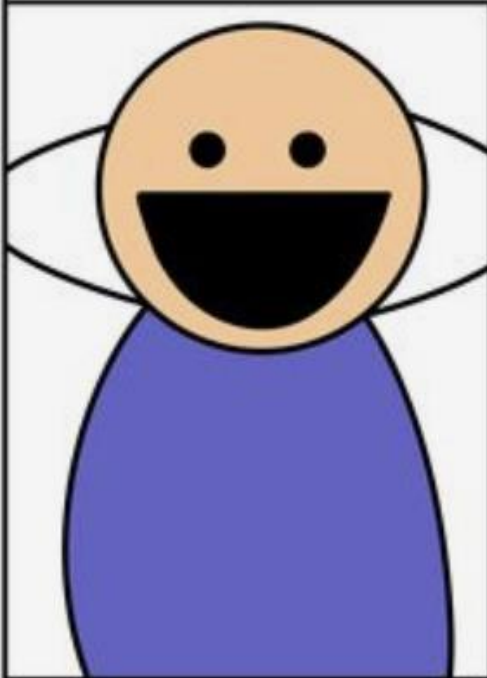
Istnieje **silna korelacja** pomiędzy **metrykami statycznymi**, a wzorcami projektowymi, czytelnością kodu oraz występowaniem **defektów** w kodzie.

V O L V O

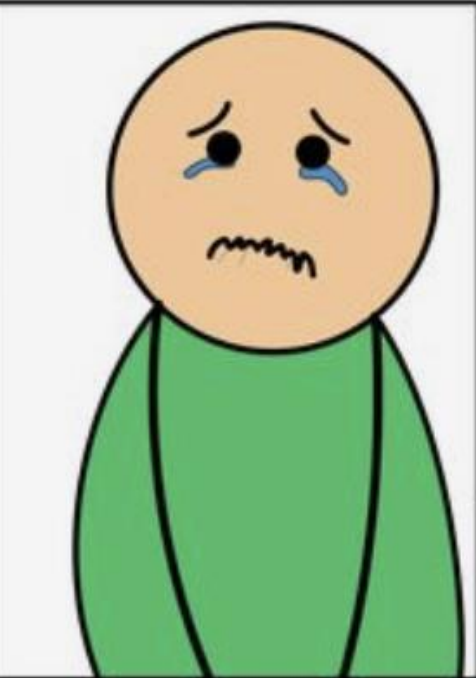


- .NET
- System kluczowy do produkcji pojazdów
- Wersja aplikacji: v10
- Legacy code

The reaction to a "BUG"



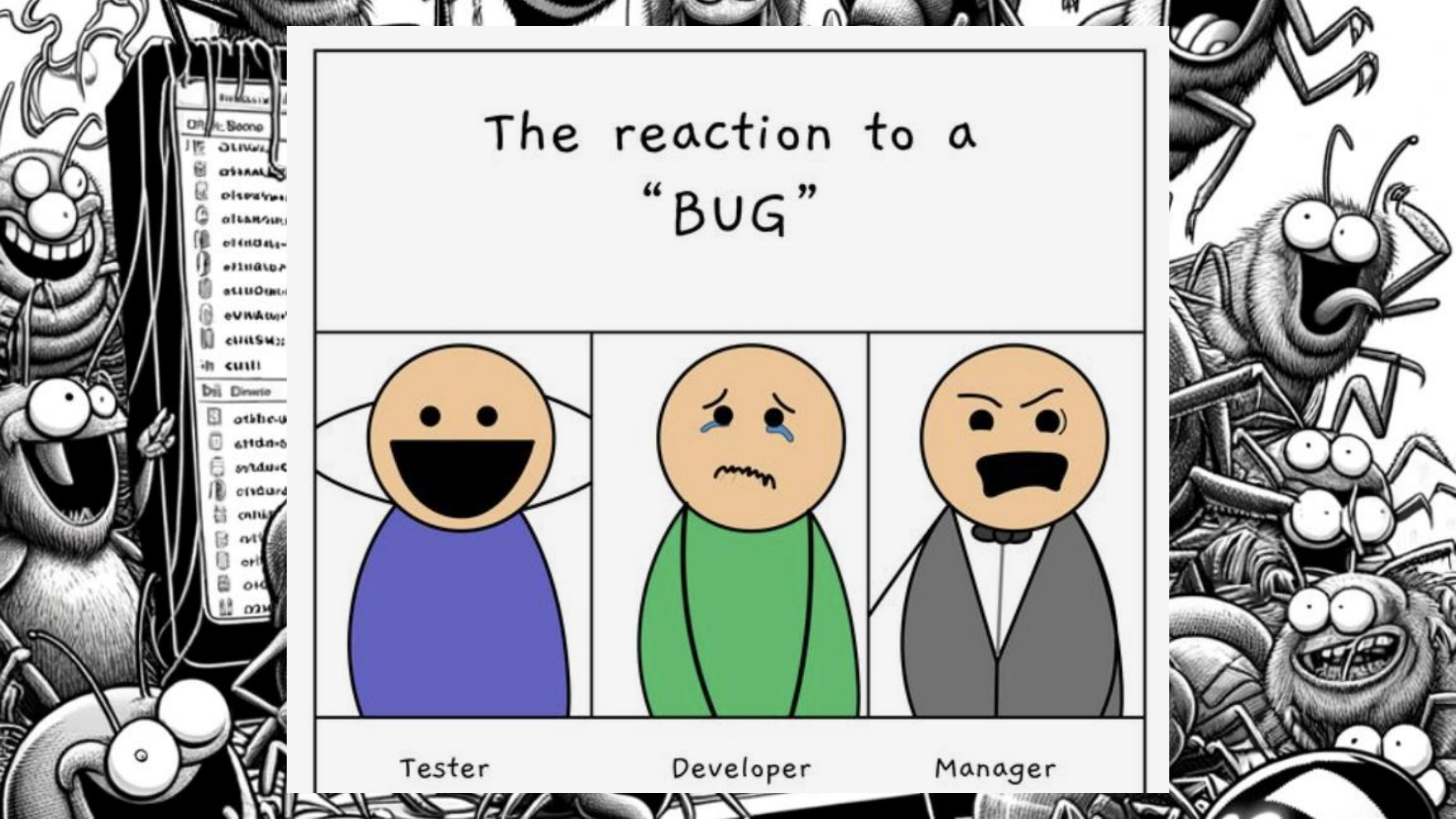
Tester

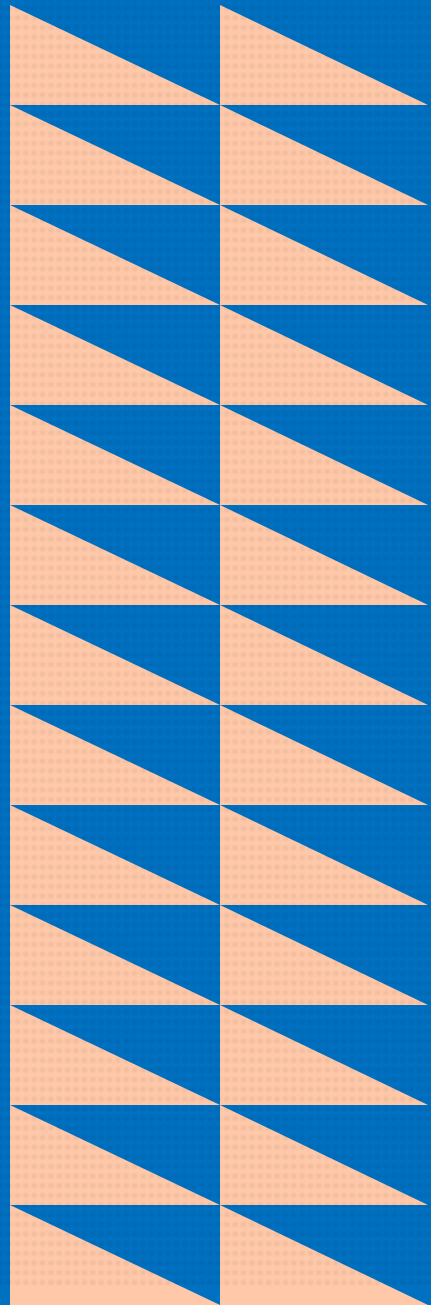


Developer



Manager

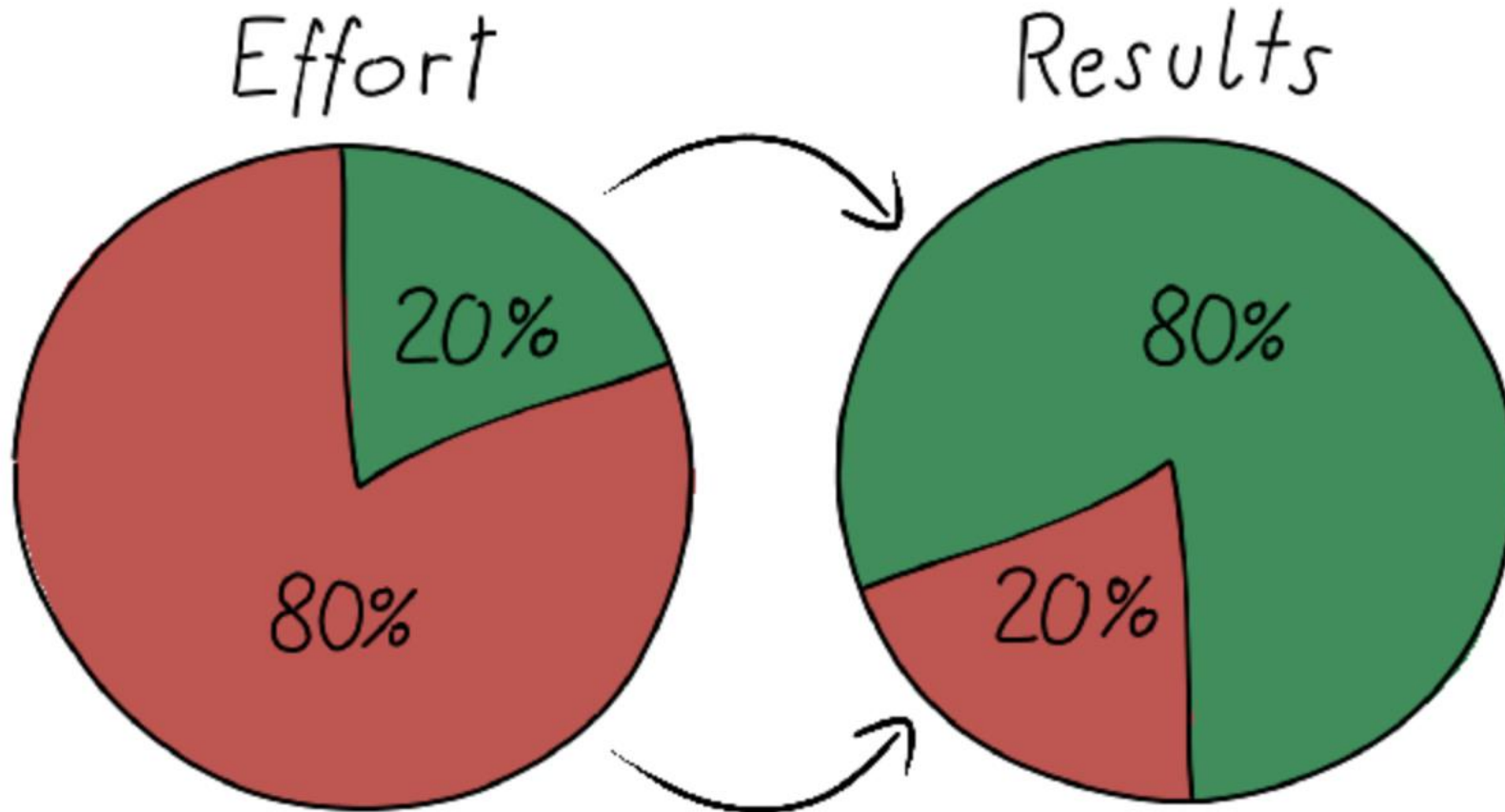




*"Mamy broń, ale nie
wiemy gdzie strzelać!"*

A czas nie jest z gumy..

Zasada Pareto (80/20)



2015 rok,
współpraca
z PWr i UJ

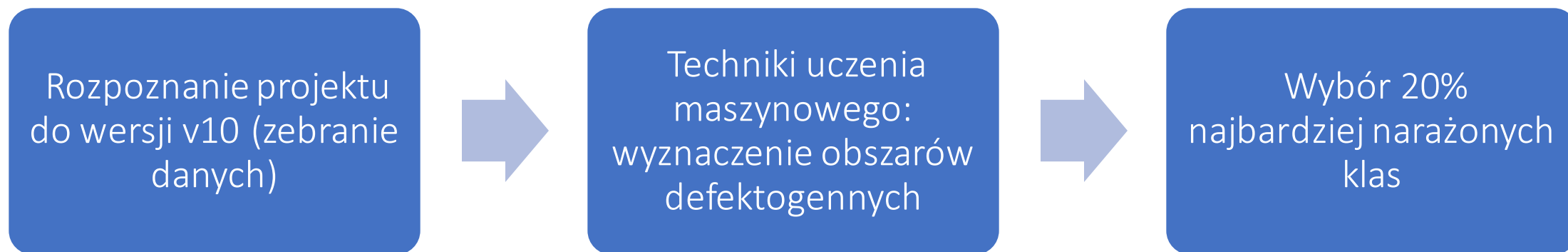


Cel: wyznaczenie 20% "najgroźniejszych" klas

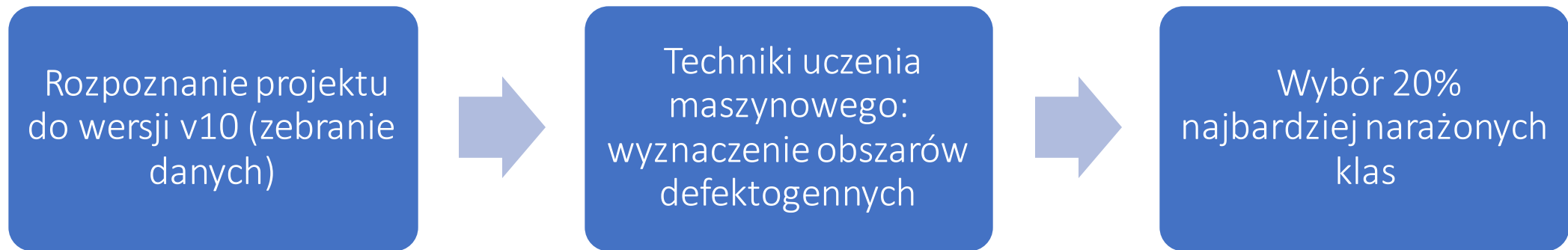
Cel: wyznaczenie 20% "najgroźniejszych" klas

Rozpoznanie projektu do
wersji v10 (zebranie danych)

Cel: wyznaczenie 20% "najgroźniejszych" klas



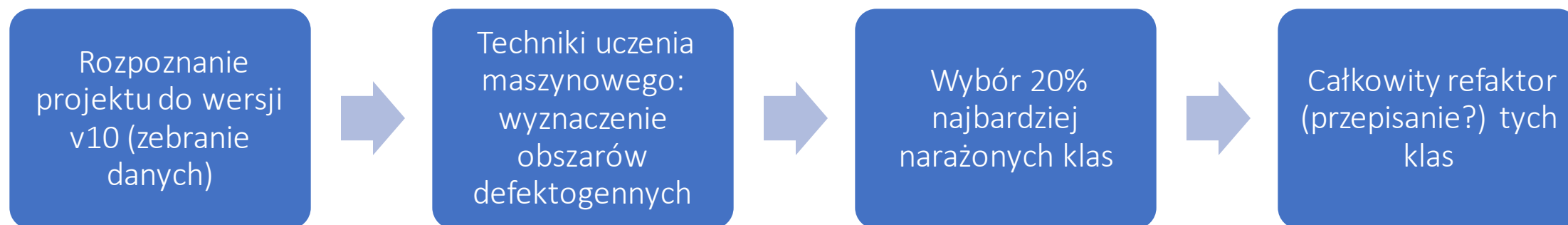
Cel: wyznaczenie 20% "najgroźniejszych" klas



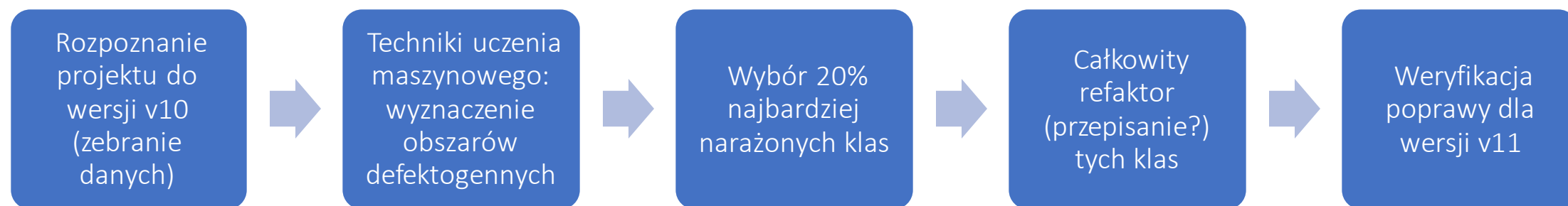
- Jira
- Historia zmian w Git
- Predykcja na bazie "zewnętrznych" repozytoriów i ich statycznych metryk

- <https://github.com/Software-Engineering-Jagiellonian/adept>
- <https://impressivecode.github.io/ic-depress/#/main>

Cel: wyznaczenie 20% "najgroźniejszych" klas



Cel: wyznaczenie 20% "najgroźniejszych" klas



``com.myapp.config.ConfigFactory.java``
``com.myapp.utils.StringUtils.java``
``com.myapp.model.UserProfile.java``
``com.myapp.service.EmailService.java``
``com.myapp.streams.FileHandler.java``
``com.myapp.parser.JsonParser.java``
``com.myapp.encryption.Encryptor.java``
``com.myapp.ui.layout.GridLayout.java``
``com.myapp.client.HttpClient.java``
``com.myapp.database.SqlConnector.java``
``com.myapp.device.DeviceManager.java``
``com.myapp.auth.AuthenticationService.java``
``com.myapp.engine.GameEngine.java``
``com.myapp.mail.EmailDispatcher.java``

Wzorce projektowe

- kreacyjne
- strukturalne
- czynnościowe

Testy

Refaktoryzacja

- przepisanie większości kodu
- SOLID i inne dobre praktyki

Code review

- zwiększenie nacisku na nowo wprowadzone zmiany w tych klasach

``com.myapp.config.ConfigFactory.java``

``com.myapp.utils.StringUtils.java``

``com.myapp.model.UserProfile.java``

``com.myapp.service.EmailService.java``

``com.myapp.streams.FileHandler.java``

``com.myapp.parser.JsonParser.java``

``com.myapp.encryption.Encryptor.java``

``com.myapp.ui.layout.GridLayout.java``

``com.myapp.client.HttpClient.java``

``com.myapp.database.SqlConnector.java``

``com.myapp.device.DeviceManager.java``

``com.myapp.auth.AuthenticationService.java``

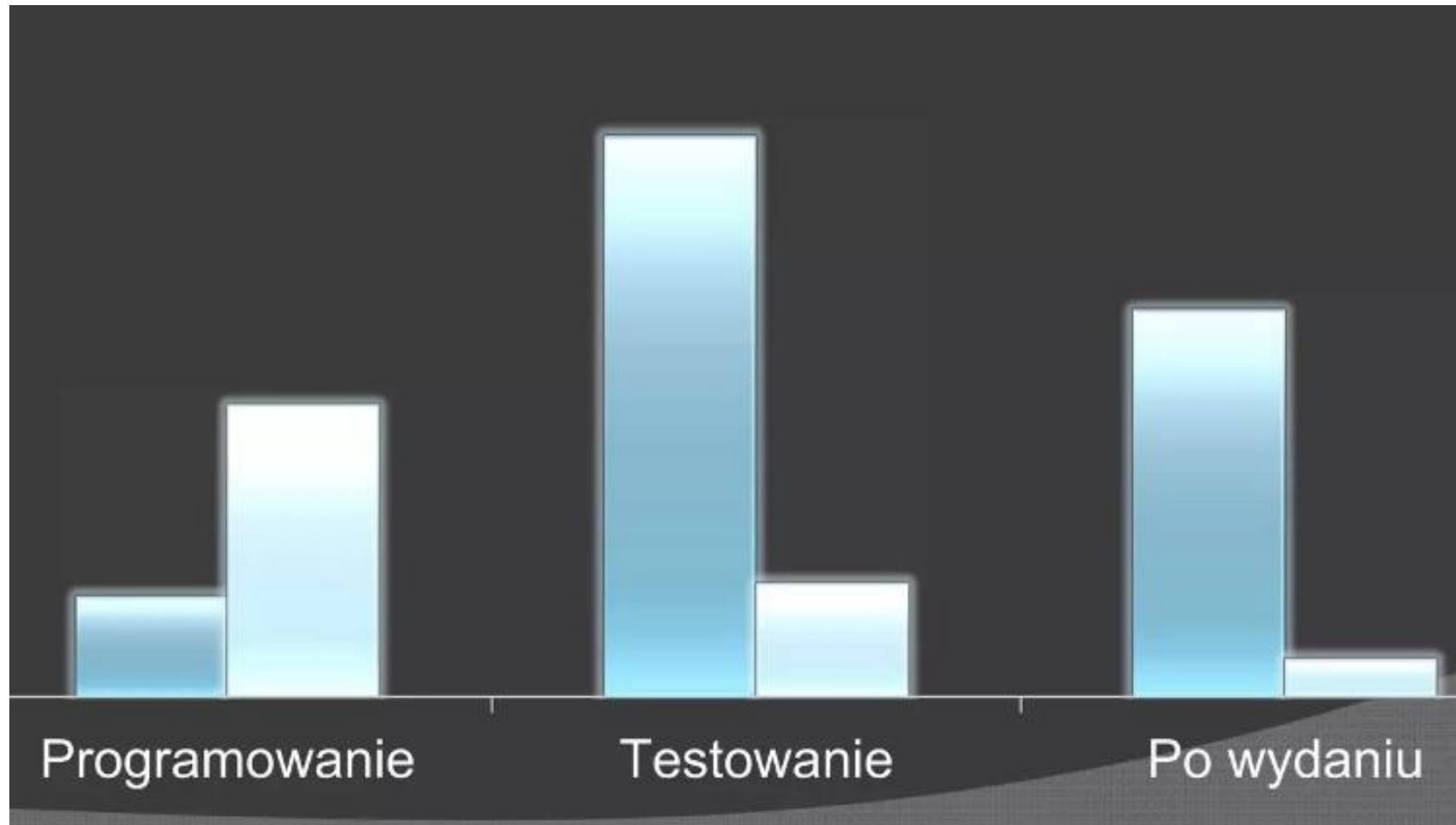
``com.myapp.engine.GameEngine.java``

``com.myapp.mail.EmailDispatcher.java``

Wyniki

Defekty	v10	v11	Różnica
Krytyczne	162	56	-65%
Duże	188	101	-48%
Średnie	248	152	-38%
Małe	200	110	-45%

Koszty zapewnienia jakości



**Narzędzia do predykcji defektów weszły
na stałe do procesu podgrywkowego
kilku zespołów w Volvo IT**

Więcej:

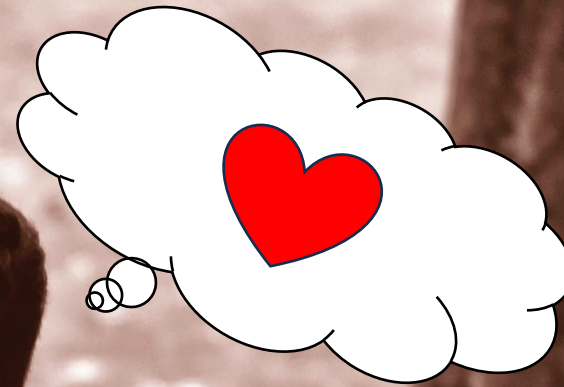
- <https://youtu.be/2j2Vf4v89kl?si=P7HsKPLVNJtHf4Ap>
- https://www.researchgate.net/publication/323424319_Cost_Effectiveness_of_Software_Defect_Prediction_in_an_Industrial_Project
- https://www.researchgate.net/publication/307089968_Assessment_of_the_Software_Defect_Prediction_Cost_Effectiveness_in_an_Industrial_Project

Zatem jaka jest najistotniejsza cecha stojąca za stosowaniem **wzorców projektowych** i **dobrych praktyk pisania kodu** w zespole?

... i wtedy powiedziałam PMowi, że po 6 tygodniach prac wiemy, że w projekcie będzie 36 defektów i jeśli zrobimy release za 4 tygodnie, to w kodzie pozostanie ich wciąż 7. Ich naprawa będzie kosztować 3 razy więcej, niż koszt wydłużenia prac o 2 tygodnie, podczas których znajdziemy 5 z tych bugów...



... i wtedy powiedziałam PMowi, że po 6 tygodniach prac wiemy, że w projekcie będzie 36 defektów i jeśli zrobimy release za 4 tygodnie, to w kodzie pozostanie ich wciąż 7. Ich naprawa będzie kosztować 3 razy więcej, niż koszt wydłużenia prac o 2 tygodnie, podczas których znajdziemy 5 z tych bugów...



Questions?

