



dr inż. Michał Malinowski
michal.malinowski@uth.edu.pl



[https://www.linkedin.com/in/
michał-malinowski-malkon/](https://www.linkedin.com/in/michał-malinowski-malkon/)

Programowanie, Testowanie, Błędy oraz Reverse engineering i no-code

Wykład 06

INŻYNIERIA
OPROGRAMOWANIA



Zagadnienia (2h)

- [Programowanie zgodne ze standardami](#)
- [Style programowania](#)
- [Automatyzacja testów i kontrola jakości](#)
- [Zarządzanie błędami](#)
- [Reverse engineering](#)
- [no-code](#)

Programowanie zgodne ze standardami

Programowanie zgodne ze standardami

to praktyka tworzenia kodu zgodnie z określonymi wytycznymi i normami, które mają na celu zapewnienie jakości, interoperacyjności i utrzymania kodu.

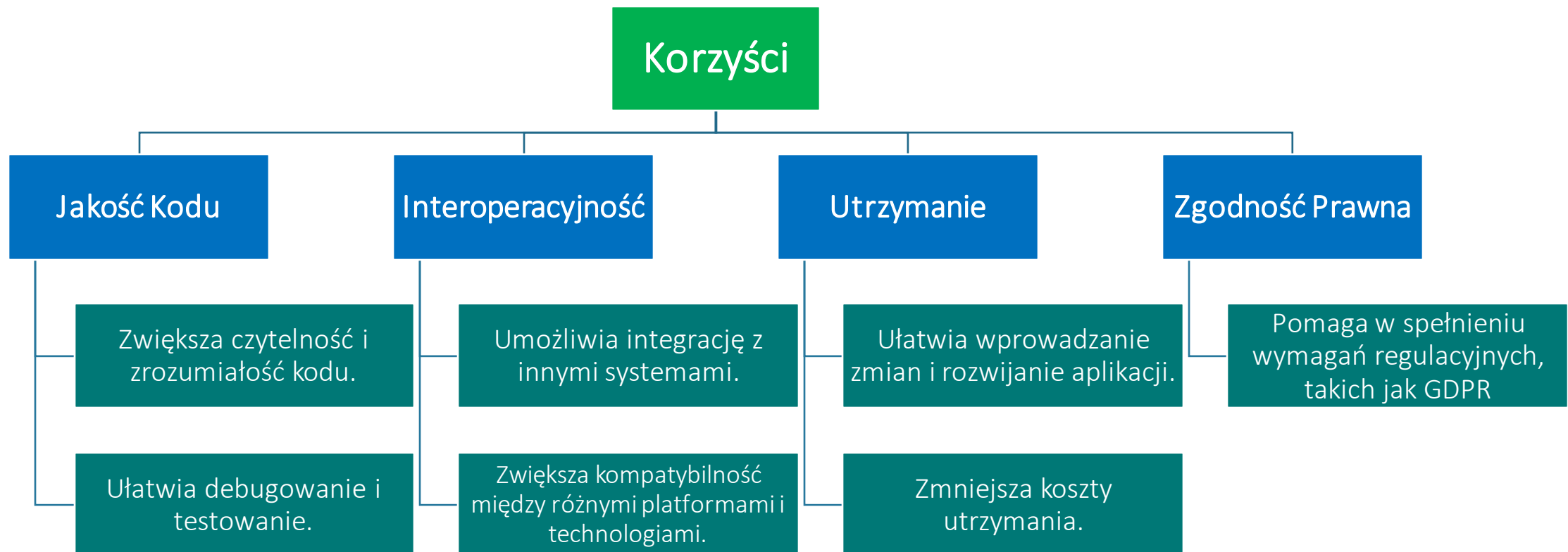
Korzyści

Łatwiejsze debugowanie,
większa efektywność, lepsza
jakość kodu.

Wady

Czas potrzebny na zapoznanie
się ze standardami,
ograniczenia kreatywności.

Programowanie zgodne ze standardami



Programowanie zgodne ze standardami

Standardy

PEP 8 (Python)

Oficjalny zestaw wytycznych dotyczących stylu kodowania w Pythonie, obejmujący zasady formatowania, nazewnictwa i komentarzy.

Java Code Conventions

Zestaw konwencji dla języka Java, który obejmuje zasady dotyczące nazewnictwa, formatowania i komentarzy.

Google's JavaScript Style Guide

Zestaw wytycznych dla języka JavaScript, zalecany przez Google, obejmujący zasady dotyczące formatowania, nazewnictwa i komentarzy.

Airbnb JavaScript Style Guide

Popularny zestaw wytycznych dla JavaScript, często stosowany w projektach open-source.

PSR (PHP-FIG Standards Recommendations)

Zestaw standardów dla języka PHP, obejmujący zasady dotyczące autoloadingu, formatowania kodu i innych aspektów.

W3C Standards

Standardy dla technologii webowych, takich jak HTML, CSS i JavaScript, zalecane przez World Wide Web Consortium (W3C).

C# Coding Conventions

Wytyczne dotyczące pisania kodu w języku C#, zalecane przez Microsoft, obejmujące zasady dotyczące nazewnictwa, formatowania i komentarzy.

Style programowania

Styl programowania

to zorganizowany zestaw konwencji, zasad i praktyk, które określają sposób strukturyzowania, formatowania i dokumentowania kodu źródłowego w celu zapewnienia jego spójności, czytelności i łatwości utrzymania.



Określenie "styl programowania" może być zastąpione terminami takimi jak:

- "konwencje kodowania" (Coding Conventions)
- "standardy programistyczne" (Programming Standards)
- "wytyczne programistyczne" (Coding Guidelines)
- "paradygmat programowania" (Programming Paradigm)

Określenie "programowanie" może być zastąpione terminami takimi jak:

- "implementacja" (Implementation)
- "kodowanie" (Coding)
- "skryptowanie" (Scripting)

Style programowania

Programowanie proceduralne

to styl programowania, w którym kod jest podzielony na funkcje (procedury). Każda funkcja wykonuje określoną czynność. Programowanie proceduralne jest często używane w językach programowania, takich jak C, C++, Java, Python czy JavaScript.

C

```
int main() {  
    int a = 10;  
    int b = 20;  
  
    int c = add(a, b);  
  
    printf("c = %d\n", c);  
  
    return 0;  
}  
  
int add(int a, int b) {  
    return a + b;  
}
```

Style programowania

Programowanie obiektowe

to styl programowania, w którym kod jest podzielony na obiekty. Obiekt to jednostka kodu, która zawiera dane i funkcje. Programowanie obiektowe jest często używane w językach programowania, takich jak Java, C#, Python czy JavaScript.

Java

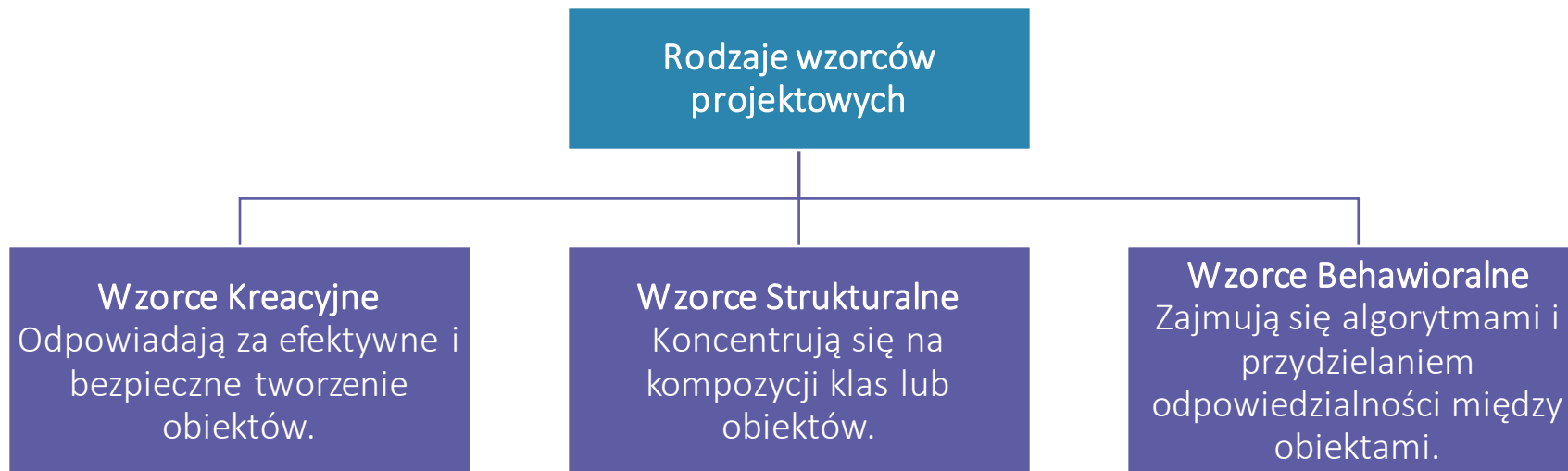
```
public class Calculator {  
  
    public static void main(String[] args) {  
        Calculator calc = new Calculator();  
  
        int c = calc.add(10, 20);  
  
        System.out.println("c = " + c);  
    }  
  
    public int add(int a, int b) {  
        return a + b;  
    }  
}
```

Style programowania

Wzorce projektowe (design patterns)

to sprawdzone rozwiązania do często występujących problemów w projektowaniu oprogramowania. Służą jako szablony, które można dostosować do konkretnych potrzeb projektu. Są opisem rozwiązania, a nie jego implementacją.

Wzorce projektowe stosowane są w projektach wykorzystujących programowanie obiektowe.



Style programowania

Programowanie deklaratywne

to styl programowania, w którym kod skupia się na opisie pożądanego wyniku, a nie na tym, w jaki sposób jest on uzyskiwany. Oznacza to, że programista nie musi martwić się o szczegóły implementacji. Programowanie deklaratywne jest często używane w językach, takich jak SQL, HTML czy CSS

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Deklarative HTML</title>
</head>
<body>
  <h1>This is a heading</h1>
  <p>This is a paragraph</p>
  <ul>
    <li>This is a list item</li>
    <li>This is another list item</li>
  </ul>
</body>
</html>
```



"U mnie działa!"

Pierwsze prawo informatyka

Automatyzacja testów i kontrola jakości

Automatyzacja testów

to proces używania oprogramowania do wykonania testów na kodzie źródłowym w celu identyfikacji błędów i niedoskonałości.



Zalety Automatyzacji:

- Szybkość: Testy są wykonywane znacznie szybciej.
- Dokładność: Eliminuje błędy ludzkie.
- Regresja: Możliwość szybkiego wykonania testów regresyjnych.

Kiedy Automatyzować?:

- Testy są często wykonywane.
- Testy są złożone i czasochłonne.
- Wysoki koszt błędów.

Automatyzacja testów i kontrola jakości

Kontrola jakości

to zestaw działań mających na celu zapewnienie, że oprogramowanie spełnia określone standardy i wymagania.

Metody Kontroli Jakości:

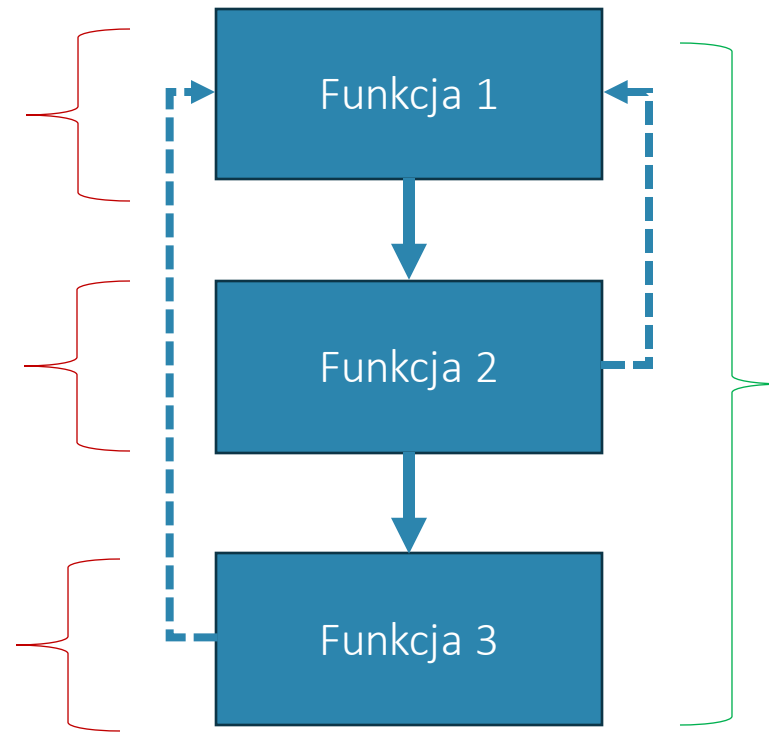
- Code Review: Ręczne sprawdzenie kodu przez innych programistów.
- Testy Jednostkowe: Sprawdzenie pojedynczych jednostek kodu.
- Testy Integracyjne: Sprawdzenie interakcji między różnymi jednostkami kodu.
- Testy Akceptacyjne: Sprawdzenie, czy system spełnia wymagania biznesowe.

Automatyzacja testów i kontrola jakości

Testy jednostkowe (unit testy)

to testy, które skupiają się na pojedynczych "jednostkach" kodu, takich jak funkcje czy metody.

Przykłady: testowanie funkcji matematycznych, testowanie metod CRUD* w aplikacjach bazodanowych.



Testy integracyjne (integration tests)

to testy, które skupiają się na interakcjach między różnymi komponentami systemu.

Przykłady: testowanie interakcji między front-endem a back-endem, testowanie interakcji między różnymi mikroserwisami

* *CRUD to akronim od angielskich słów Create, Read, Update, Delete, które są podstawowymi operacjami wykonywanymi na danych w systemach bazodanowych i aplikacjach.*

Zarządzanie błędami

Zarządzania Błędami

to proces identyfikacji, klasyfikacji i naprawy błędów w oprogramowaniu.

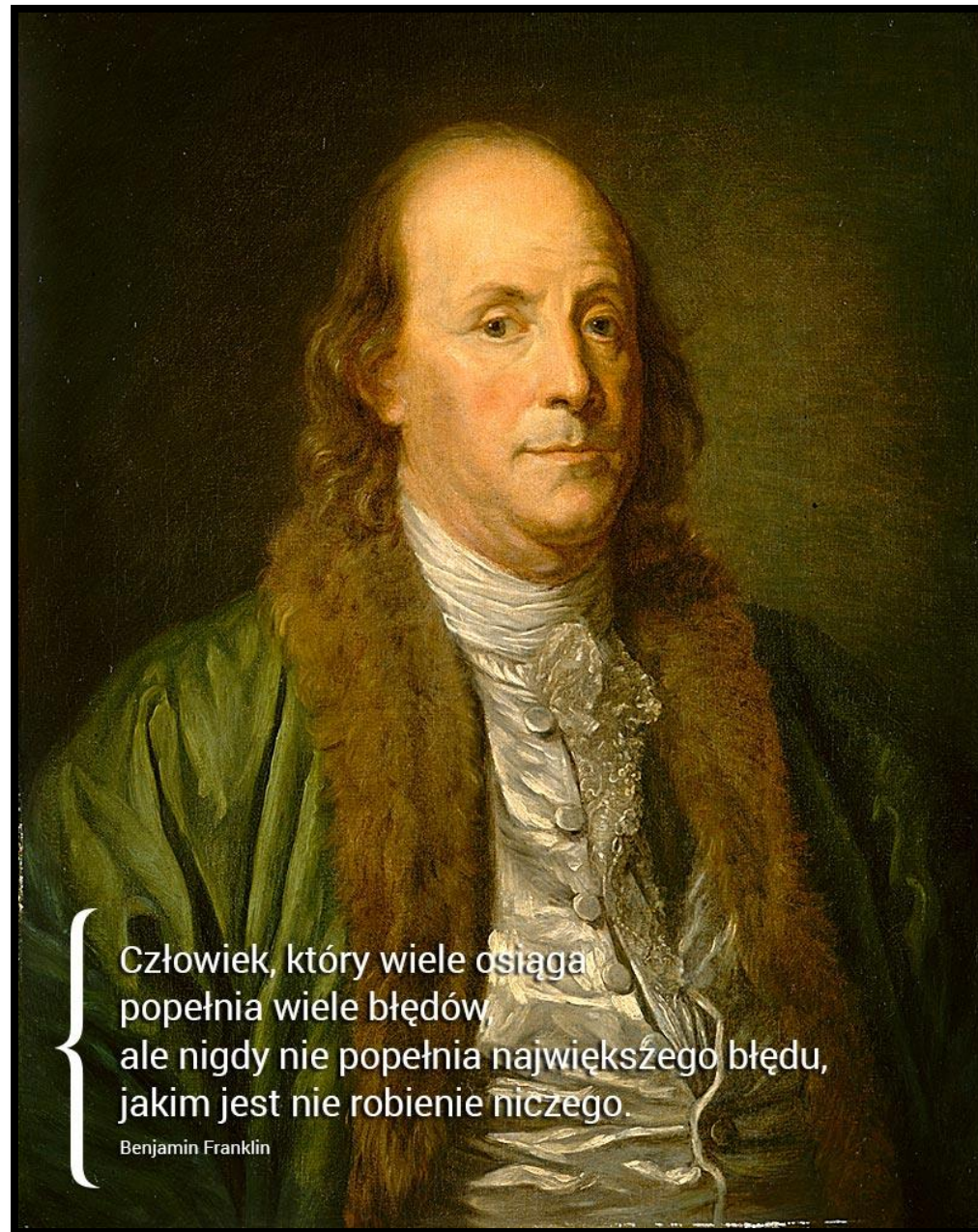
Metody Kontroli Jakości:

- Code Review: Ręczne sprawdzenie kodu przez innych programistów.
- Testy Jednostkowe: Sprawdzenie pojedynczych jednostek kodu.
- Testy Integracyjne: Sprawdzenie interakcji między różnymi jednostkami kodu.
- Testy Akceptacyjne: Sprawdzenie, czy system spełnia wymagania biznesowe.



Tylko ten nie popełnia błędów,
kto nic nie robi.

Napoléon Bonaparte

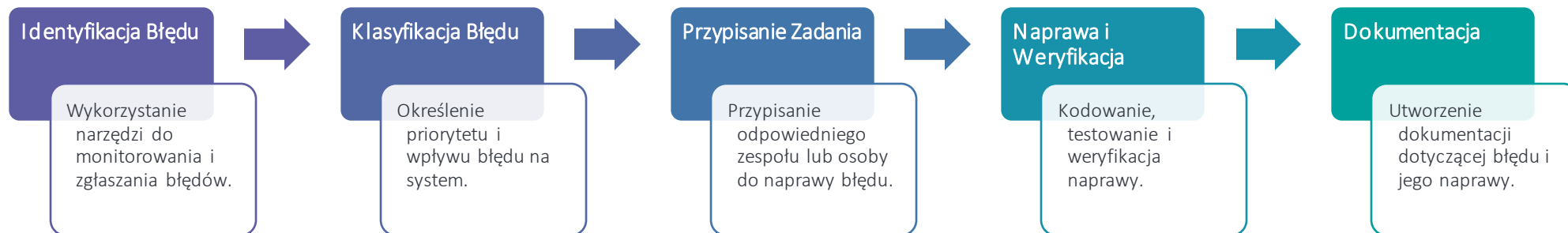


Człowiek, który wiele osiąga
popełnia wiele błędów,
ale nigdy nie popełnia największego błędu,
jakim jest nie robienie niczego.

Benjamin Franklin

Zarządzanie błędami

Etapy Procesu Zarządzania Błędami



Zarządzanie błędami

Rodzaje Błędów w Oprogramowaniu

Błędy Składniowe (Syntax Errors)

Błędy w strukturze kodu, łatwe do wykrycia przez kompilatory.

Błędy Czasu Wykonania (Runtime Errors)

Błędy występujące podczas działania programu, np. pętla w pętli

Błędy Logiczne (Logical Errors)

Błędy w algorytmach i logice programu, trudne do wykrycia.

Błędy Semantyczne (Semantic Errors)

Błędy związane z nieprawidłowym użyciem elementów języka programowania.

Błędy Zasobów (Resource Errors)

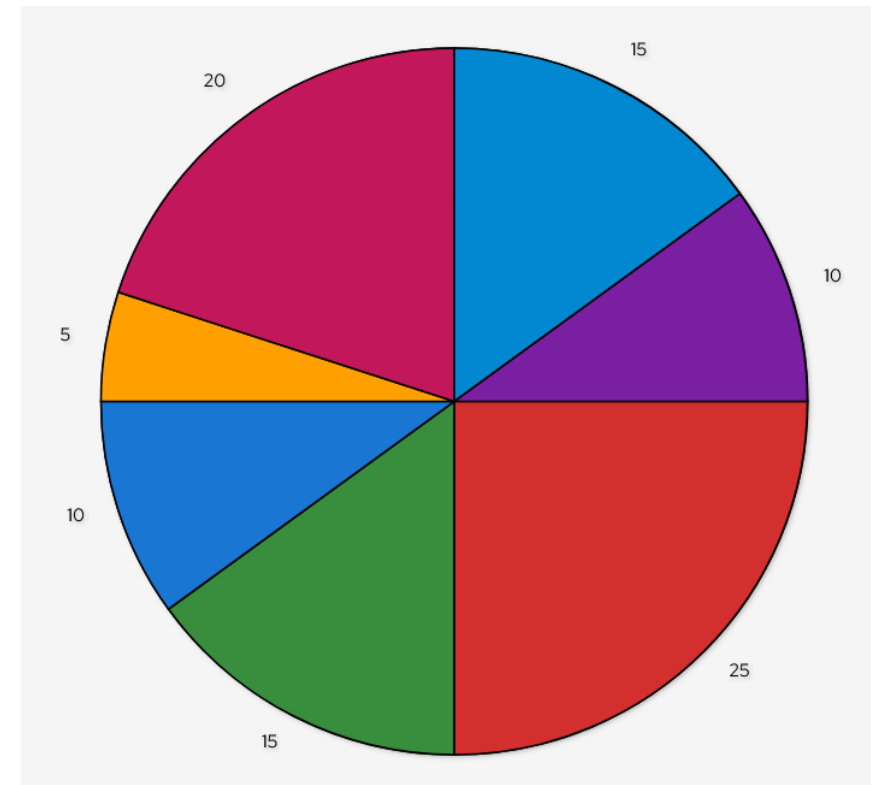
Błędy związane z niedostępnością zasobów, np. brak pamięci.

Błędy Interfejsu (Interface Errors)

Błędy w komunikacji między różnymi modułami lub systemami.

Błędy Konfiguracyjne (Configuration Errors)

Błędy związane z nieprawidłową konfiguracją systemu lub środowiska.



Statystyki Występowania Rodzajów Błędów

Reverse engineering

Reverse engineering (inżynieria odwrotna)

w informatyce odnosi się do procesu analizy produktu, często oprogramowania, w celu określenia jego składników i funkcjonowania, a następnie tworzenia reprezentacji tego produktu w innej formie lub na wyższym poziomie abstrakcji.

Proces ten jest przeciwny do tradycyjnego inżynierii, gdzie tworzy się produkt na podstawie określonych specyfikacji.

Warto zaznaczyć, że reverse engineering może być ograniczony z powodów prawnych, w tym praw autorskich i licencji. Dlatego zawsze ważne jest, aby przestrzegać przepisów prawa i umów licencyjnych podczas podejmowania takich działań

Reverse engineering

Główne cele reverse engineering

Zrozumienie kodu źródłowego

Jeśli organizacja posiada oprogramowanie, ale nie ma dostępu do jego kodu źródłowego, reverse engineering może pomóc w zrozumieniu jego działania.

Wykrywanie błędów i luki w zabezpieczeniach

Specjaliści ds. bezpieczeństwa często używają technik reverse engineering, aby znaleźć i naprawić potencjalne luki w zabezpieczeniach oprogramowania.

Integracja z innym oprogramowaniem

Jeśli oprogramowanie nie posiada dokumentacji lub API, reverse engineering może pomóc w zrozumieniu, jak można je zintegrować z innymi systemami.

Migracja lub aktualizacja systemów

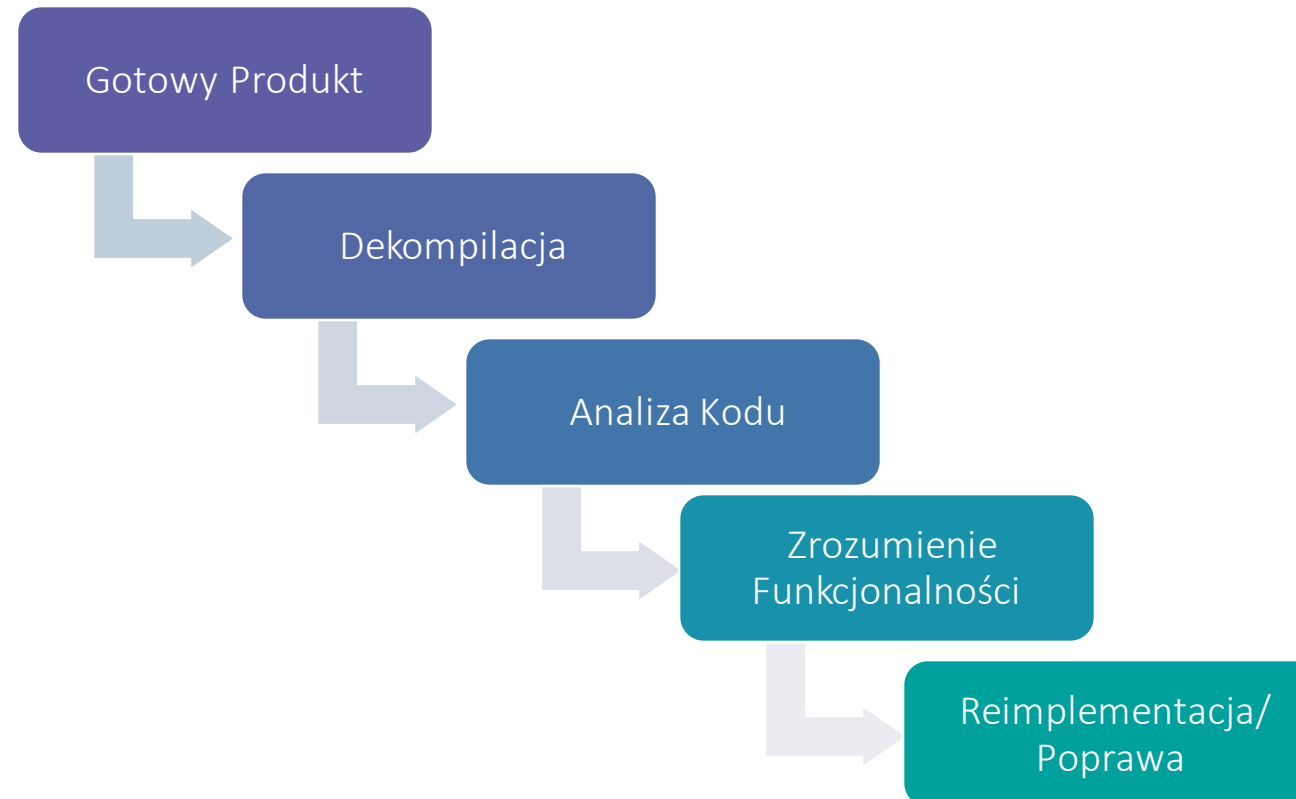
Jeśli istniejące systemy mają być zastąpione lub zaktualizowane, ale brakuje pełnej dokumentacji, reverse engineering może pomóc w zrozumieniu ich działania.

Analiza konkurencyjna

Firmy mogą analizować produkty konkurencji w celu zrozumienia ich funkcji i możliwości.

Reverse engineering

Etapy reverse engineering



Reverse engineering

Reverse engineering a Cyberbezpieczeństwo

Analiza malware

Specjaliści od bezpieczeństwa używają reverse engineering, aby zrozumieć, jak działa złośliwe oprogramowanie, jakie ma możliwości i jak może być wykryte oraz usunięte.

Znajdowanie i łatanie luk

Przez analizę oprogramowania można zidentyfikować słabości w zabezpieczeniach i luki, które mogą być następnie załatane, aby wzmocnić bezpieczeństwo.

Weryfikacja zabezpieczeń

Reverse engineering jest wykorzystywane do testowania zabezpieczeń aplikacji, w tym mechanizmów szyfrowania, uwierzytelniania i ochrony prywatności.

Odzyskiwanie utraconych danych

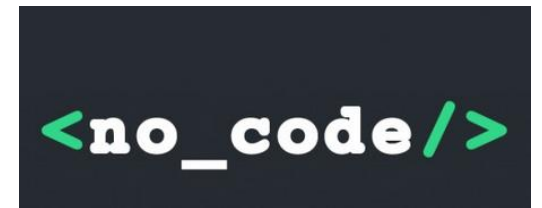
Może być stosowane do odzyskiwania utraconego kodu źródłowego lub dokumentacji, co jest ważne w przypadku zarządzania kryzysowego.

no-code

no-code

to termin odnoszący się do platform i narzędzi, które umożliwiają tworzenie aplikacji komputerowych i cyfrowych rozwiązań bez konieczności pisania tradycyjnego kodu programistycznego.

Są to zazwyczaj graficzne interfejsy użytkownika (GUI), gdzie użytkownicy mogą przeciągać i upuszczać komponenty, aby budować aplikacje, procesy lub systemy.



Łagodniejsza wersja to **low-code**

Tworzenie Stron
i Sklepów
Internetowych

Automatyzacja
Procesów
Biznesowych

Rozwój Aplikacji
Wewnętrznych
i Narzędzi
Biznesowych

no-code

Zalety no-code

Łatwość Użycia

No-code umożliwia tworzenie aplikacji bez głębokiej wiedzy technicznej. Użytkownicy mogą korzystać z interfejsów typu "przeciągnij i upuść", co znacznie upraszcza proces tworzenia.

Szybszy Rozwój

Pozwala na szybsze tworzenie i wdrażanie aplikacji, co jest szczególnie ważne w szybko zmieniających się warunkach biznesowych.

Obniżone Koszty

Redukcja potrzeby zaawansowanych umiejętności programistycznych może obniżyć koszty rozwoju i utrzymania aplikacji.

Demokratyzacja Tworzenia Aplikacji

Umożliwia niemal każdemu tworzenie aplikacji, co otwiera drzwi do innowacji i kreatywności osób, które nie są programistami.

Elastyczność i Skalowalność

Wiele narzędzi no-code oferuje łatwe skalowanie i dostosowywanie aplikacji do zmieniających się potrzeb biznesowych.

no-code

Wady no-code

Ograniczenia Dostosowania

Chociaż no-code oferuje wiele funkcji, może być ograniczający w przypadku bardziej złożonych lub specyficznych wymagań. Dostosowanie poza standardowymi opcjami może być trudne lub niemożliwe.

Problemy z Wydajnością

Aplikacje tworzone przy użyciu narzędzi no-code mogą nie być tak wydajne jak te stworzone tradycyjnymi metodami programowania, szczególnie w przypadku obsługi dużej liczby użytkowników lub danych.

Zależność od Dostawcy

Użytkowanie platformy no-code często wiąże się z zależnością od dostawcy narzędzia, zarówno pod względem możliwości technicznych, jak i cen.

Bezpieczeństwo i Prywatność

Aplikacje no-code mogą stwarzać wyzwania związane z bezpieczeństwem i prywatnością, zwłaszcza jeśli platforma nie przestrzega rygorystycznych standardów branżowych.

Skalowalność

Mimo że narzędzia no-code są skalowalne do pewnego stopnia, mogą pojawić się problemy przy bardzo dużych lub złożonych projektach.