Predykcja defektów oprogramowania przy użyciu wybranych technik eksploracji danych

Autor:
mgr inż. DANIEL CZYCZYN-EGIRD

Opiekun naukowy:
dr hab. inż. profesor PK ADAM SŁOWIK

Politechnika Koszalińska 2017

Plan prezentacji

1. Zapewnienie jakości w inżynierii oprogramowania
2. Modele predykcji defektów
3. Sformułowanie problemu
4. Koncepcja proponowanego rozwiązania
5. Podsumowanie
6. Dodatek: Lista osiągnięć

Zapewnienie jakości w inżynierii oprogramowania

Definicja jakości oprogramowania wg ISO/IEC 9126-1:2001 *:

1. **Funkcjonalność**
2. **Niezawodność**
3. **Użyteczność**
4. **Efektywność**
5. **Konserwowalność**
6. **Przenośność**

* 9126-3 Software Engineering – Product quality – Part 3: Internal metrics

Zapewnienie jakości w inżynierii oprogramowania

Definicja jakości oprogramowania w literaturze:

1. **Zgodność z wymaganiami** [Philip B. Crosby]
2. **Przydatność do użytku** [Frank M. Gryna]
3. **BRAK DEFEKTÓW W PRODUKCIE** [Stephen H. Kan]

Zapewnienie jakości w inżynierii oprogramowania

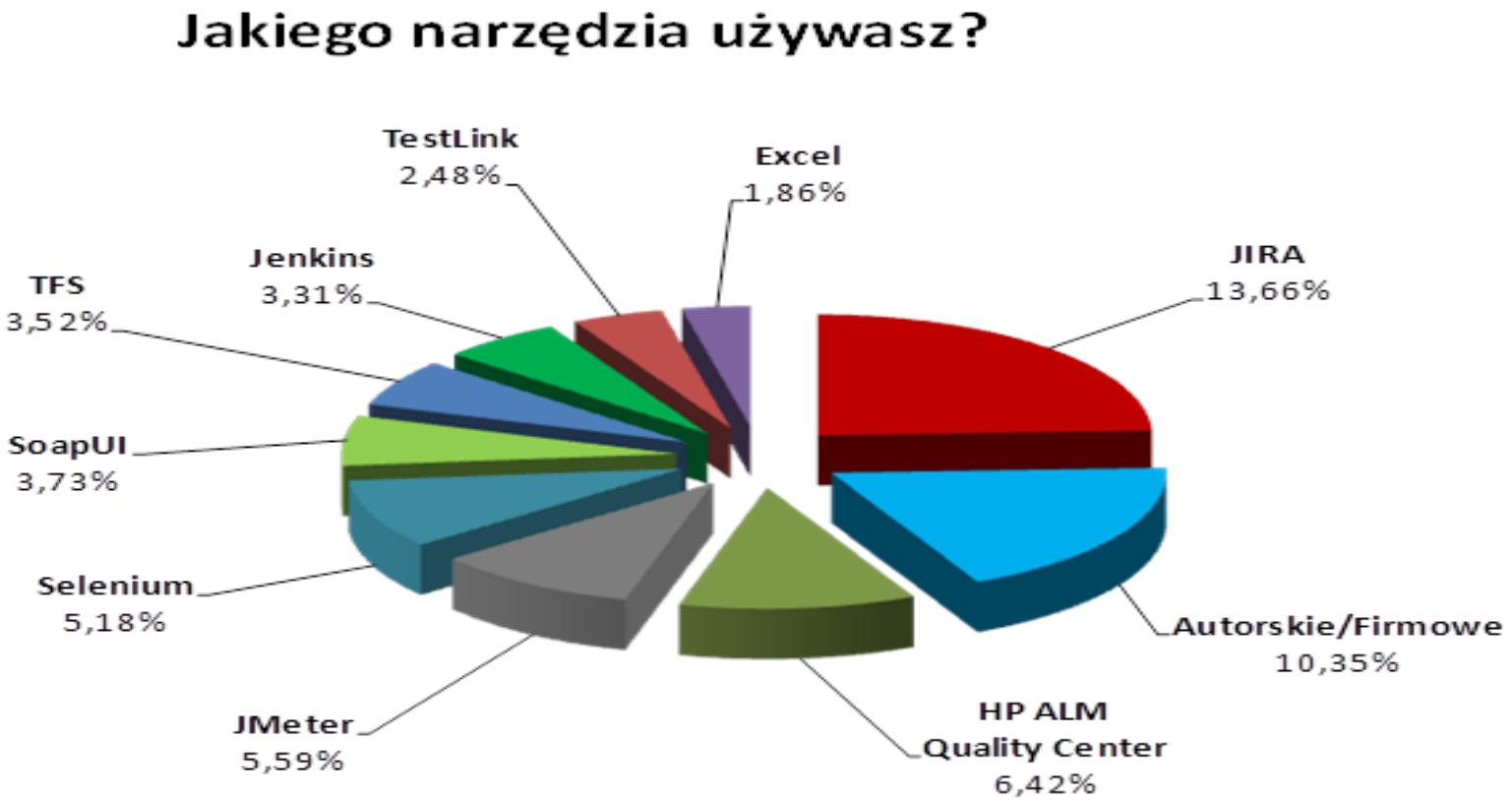
Definicja defektu:

Grafika: http://testerzy.pl/baza-wiedzy/defekt-blad

Przegląd działań wspierających jakość oprogramowania

- **Testowanie oprogramowania**
  - Najobszerniejsze działania zapewniające wysoką jakość
  - Koszty potrafią przekroczyć 50% całkowitych kosztów wytworzenia systemu informatycznego
  - W metodykach zwinnych występuje tzw. programowanie przez testy (TDD), zaczyna się od przygotowania testów, a następnie wykonuje się implementacje
  - Istnieje szereg narzędzi do zarządzania testami i raportowania, do automatyzacji oraz narzędzi do zarządzania incydentami.

Przegląd działań wspierających jakość oprogramowania



http://testerzy.pl/images/narzedzia-w-pracy-testera-02.png

Przegląd działań wspierających jakość oprogramowania

- **Inspekcje oprogramowania**
  - Przegląd formalny stanu prac
  - Statyczna analiza kodu programu lub dokumentacji projektowej
  - Kilkugodzinne spotkanie, na którym autor dokumentu lub fragmentu programu relacjonuje wykonane przez siebie modyfikacje
  - Popularna formuła inspekcji -> tzw. Inspekcje Fagana

Przegląd działań wspierających jakość oprogramowania

- **Modele przyrostu niezawodności**
  - Ukazanie poprawy niezawodności modułu lub systemu jako wyniku poprawiania defektów w czasie
  - Przewidywanie przyszłej niezawodności oprogramowania na podstawie danych historycznych

Szacowania poziomu niezawodności

- charakterystyki kodu (metryki)
- statystyczne korelacje danych (np. z funkcją wykładniczą)

Przegląd działań wspierających jakość oprogramowania

- **Metryki oprogramowania**
  - Miara pewnej właściwości oprogramowania lub jego specyfikacji (miara atrybutu)
  - Funkcja odwzorowująca jednostkę oprogramowania w wartość liczbową [IEEE 1061-1998]

METRYKI PRODUKTU	METRYKI PROCESU
np. - Number of Public Methods (NPM) - Data Access Metric (DAM) - etc.	np. - Number of Revisions (NR) - Number of Distinct Commiters (NDC) - etc.

Modele predykcji defektów

- Narzędzia, które na podstawie wartości metryk danego artefaktu dokonują oszacowania (predykcji) liczby defektów znajdujących się w tym artefakcie
- Modele takie są bardzo przydatne w procesie testowania, kiedy w wyniku ograniczeń budżetowych nie jesteśmy w stanie przetestować wszystkich wytworzonych artefaktów
- Zasada 80:20 *

  - część kodu źródłowego (20%) -> defekty (80%)

* Elaine J. Weyuker, Thomas J. Ostrand, Robert M. Bell, "Do too many cooks spoil the broth? Using the number of developers to enhance defect prediction models", Empirical Software Engineering, 13(5):539–559, 2008

Modele predykcji defektów

- Cel stosowania?

Sformułowanie problemu

- **Dane**
  - Dostęp do kodu źródłowego
  - Dostęp do systemów kontroli wersji (np. SVN)
  - Dostęp do systemów zarządzania projektem i śledzenia defektów (np. JIRA)
- **Ograniczenia**
  - Różnorodność metryk oprogramowania
  - Zastosowanie do obiektowych języków programowania
- **Pytanie**
  - Czy opierając się na kodzie źródłowym oraz danych historycznych dotyczących wykrytych wcześniej defektów, możliwe jest pozyskanie informacji o potencjalnych ukrytych defektach, które nie zostały jeszcze zdiagnozowane?

Sformułowanie problemu

- **Cel**

  Opracowanie oraz zweryfikowanie zestawu modeli klasyfikacyjnych służących do predykcji defektów oprogramowania w projektach programistycznych opartych na paradygmacie programowania obiektowego.

  Wynik klasyfikacji opisujący predykcję możliwych defektów oprogramowania w oparciu o dane wejściowe **przyczyni się do zapewnienia lub utrzymania jakości tworzonego oprogramowania.**

Sformułowanie problemu

- **Teza**

  Hybrydowe klasyfikatory pozwalają na efektywniejszą predykcję defektów oprogramowania w projektach informatycznych tworzonych według paradygmatu programowania obiektowego.

  Są one skuteczniejsze w stosunku do typowo stosowanych modeli klasyfikacyjnych zapewniając tym samym utrzymanie lub poprawę jakości tworzonego oprogramowania.

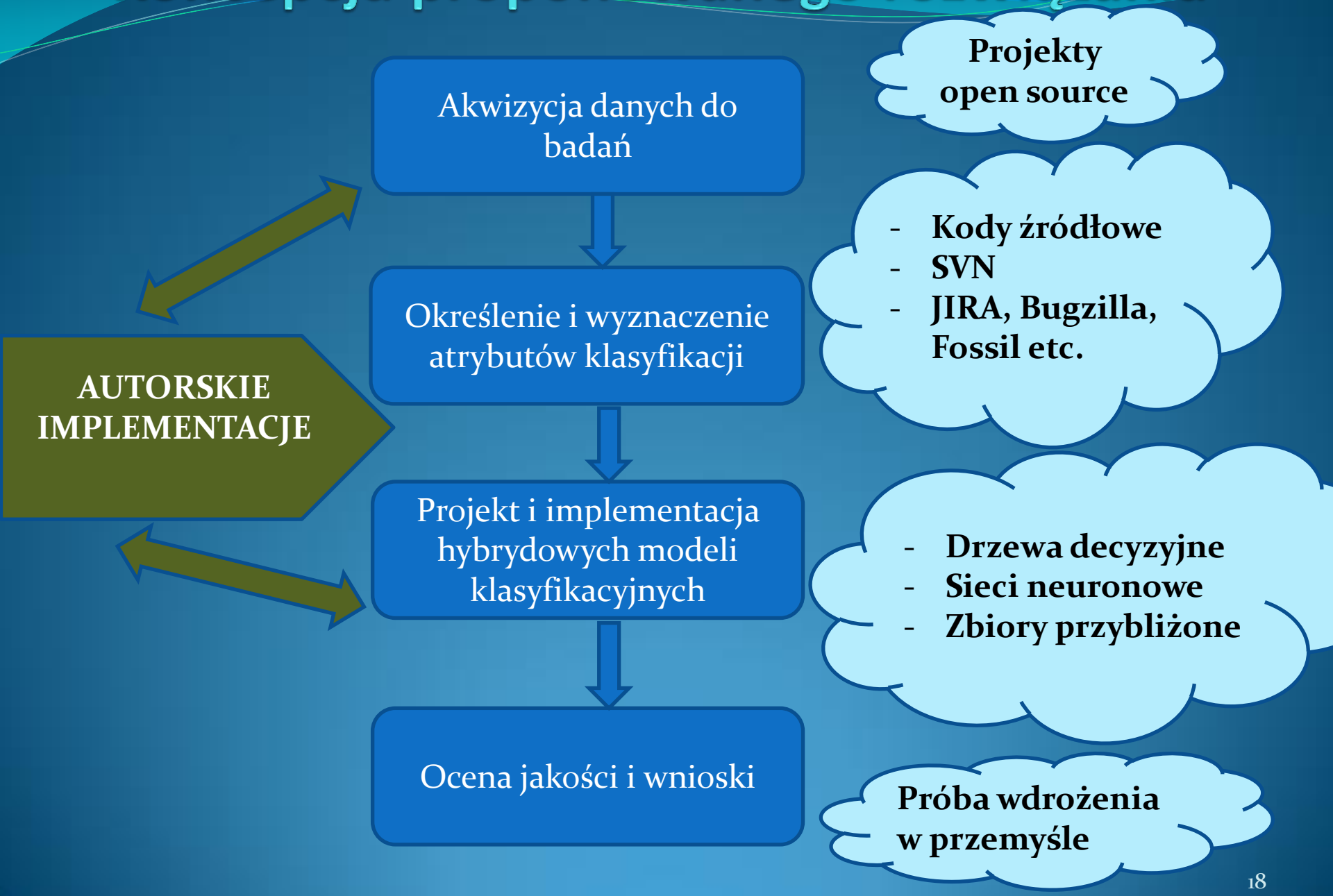
Cele szczegółowe

- Przegląd i porównanie metod budowy modeli predykcji defektów w oprogramowaniu
- Opracowanie hybrydowego (=>BARDZIEJ SKUTECZNEGO) modelu opisującego wystąpienie możliwych defektów
- Zastosowanie wybranych(i zmodyfikowanych autorsko) technik eksploracji danych do budowy modelu (np. do redukcji wymiarowości zadania, odrzucanie najmniej istotnych atrybutów itp.)
- Opracowanie modelu matematycznego predykcji defektów
- Wykonanie eksperymentów testowych (jakość, skuteczność)
- Analiza i ocena otrzymanych wyników, sformułowanie wniosków końcowych

Hipotezy robocze

- Ocena metryk oprogramowania wraz z analizą danych historycznych pozwolą na predykcję defektów w badanym oprogramowaniu.
- Zaproponowane rozwiązania techniczne pozwolą na częściową automatyzację procesu akwizycji danych.
- Modele predykcji defektów będą możliwe do zastosowania dla technologii obiektowych.

Koncepcja proponowanego rozwiązania



Przykładowa predykcja

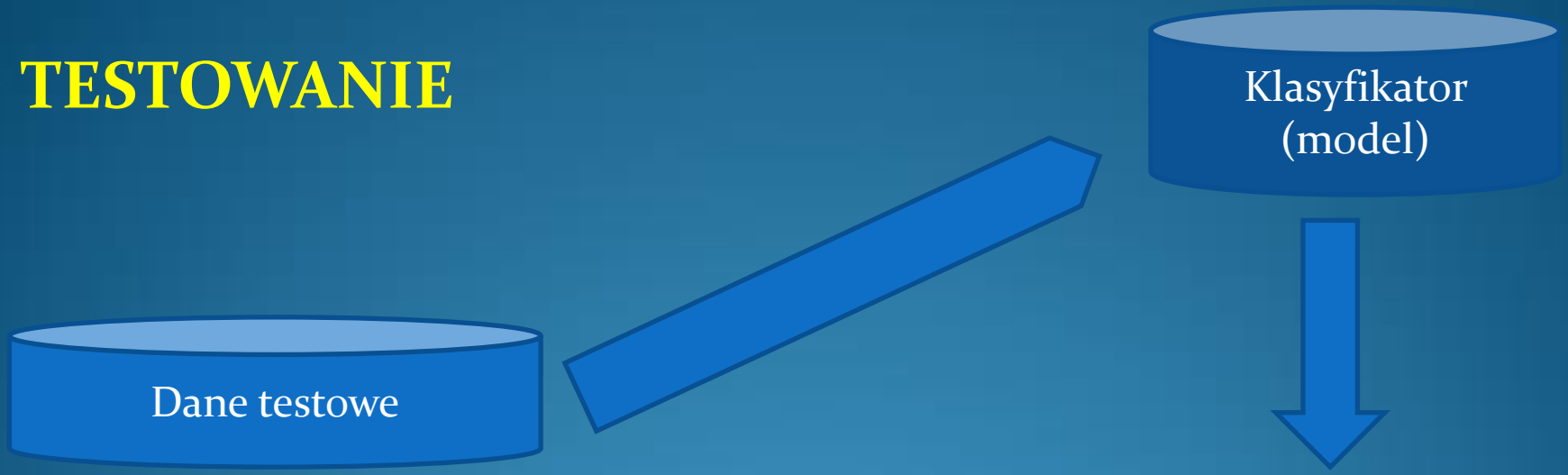
UCZENIE

Liczba klas w pliku	Śr. ilość metod w klasie	Ryzyko defektu
12	5	NISKIE
15	20	WYSOKIE
19	13	NISKIE
50	8	WYSOKIE
7	17	WYSOKIE

if Liczba klas w pliku **> 20**
or Śr. il. metod w klasie **> 15**
then Ryzyko = **„WYSOKIE”**

Przykładowa predykcja

- **TESTOWANIE**



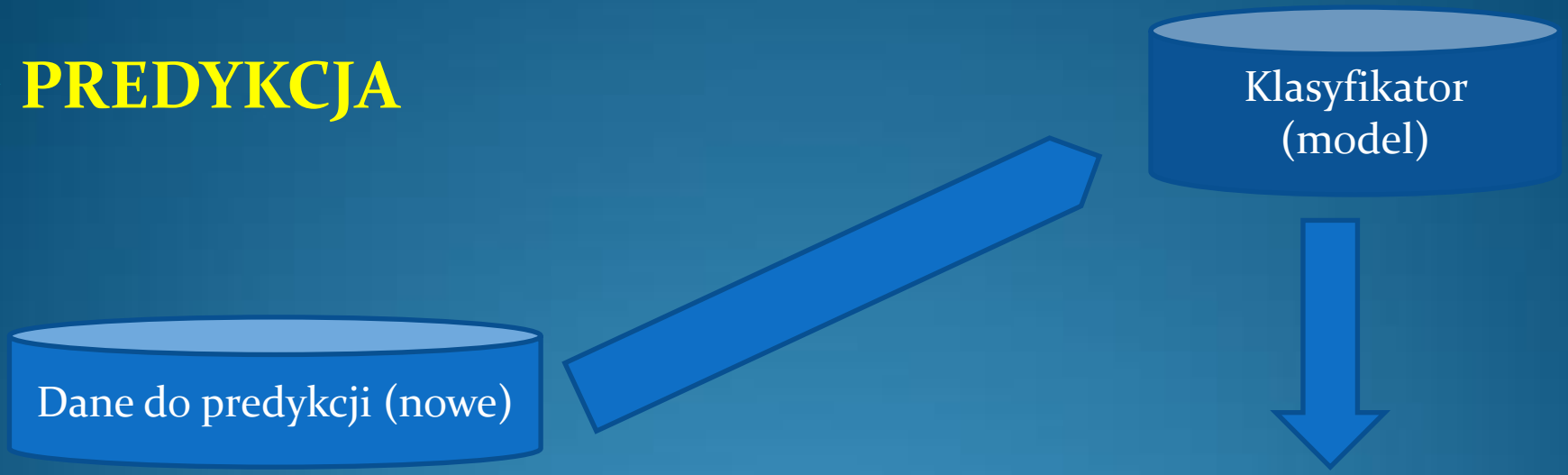
Liczba klas w pliku	Śr. ilość metod w klasie	Ryzyko defektu
18	7	NISKIE
22	12	WYSOKIE
16	6	WYSOKIE
37	8	WYSOKIE

Ryzyko defektu
NISKIE
WYSOKIE
NISKIE
WYSOKIE

Dokładność = 3/4 = 75%

Przykładowa predykcja

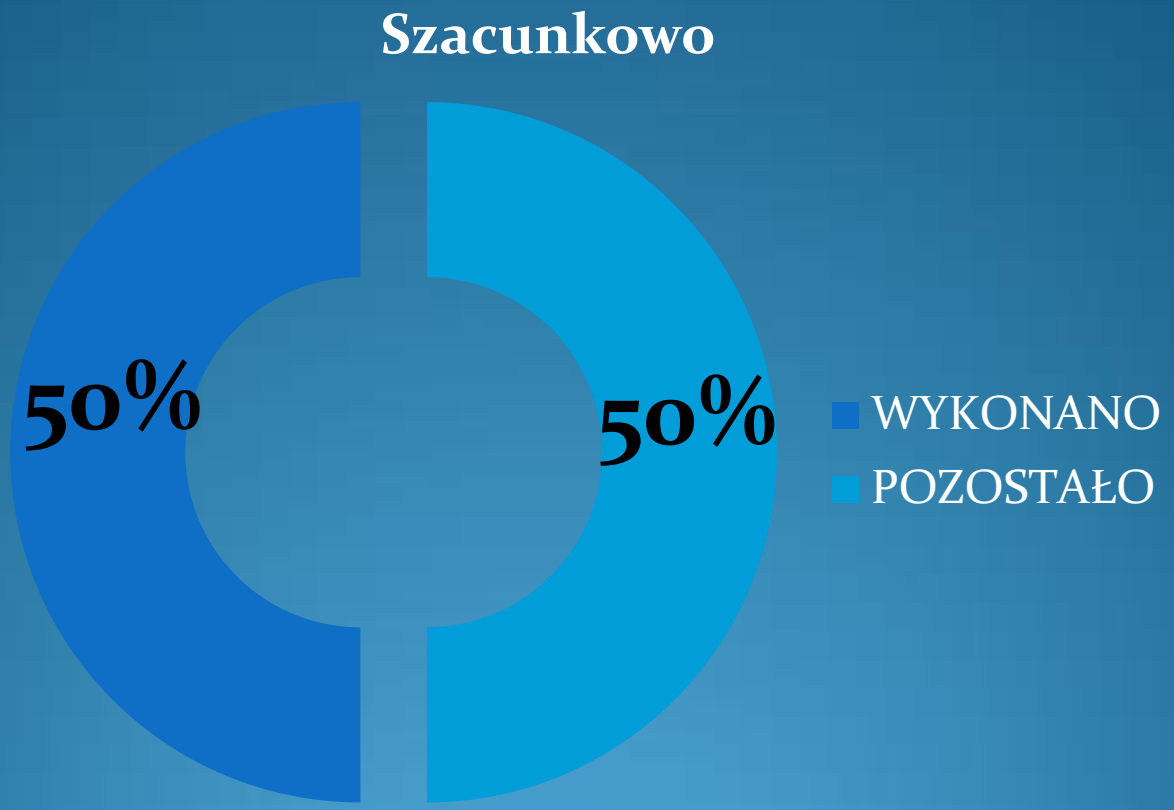
- **PREDYKCJA**



Liczba klas w pliku	Śr. ilość metod w klasie	Ryzyko defektu
100	5	?
30	12	?
16	6	?
67	17	?

Ryzyko defektu
WYSOKIE
WYSOKIE
NISKIE
WYSOKIE

Aktualny stan prac



Przewidywany termin zamknięcia przewodu doktorskiego:
rok akademicki 2019/2020

Plan dysertacji

- **Zapewnienie jakości w inżynierii oprogramowania**
  - Wstęp
  - Aspekty dotyczące jakości oprogramowania
  - Opis metryk oprogramowania
  - Modele predykcji defektów
  - Cel i teza pracy
  - Opis rozdziałów
- **Przegląd stanu wiedzy o modelach predykcji defektów**
  - Metryki produktu
  - Metryki procesu
  - Opis czynników istotnych dla procesu predykcji defektów
  - Predykcja pomiędzy projektami
- **Przegląd stanu wiedzy o wybranych technikach eksploracji danych**
  - Opis wybranych technik
- **Pozyskiwanie danych do badań**
  - Informacje o przebadanych projektach
  - Przykłady użytych metryk produktu
  - Przykłady użytych metryk procesu
  - Opis narzędzi zastosowanych do zbierania i analizowania danych

Plan dysertacji c.d.

- **Zastosowanie hybrydowych modeli predykcji defektów**
  - Definicja hybrydowego modelu predykcji defektów
  - Opis eksperymentu
  - Wyniki eksperymentu
  - Walidacja uzyskanych wyników
  - Podsumowanie
- **Zastosowanie uzyskanych wyników w praktyce**
  - Motywacja zastosowania
  - Wdrożenie modelu predykcji defektów na konkretnym przykładzie
  - Podsumowanie efektów wdrożenia
- **Podsumowanie**
  - Część badawcza
  - Część implementacyjna
- **Literatura**

Podsumowanie prac

- **Publikacje w języku polskim:**

1. Daniel Czyczyn-Egird: „**Eksploracja danych na podstawie szacowania informacji z wykorzystaniem regresji liniowej**”, Modele Inżynierii Teleinformatyki, 2015
2. Daniel Czyczyn-Egird, Rafał Wojszczyk: „**Zastosowanie technik eksploracji danych na przykładzie badania popularności wzorców projektowych w serwisie społecznościowym Stackoverflow.com**”, Zeszyty Naukowe WEiI, 2016
3. Daniel Czyczyn-Egird, Rafał Wojszczyk: „Predykcja ataków DDoS za pomocą technik eksploracji danych” Zeszyty Naukowe WEiI (przyjęte do druku), 2017

Podsumowanie prac

- **Publikacje w języku angielskim:**

1. Daniel Czyczyn-Egird, Rafał Wojszczyk: „**Determining the Popularity of Design Patterns Used by Programmers Based on the Analysis of Questions and Answers on Stackoverflow.com Social Network**”, **Springer International Publishing, 2016**
2. Daniel Czyczyn-Egird, Rafał Wojszczyk: „**The effectiveness of data mining techniques in the detection of DDoS attacks**”, **Springer International Publishing, 2017**
3. Daniel Czyczyn-Egird, Rafał Wojszczyk: „**DDoS Attacks Prediction in a Simulation Environment by means of Data Mining Techniques**”, Studia Informatica Politechnika Śląska, 2017

Podsumowanie prac

- **Konferencje:**

1. „Krajowa Konferencja Studentów i Młodych Pracowników Nauki” w Unieściu w dniach 20-22 maja 2015
2. „Krajowa Konferencja Studentów i Młodych Pracowników Nauki” w Szczecinie w dniach 20-21 kwietnia 2016
3. „23rd International Science Conference on Computer Networks CN2016” w Brunowie w dniach 14-17 czerwca 2016
4. „Krajowa Konferencja Studentów i Młodych Pracowników Nauki” w Mielnie w dniach 20-22 maja 2017
5. **„14th International Conference on Distributed Computing and Artificial Intelligence (DCAI'17)” Polytechnic of Porto - Porto (Portugal) | 21st-23rd June, 2017**

Dziękuję za uwagę

Bibliografia

- Barry W. Boehm, Philip N. Papaccio. Understanding and controlling software costs. IEEE Transactions on Software Engineering, 14:1462–1477, Październik 1988.
- Elaine J. Weyuker, Thomas J. Ostrand, Robert M. Bell. Do too many cooks spoil the broth? Using the number of developers to enhance defect prediction models. Empirical Software Engineering, 13(5):539–559, 2008.
- Elaine J. Weyuker, Thomas J. Ostrand, Robert M. Bell. Using developer information as a factor for fault prediction. PROMISE '07: Proceedings of the Third International Workshop on Predictor Modelsin Software Engineering, Washington, DC, USA, 2007. IEEE Computer Society
- Stephen H. Kan. Metrics and Models in Software Quality Engineering. Addison-Wesley Longman PublishingCo., Inc., Boston, MA, USA, 2002.
- Cagatay Catal, Banu Diri. Review: A systematic review of software fault prediction studies. Expert Systems with Applications, USA, 2009.
- Lionel C. Briand, Jurgen Wust, John W. Daly, D. Victor Porter. Exploring the relationship between design measures and software quality in object-oriented systems. Journal of Systems and Software, 51:245–273, Maj 2000

DODATEK: MDA

- **MDA - Model Driven Architecture**

  Zbiór metod porządkujących proces tworzenia systemów komputerowych opartych na budowie modeli i ich transformacji.

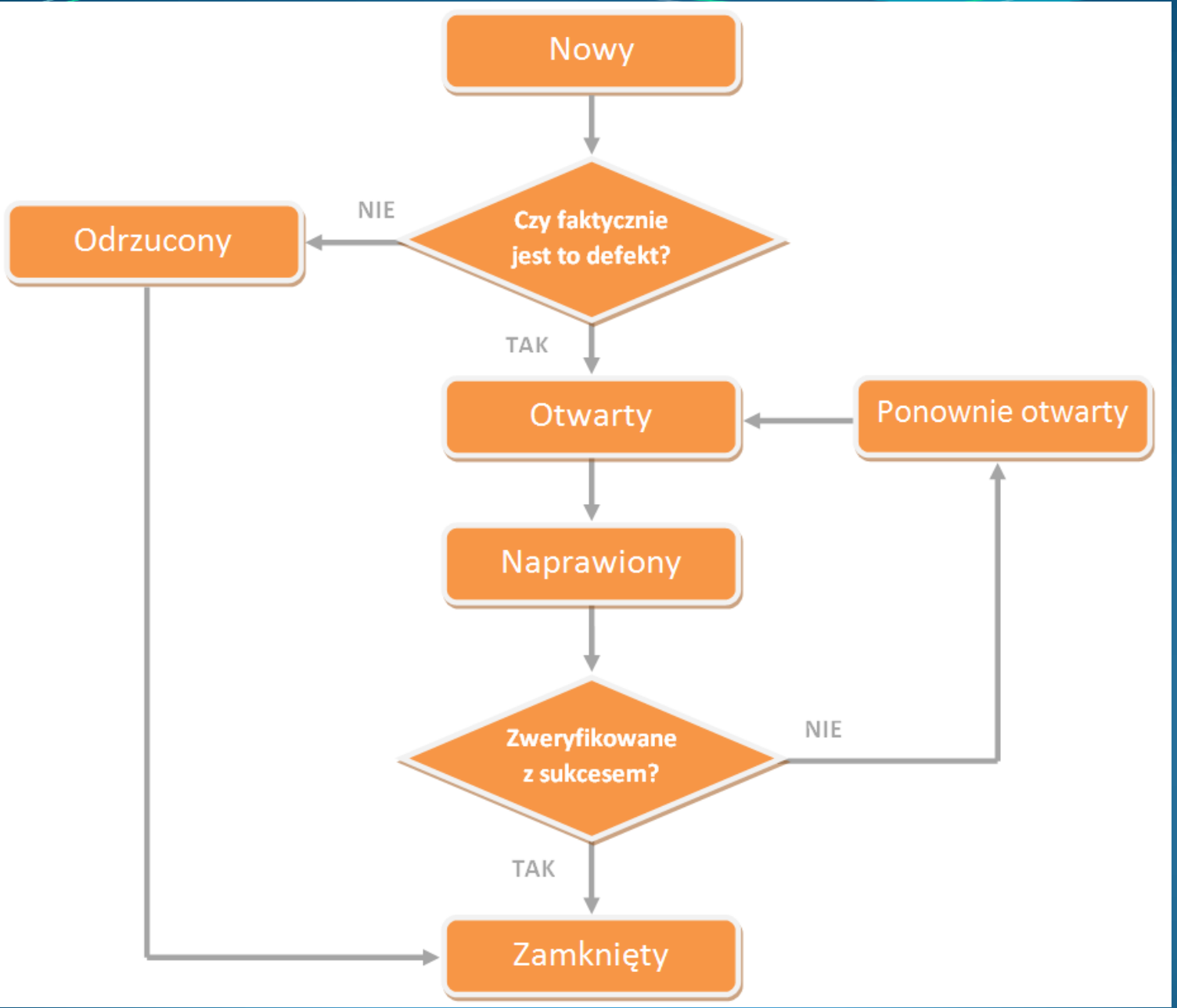
  Cykl życia oprogramowania:

  1. **CIM** (ang. Computational Independent Model) - model domenowy, środowisko, czy też wymagania.
  2. **PIM** (ang. Platform-Independent Model) - perspektywa logiczna (projektowa) systemu.
  3. **PSM** (ang. Platform-Specific Model) – opis systemu w perspektywie implementacyjnej
  4. **PDM** (ang. Platform Description Model) – opis szczegółów technicznych perspektywy wdrożeniowej

DODATEK: METODY KLASYFIKACJI

- **PRZYKŁADOWE METODY**
  - Klasyfikacja poprzez indukcję drzew decyzyjnych
  - Klasyfikatory Bayes'owskie
  - Sieci Neuronowe
  - Analiza statystyczna
  - Metaheurystyki (np. algorytmy genetyczne)
  - Zbiory przybliżone
  - k-NN – k-najbliższe sąsiedztwo

DODATEK: Proces „życia” defektu w systemie



DODATEK: Definicja defektu w JIRA