Metodyki zwinne wytwarzania oprogramowania

Wykład 6

Marcin Młotkowski

16 listopada 2016

Plan wykładu

Terminologia

Project

Zorganizowane działanie zmierzające do osiągnięcia zamierzonego celu.

Pierwszy wykład

Terminologia

Project

Zorganizowane działanie zmierzające do osiągnięcia zamierzonego celu.

Pierwszy wykład

Desgin

Rysunek, plan,

Sposoby projektowania oprogramowania

- diagramy UML
- dokumentacja
- kod źródłowy

Pytanie

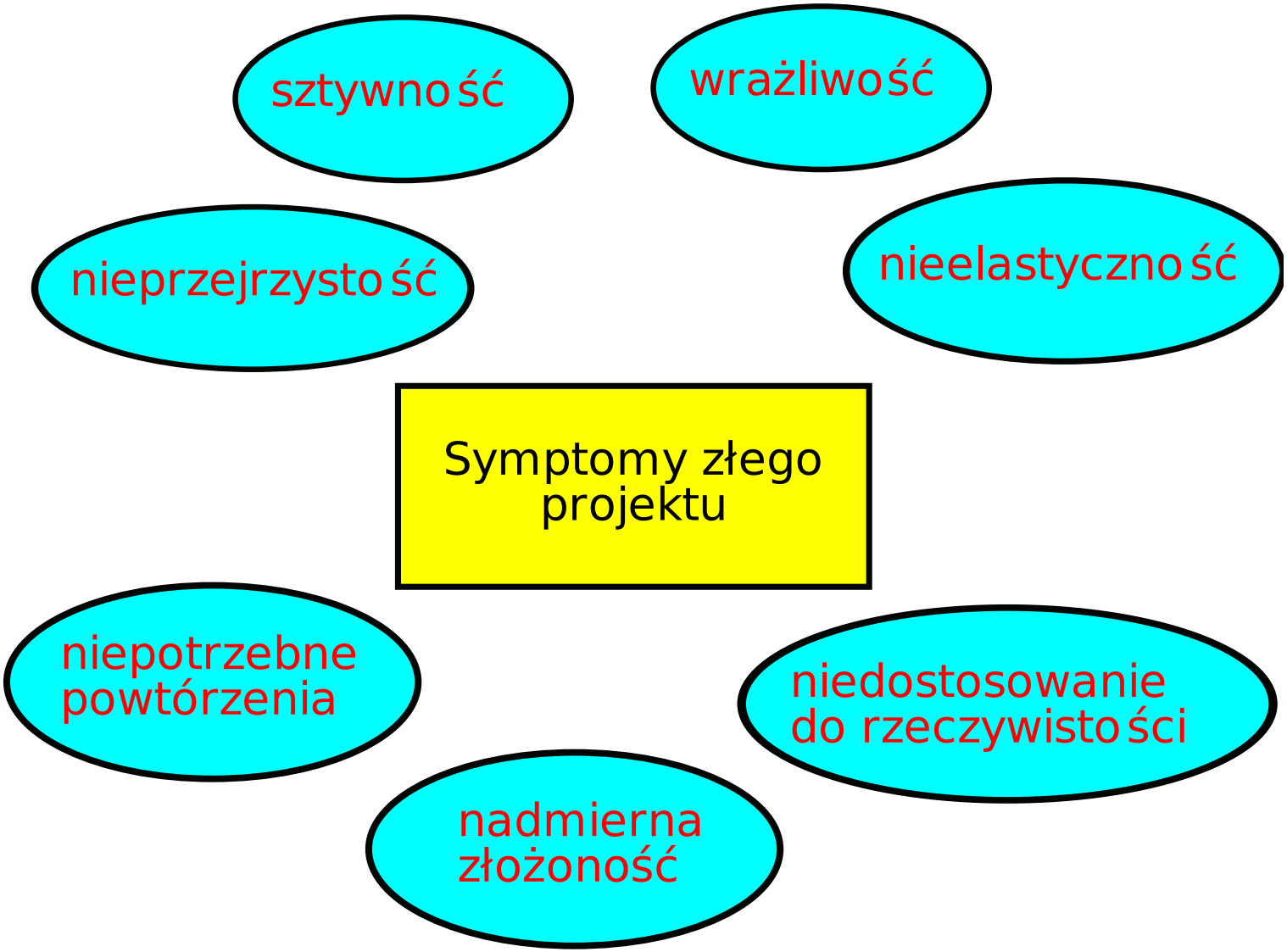
Jeśli program działa, to czy warto się przejmować jego (programu) jakością?

Pytanie

Jeśli program działa, to czy warto się przejmować jego (programu) jakością?

Zmiany, zmiany zmiany.

Dlaczego projekty są złe



Sztywność

Z życia wzięte

Drobna zmiana w specyfikacji (tak to w projektach zwinnych bywa ;-)

Sztywność

Z życia wzięte

Drobna zmiana w specyfikacji (tak to w projektach zwinnych bywa ;-)

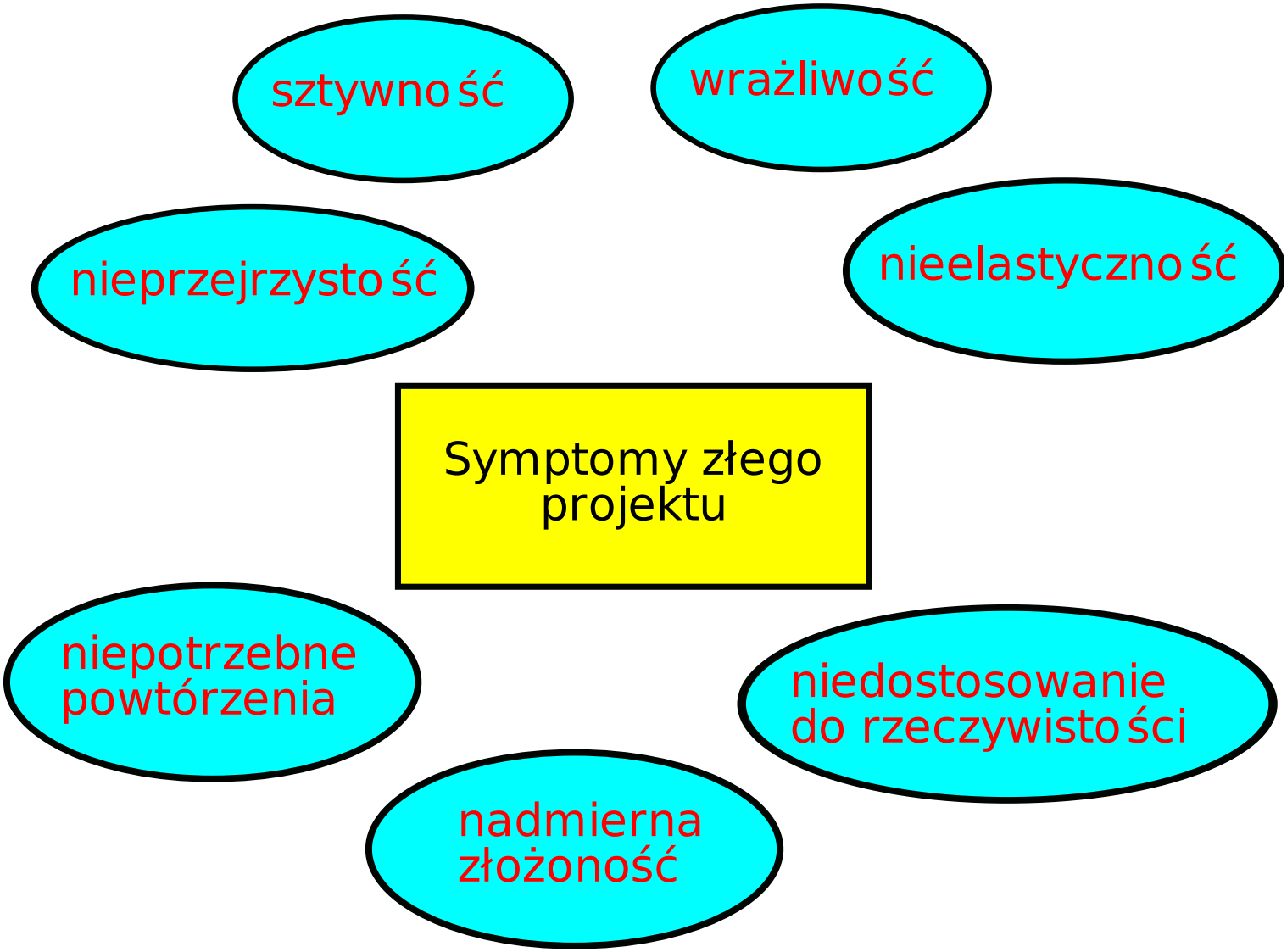
Rezultat

Wymaganie kaskadowych zmian w modułach zależnych i mnóstwo czasu na poprawki.

Wrażliwość

Drobna zmiana powoduje błędy nawet w odległych modułach.

Przypomnienie



Nieelastyczność

Brak możliwości powtórnego użycia kodu.

Niedostosowanie do rzeczywistości

Niedostosowanie oprogramowania do projektu

Implementacja zmian "kusi" pójściem na skróty, nie zawsze zgodnie z zasadami dobrego projektowania.

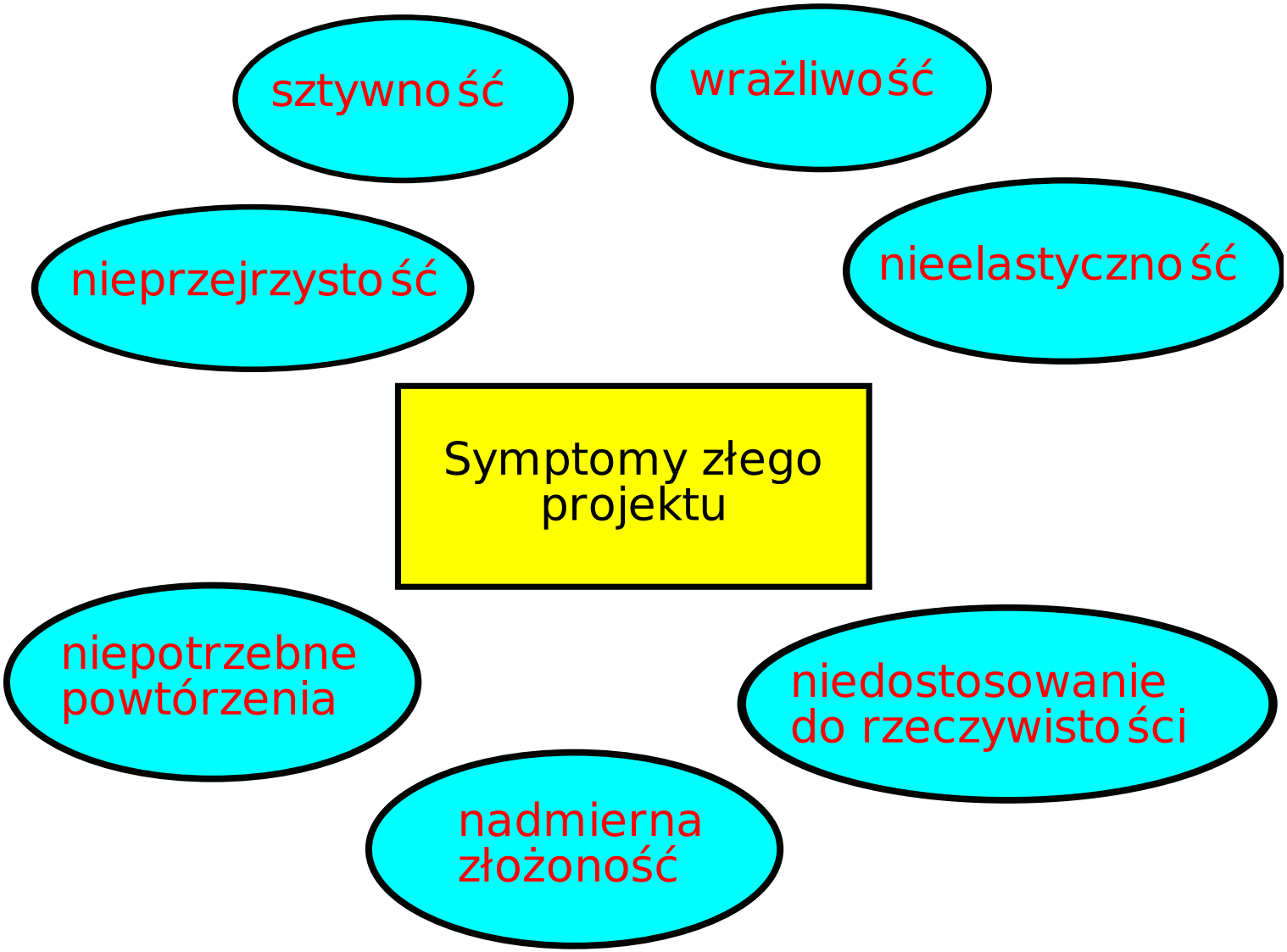
Niedostosowanie środowiska do rzeczywistości

Występuje w przypadku gdy środowisko jest wolne, tj. każda kompilacja wymaga wielu godzin, programiści są kuszeni by zaniechać stosowania częstej kompilacji kodu i testowania.

Nadmierna złożoność

Projekt zawiera elementy, które są zbędne. Powstaje przy próbie przedwczesnego przewidywania przyszłych zmian i ich implementacji.

Przypomnienie



Niepotrzebne powtórzenia

Powstaje przy programowaniu `ctr-c ctrl-v`.

Nieprzejrzystość

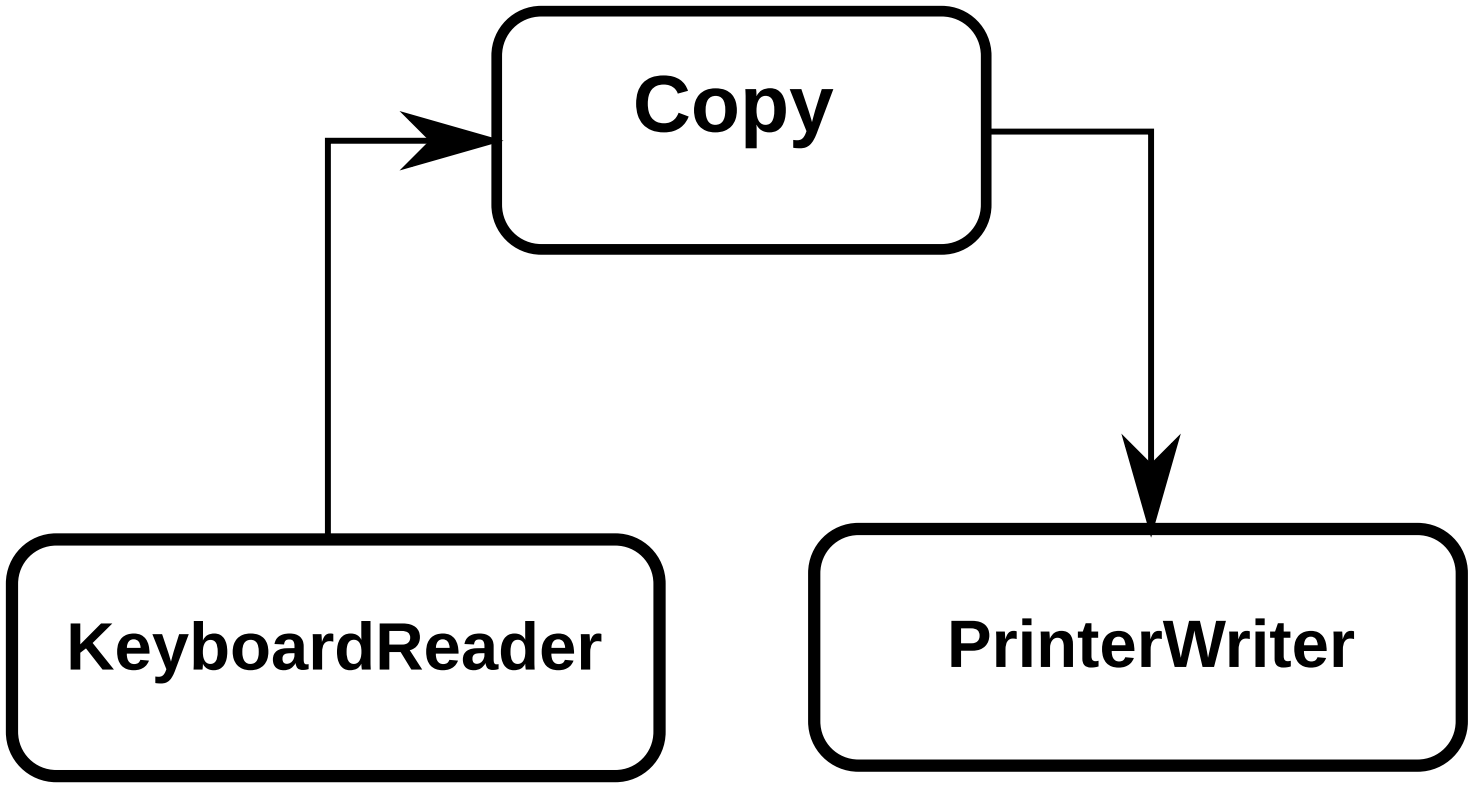
Kod jest nieczytelny, jeśli autor kodu po miesiącu lub inny programista mają kłopoty z jego zrozumieniem.

Plan wykładu

Zadanie

Napisać program, który kopiuje znaki z klawiatury na drukarkę

Projekt



Rozwiązanie

```
public class Copier
{
  public static void Copy()
  {
    int c;
    while ((c = Keyboard.Read()) != -1)
      Printer.Write(c);
  }
}
```

Zmiana wymagań

Dodanie do wymagań możliwości odczytu z czytnika taśmy papierowej.

Uzupełnienie kodu

```
public class Copier
{
  public static bool ptFlag = false;
  public static void Copy()
  {
    int c;
    while ((c = ptFlag ? PaperTape.Read()
                       : Keyboard.Read()) != -1)
      Printer.Write(c);
  }
}
```

Analiza

Czy programy korzystające z tego kawałka kodu dalej działają poprawnie?

Analiza

Czy programy korzystające z tego kawałka kodu dalej działają poprawnie?

```
...
Copier.Copy();
...
```

Analiza

Czy programy korzystające z tego kawałka kodu dalej działają poprawnie?

```
...
Copier.ptFlag = true;
Copier.Copy();
...
```

```
...
Copier.Copy();
...
```

Kolejna zmiana

Dodanie możliwości zapisywania na taśmie perforowanej.

Uzupełnienie kodu

```
public class Copier
{
  public static bool ptFlag = false;
  public static bool punchFlag = false;
  public static void Copy()
  {
    int c;
    while ((c = ptFlag ? PaperTape.Read()
                       : Keyboard.Read()) != -1)
      punchFlag ? PaperTape.Write(c) : Printer.Write(c);
  }
}
```

Ciąg dalszy historyjki

Dodajmy coś jeszcze...

Dodanie nowego czytnika: inne podejście

```
public interface Reader
{
  int Read();
}

public class KeyboardReader : Reader
{
  public int Read() { return Keyboard.Read(); }
}
```

Wykorzystanie

```
public class Copier
{
  public static Reader reader = new KeyboardReader();
  public static void Copy()
  {
    int c;
    while ((c = (reader.Read())) != -1)
      Printer.Write(c);
  }
}
```

Do rozważenia

Czy jeśli mamy już lepszą obsługę urządzeń wejściowych to czy warto do razu zrobić to samo dla urządzeń wyjściowych?

Podsumowanie

Zmiany były, są i będą; trzeba sobie z nimi radzić.

Podsumowanie

Zmiany były, są i będą; trzeba sobie z nimi radzić.

Wskazówka

Zespół musi więc:

- projektować tak oprogramowanie aby było one jak najbardziej odporne na zmiany (następny wykład);
- zarządzanie zmianą[a] jako część zarządzania projektem.

[a] czasem też używa się określenia *zarządzanie konfiguracją*

Źródła zmian

Zewnętrzne źródła zmian

- zmiany wymagań (prawo, zmiany potrzeb klienta);
- wersje specjalne dla poszczególnych klientów;
- wersje na różne platformy (wersje SO, systemy mobilne/stacjonarne).

Źródła zmian

Zewnętrzne źródła zmian

- zmiany wymagań (prawo, zmiany potrzeb klienta);
- wersje specjalne dla poszczególnych klientów;
- wersje na różne platformy (wersje SO, systemy mobilne/stacjonarne).

Wewnętrzne źródła zmian

- zmiana architektury;
- zmiana środowiska (np. silnika bazy danych);

Zwinne podejście do zmian

Z manifestu programowania zwinnego

Bądź otwarty na zmieniające się wymagania nawet na zaawansowanym etapie projektu. Zwinne procesy wykorzystują zmiany dla uzyskania przewagi konkurencyjnej Klienta.

Zarządzanie zmianą

- ustalenie zmian (zatwierdzenie i oszacowanie);
- odnotowanie zmiany w specyfikacji;
- poinformowanie zespołu o zmianach (zmiany projektu, zmiany implementacji, zmiany testów, zmiany podręcznika);
- zmiana harmonogramów prac.