Piotr Dobosz

Podstawy inżynierii oprogramowania

Tematyka zajęć

- Modele życia oprogramowania i poszczególne fazy wdrażania oprogramowania
- Zarządzanie przedsięwzięciem programistycznym
- Język UML
- Diagramy ERD
- Oprogramowanie wspomagające
- Tworzenie dokumentacji projektowej

Zaliczenie

- Zadanie praktyczne
- Test zaliczeniowy
- Odpowiedź ustna

Czym jest paradygmat?

Paradygmat (gr. παράδειγμα parádeigma „przykład, wzór”) - zbiór pojęć i teorii tworzące podastawy wybranej dziedziny nauki.

Paradygmat w programowaniu

- Definiuje przepływ sterowania w tworzonym programie
- Progamowanie funkcyjne – wykonanie, nie sposób
- Programowanie obiektowe - przepływ informacji pomiędzy obiektami (ich współpraca)
- Wskazują praktyki dozwolone i zakazane

Rodzaje paradygmatów

- Proceduralny (podział na części, z zaleceniem nie korzystania ze zmiennych globalnyc)
- Strukturalny (jeden punkt wejścia oraz wiele punktów wyjścia)
- Funkcyjny (funkcje mają nadrzędną wartość; zachowują się jak funkcje matematyczne)

Rodzaje paradygmatów

- Imperatywny (sekwencja rozkazów do wykonania)
- Obiektowy (stan i zachowanie obiektów, komunikacja pomiędzy nimi)
- Uogólniony (programowanie bez znajomości typów danych)
- Zdarzeniowy (bez sterowania przepływem, odpowiedzi na zdarzenia wywoływane przez czynniki zewnętrzne i wewnętrzne)

Rodzaje paradygmatów

- Logiczny (program przyjmuje zestaw zależności, który jest udowadniany w toku obliczeń)
- Aspektowy (separacja zagadnień)
- Deklaratywny (warunki, które musi spełniać rozwiązanie)
- Agentowy
- Modularny

Przykład programu proceduralnego

```
1  def im_Suspended():
2    print("The program loop runs from the
3    top down to the position I came from above, then
4    references me and suspends me
5    until I am later called by reference to performs
6    my suspended actions when called.")
7  a=5
8  b=7
9  im_Suspended()
10 c=8
11 d=10
12 im_Suspended() # Again REFERENCED
```

Źródło: https://www.quora.com/What-is-the-example-of-procedural-programming

Przykład programu strukturalnego

```
 INPUT "How many numbers to sort? "; T
 DIM n(T)
 FOR i = 1 TO T
   PRINT "NUMBER:"; i
   INPUT n(i)
 NEXT i
 'Calculations:
 C = T
E180:
 C = INT(C / 2)
 IF C = 0 THEN GOTO C330
 D = T - C
 E = 1
I220:
 f = E
F230:
 g = f + C
 IF n(f) > n(g) THEN SWAP n(f), n(g)
 f = f - C
 IF f > 0 THEN GOTO F230
 E = E + 1
 IF E > D THEN GOTO E180
GOTO I220
C330:
 PRINT "The sorted list is"
 FOR i = 1 TO T
   PRINT n(i)
 NEXT i
```

Źródło: https://en.wikipedia.org/wiki/Spaghetti_code

Przykład programowania funkcyjnego

```c
#include <stdio.h>

int addNum(int a, int b);      // function prototype

int main() {
   int sum;
   sum = addNum(5,6);          // function call
   printf("sum = %d",sum);
   return 0;
}
int addNum (int a,int b) {     // function definition
   int result;
   result = a + b;
   return result;              // return statement
}
```

Źródło: https://www.tutorialspoint.com/functional_programming/functional_programming_functions_overview.htm

Przykład programowania imperatywnego

```
1  ...
2  while (true) do
3    if((object_1.attribute_1 = 1) and
4       (object_2.attribute_1 = 1) and
5       (object_3.attribute_1 = 1))
6    then
7        object_1.method_1();
8        object_2.method_1();
9        object_3.method_1();
10   end_if
11   ...
12   if((object_1.attribute_1 = 1) and
13      (object_2.attribute_n = n) and
14      (object_3.attribute_n = n))
15   then
16       object_1.method_n();
17       object_2.method_n();
18       object_3.method_n();
19   end_if
20 end_while
21 ...
```

Źródło: https://www.researchgate.net/figure/Example-of-imperative-code-1_fig1_266013049

Przykład programowania obiektowego

```
class Dog:
    species = "Canis familiaris"

    def __init__(self, name, age):
        self.name = name
        self.age = age

    def __str__(self):
        return f"{self.name} is {self.age} years old"

    def speak(self, sound):
        return f"{self.name} says {sound}"
```

Źródło: https://realpython.com/python3-object-oriented-programming/

Przykład programowania uogólnionego

```cpp
#include <iostream>
using namespace std;

// One function works for all data types.
// This would work even for user defined types
// if operator '>' is overloaded
template <typename T>

T myMax(T x, T y)
{
    return (x > y) ? x : y;
}

int main()
{

    // Call myMax for int
    cout << myMax<int>(3, 7) << endl;

    // call myMax for double
    cout << myMax<double>(3.0, 7.0) << endl;

    // call myMax for char
    cout << myMax<char>('g', 'e') << endl;

    return 0;
}
```

Output:

```
7
7
g
```

Źródło: https://www.geeksforgeeks.org/generics-in-c/

Przykład programowania zdarzeniowego

```
do forever: // the main scheduler loop

    get event from input stream

    if event type == EndProgram:
        quit // break out of event loop

    else if event type == event_01:
        call event-handler for event_01 with event parameters

    else if event type == event_02:
        call event-handler for event_02 with event parameters

    .
    .
    .

    else if event type == event_nn:
        call event-handler for event_nn with event parameters

    else handle unrecognized event // ignore or raise exception

end loop
```

Źródło: http://www.technologyuk.net/computing/software-development/software-design/event-driven-programming.shtml

Przykład programowania logicznego

```haskell
import Control.Monad (guard)


data Sex = Male | Female deriving (Eq, Show)

data PuzzleAnswer = PuzzleAnswer {
    parent1 :: Sex,
    parent2 :: Sex,
    child :: Sex,
    child_desc :: Sex
}

instance Show (PuzzleAnswer) where
    show pa = "Parent1 is " ++ (show $ parent1 pa) ++ "\n" ++
              "Parent2 is " ++ (show $ parent2 pa) ++ "\n" ++
              "The child is " ++ (show $ child pa) ++ "\n" ++
              "The child said they were " ++ (show $ child_desc pa) ++ "\n"

childs_statement_is_valid :: Sex -> Sex -> Bool
childs_statement_is_valid Male Female = False
childs_statement_is_valid _ _ = True

parent1_statement_is_valid :: Sex -> Sex -> Bool
parent1_statement_is_valid Male Female = False
parent1_statement_is_valid _ _ = True

parent2_statement_is_valid :: Sex -> Sex -> Sex -> Bool
parent2_statement_is_valid Male Female Male = True
parent2_statement_is_valid Female _ Female = True
parent2_statement_is_valid _ _ _ = False

solve_puzzle :: (Sex -> Sex -> Bool) -> [PuzzleAnswer]
solve_puzzle sexuality_pred = do
    parent1 <- [Male, Female]
    parent2 <- [Male, Female]
    child <- [Male, Female]
    child_desc <- [Male, Female]
    guard $ sexuality_pred parent1 parent2
    guard $ childs_statement_is_valid child child_desc
    guard $ parent1_statement_is_valid parent1 child_desc
    guard $ parent2_statement_is_valid parent2 child child_desc
    return $ PuzzleAnswer {
        parent1=parent1,
        parent2=parent2,
        child=child,
        child_desc=child_desc
    }

main = do
    putStrLn "----------- Heterosexual Couple -----------"
    mapM_ print (solve_puzzle (/=))
    putStrLn "----------- Gay Couple -----------"
    mapM_ print (solve_puzzle (\x y -> x == y && x == Male))
    putStrLn "----------- Lesbian Couple -----------"
    mapM_ print (solve_puzzle (\x y -> x == y && x == Female))
```

Źródło: https://wiki.haskell.org/Logic_programming_example

Przykład programowania aspektowego

```
void transfer(Account fromAcc, Account toAcc, int amount) throws Exception {
  if (fromAcc.getBalance() < amount)
      throw new InsufficientFundsException();

  fromAcc.withdraw(amount);
  toAcc.deposit(amount);
}
```

```
void transfer(Account fromAcc, Account toAcc, int amount, User user,
    Logger logger, Database database) throws Exception {
  logger.info("Transferring money...");

  if (!isUserAuthorised(user, fromAcc)) {
    logger.info("User has no permission.");
    throw new UnauthorisedUserException();
  }

  if (fromAcc.getBalance() < amount) {
    logger.info("Insufficient funds.");
    throw new InsufficientFundsException();
  }

  fromAcc.withdraw(amount);
  toAcc.deposit(amount);

  database.commitChanges();  // Atomic operation.

  logger.info("Transaction successful.");
}
```

Źródło: https://en.wikipedia.org/wiki/Aspect-oriented_programming

Przykład programowania deklaratywnego

```
1 using System;
2 using System.Linq;
3
4 class Example
5 {
6     static void Main()
7     {
8         var sum = Enumerable.Range(0, 99)
9                         .Where(isEven)
10                        .Sum();
11        Console.WriteLine(sum);
12    }
13
14    static bool isEven(int number)
15    {
16        return number % 2 == 0;
17    }
18 }
```

Źródło: https://codewithshadman.com/declarative-programming/

Przykład programowania agentowego

```java
package helloworld;
import jade.core.Agent;

public class Hello extends Agent {

    protected void setup() {
        System.out.println("Hello World. ");
        System.out.println("My name is "+ getLocalName());
    }

    public Hello() {
        System.out.println("Constructor called");
    }

}
```

```
  started.

  +started <- .print("Hello World. ").
```

Źródło: https://en.wikipedia.org/wiki/Agent-oriented_programming

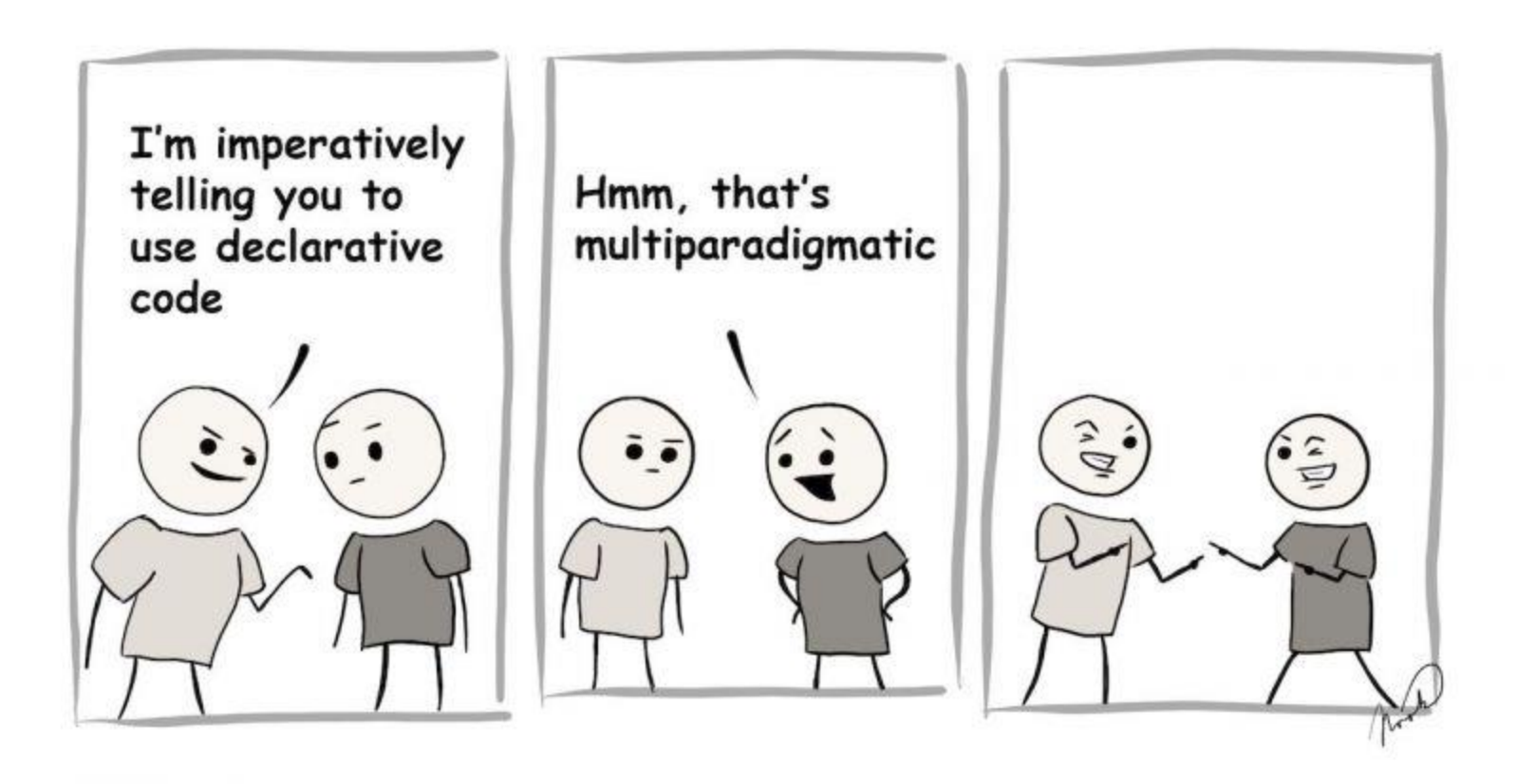
Przykład programowania modularnego

```
stack.h:
        extern stack_var1;
        extern int stack_do_something(void);

stack.c
#include
int stack_var1;
static int stack_var2;

int stack_do_something(void)
{
  stack_var1 = 2;
  stack_var2 = 5;
}
```

```
#include
int main(int argc, char*argv[]){
while(1){
  stack_do_something();
    }
}
```

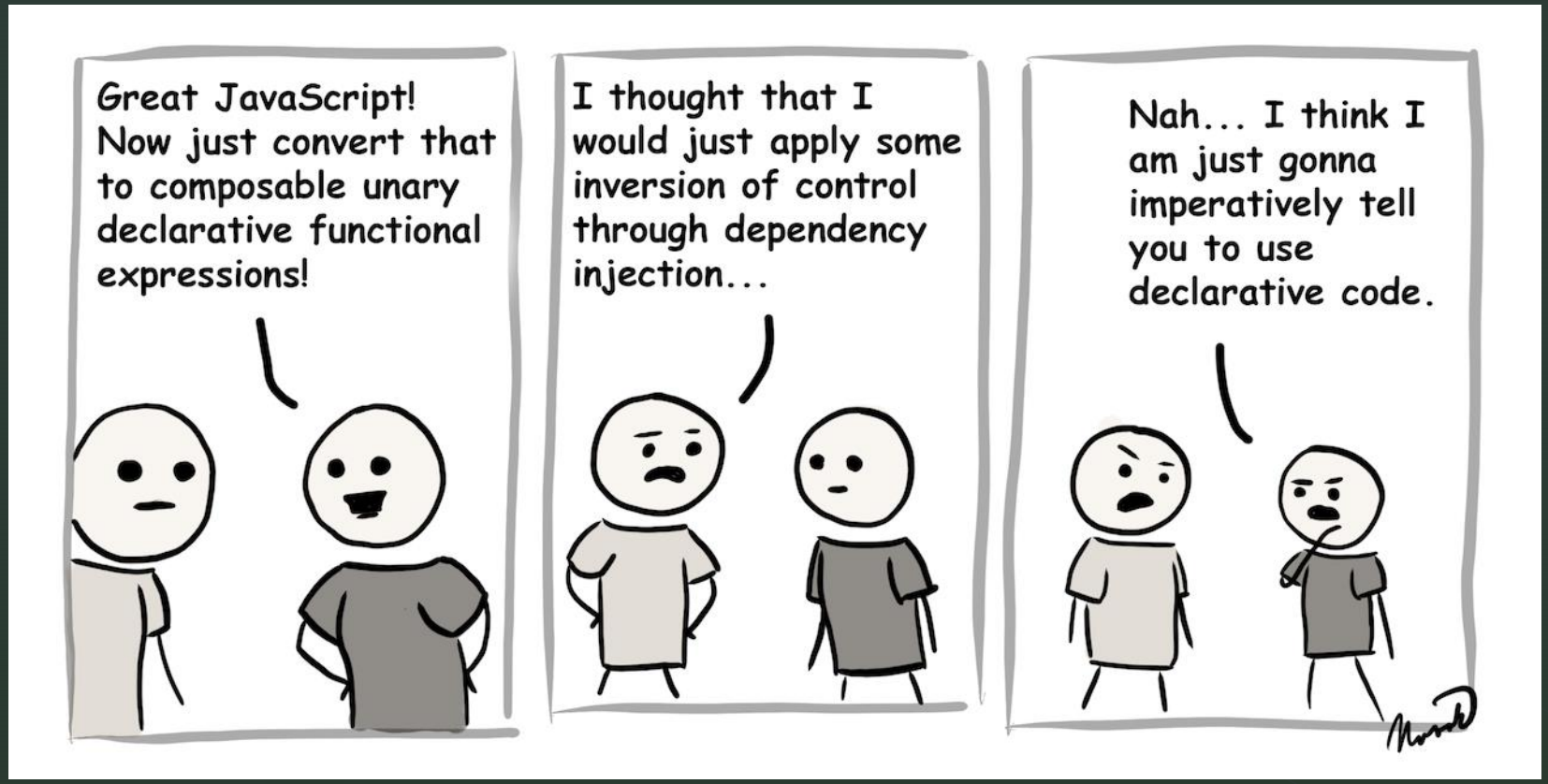
Źródło: https://www.geeksforgeeks.org/modular-approach-in-programming/

Którego paradygmatu używać?



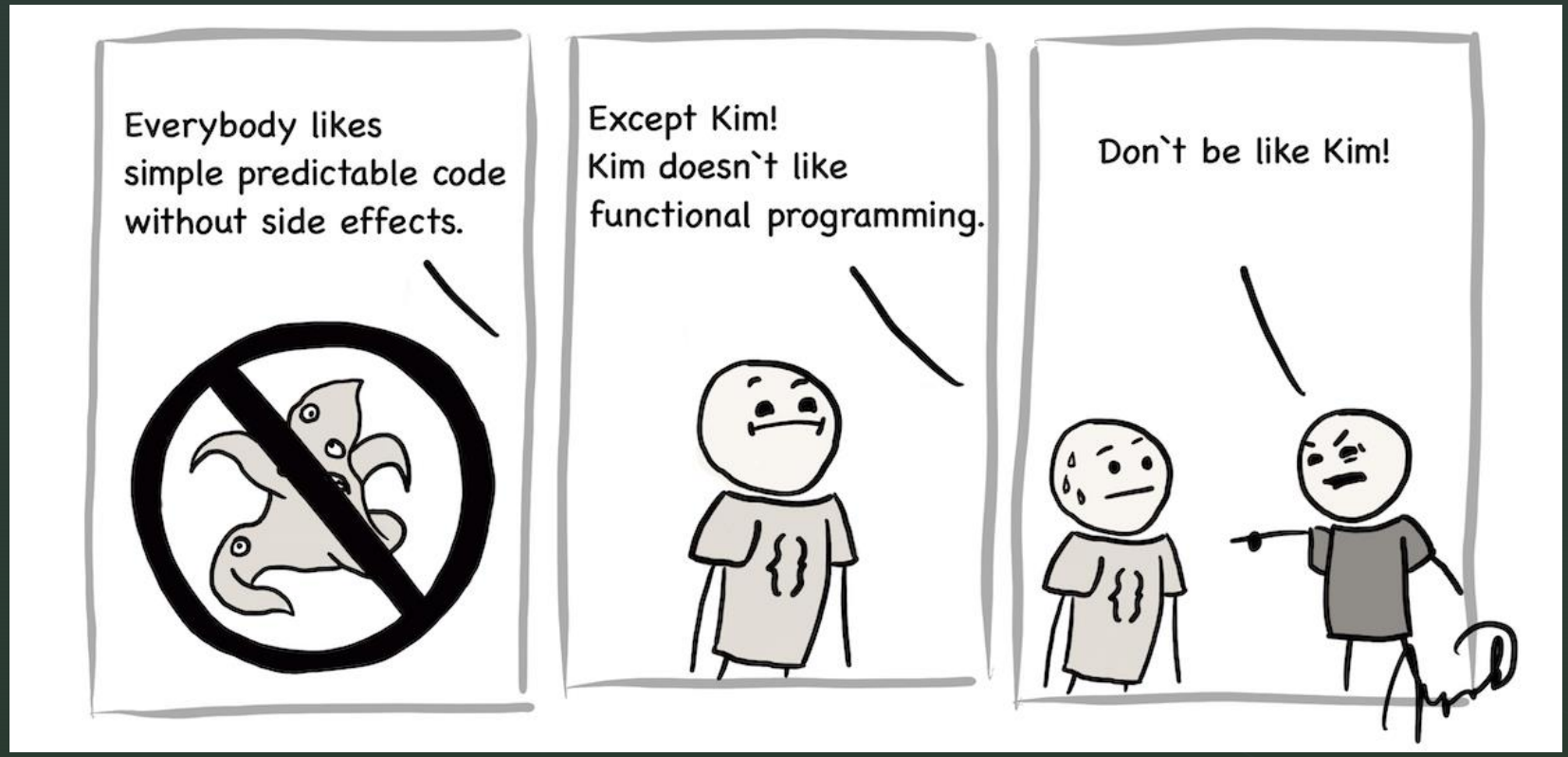
Źródło: https://img.devrant.com/devrant/rant/r_1541501_oepL5.jpg

Którego paradygmatu używać?



Źródło: https://miro.medium.com/max/2560/1*V-RqwITiRRdXCVJWD1xXIw.png

Którego paradygmatu używać?

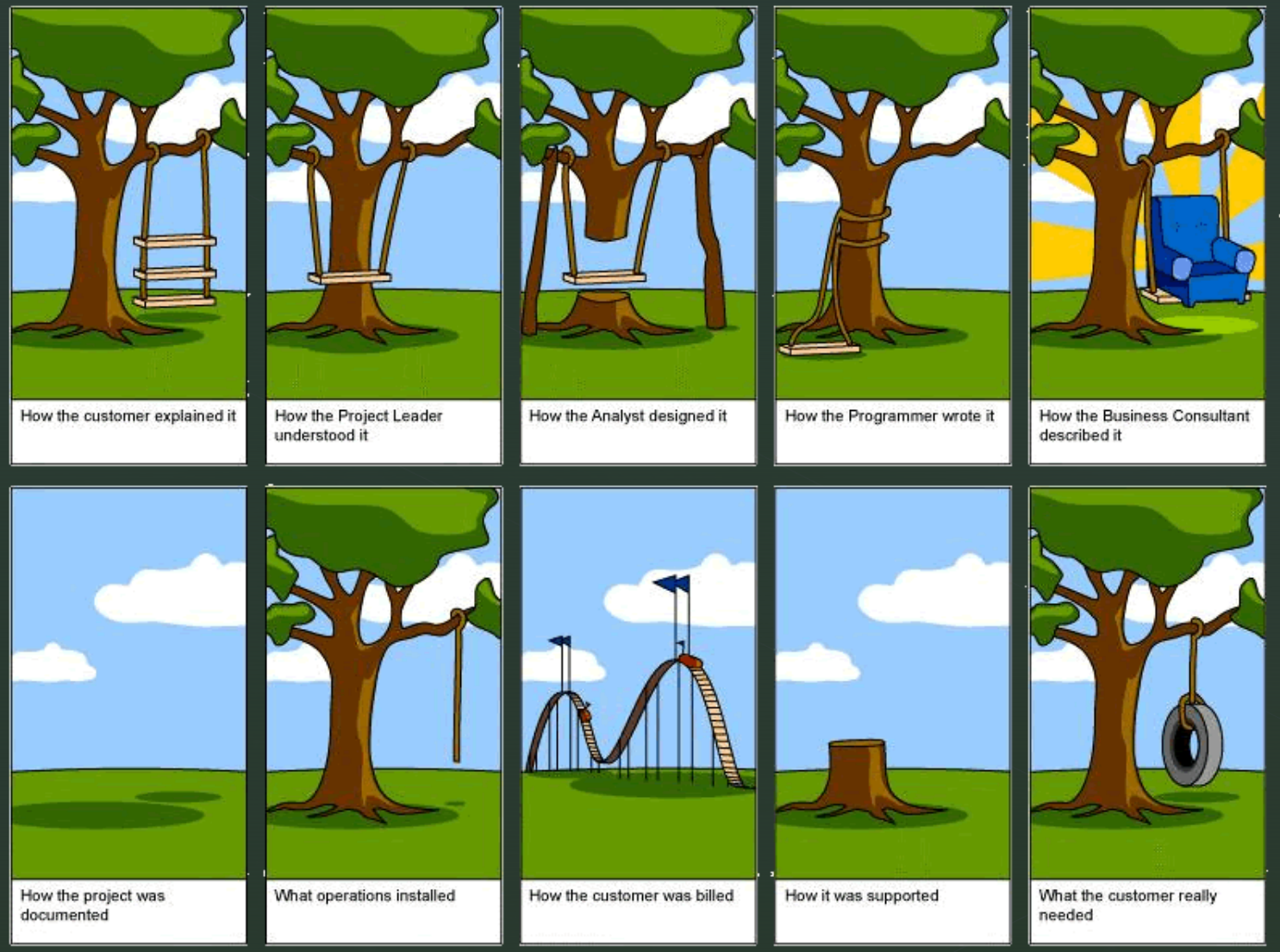


Źródło: https://miro.medium.com/max/1200/1*Uqm22oIfuUKYuG4qovigNw.png

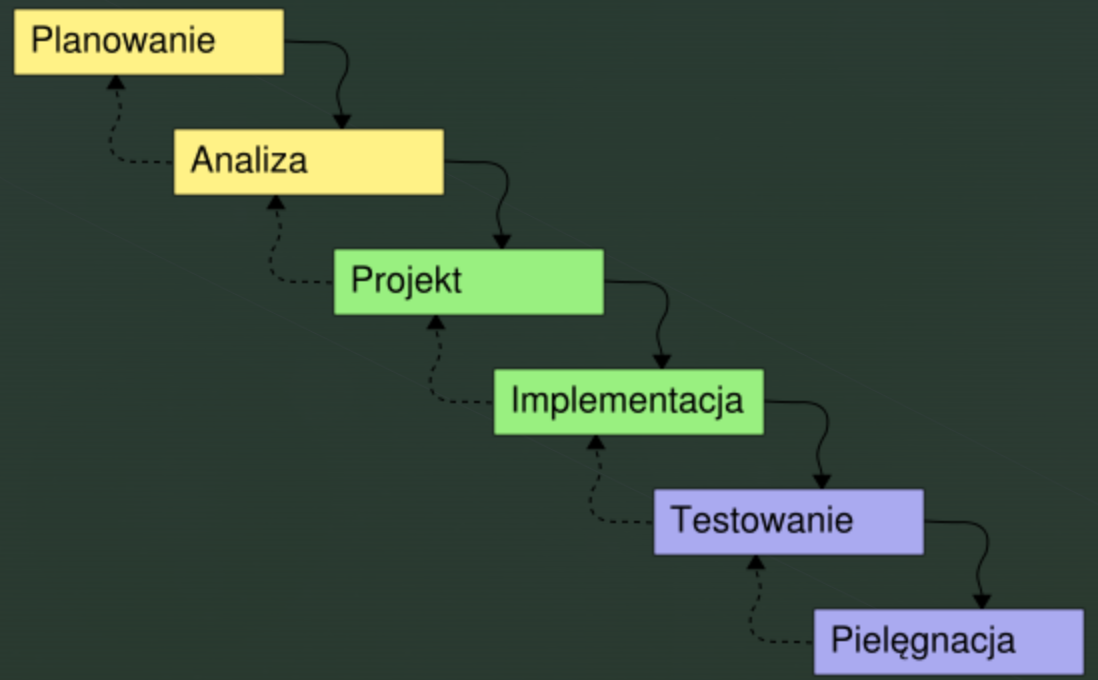
Cykl życia oprogramowania

- Określanie wymagań i specyfikacji
- Projektowanie
- Implementacja
- Testowanie – walidacja (atestowanie) i weryfikacja
- Konserwacja (pielęgnacja)

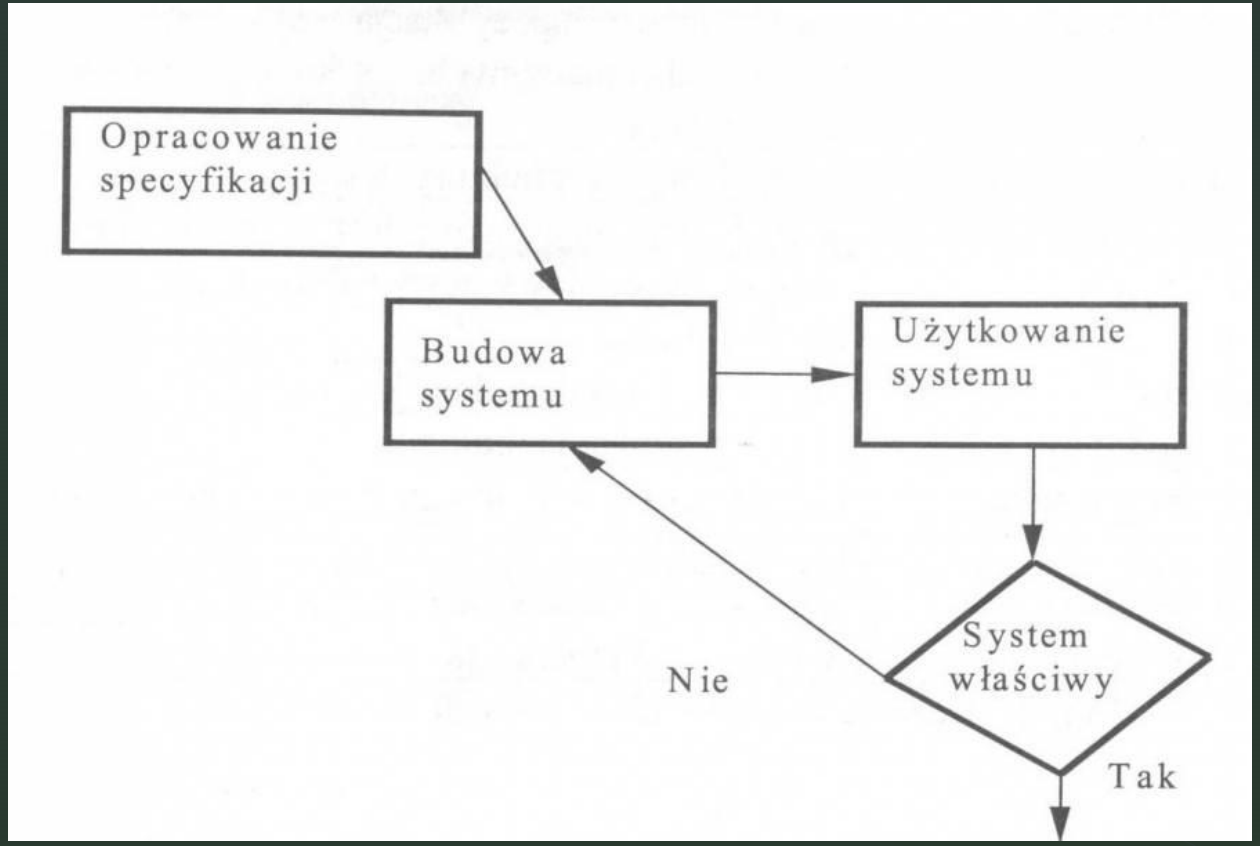
Cykl życia oprogramowania



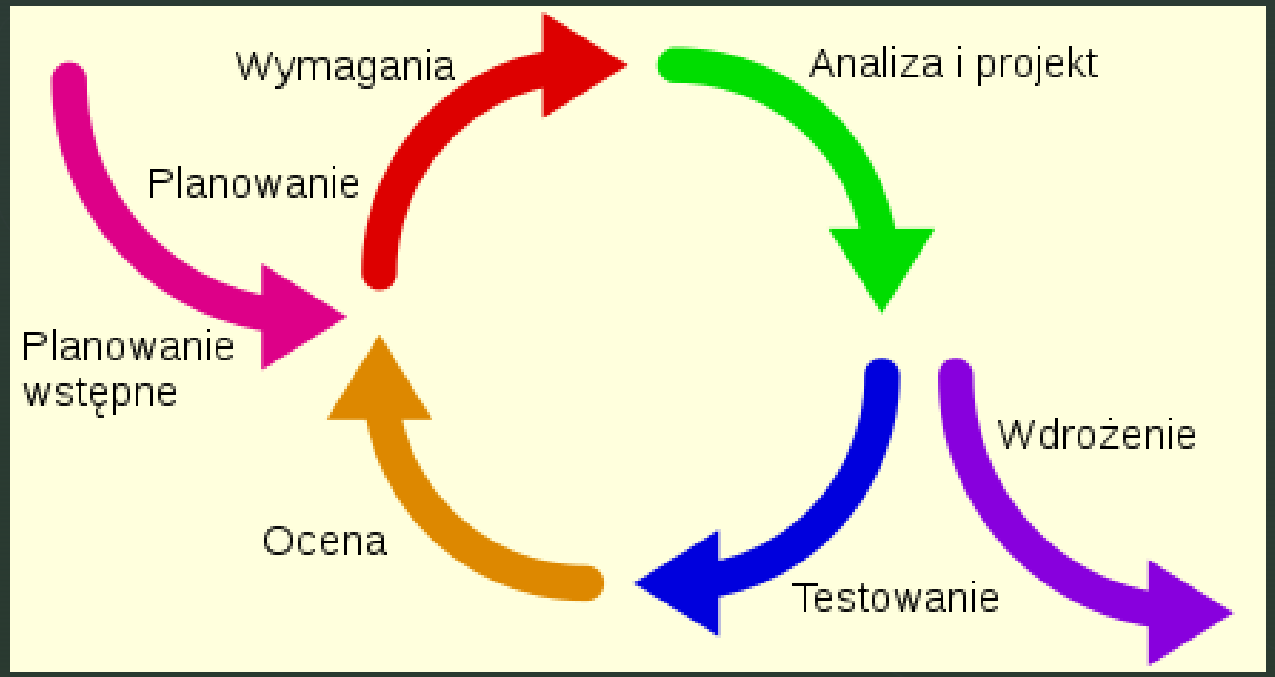
Model kaskadowy (sekwencja)



Ewolucyjny


Opracowanie specyfikacji
Budowa systemu
Użytkowanie systemu
System właściwy
Nie
Tak

Przyrostowy



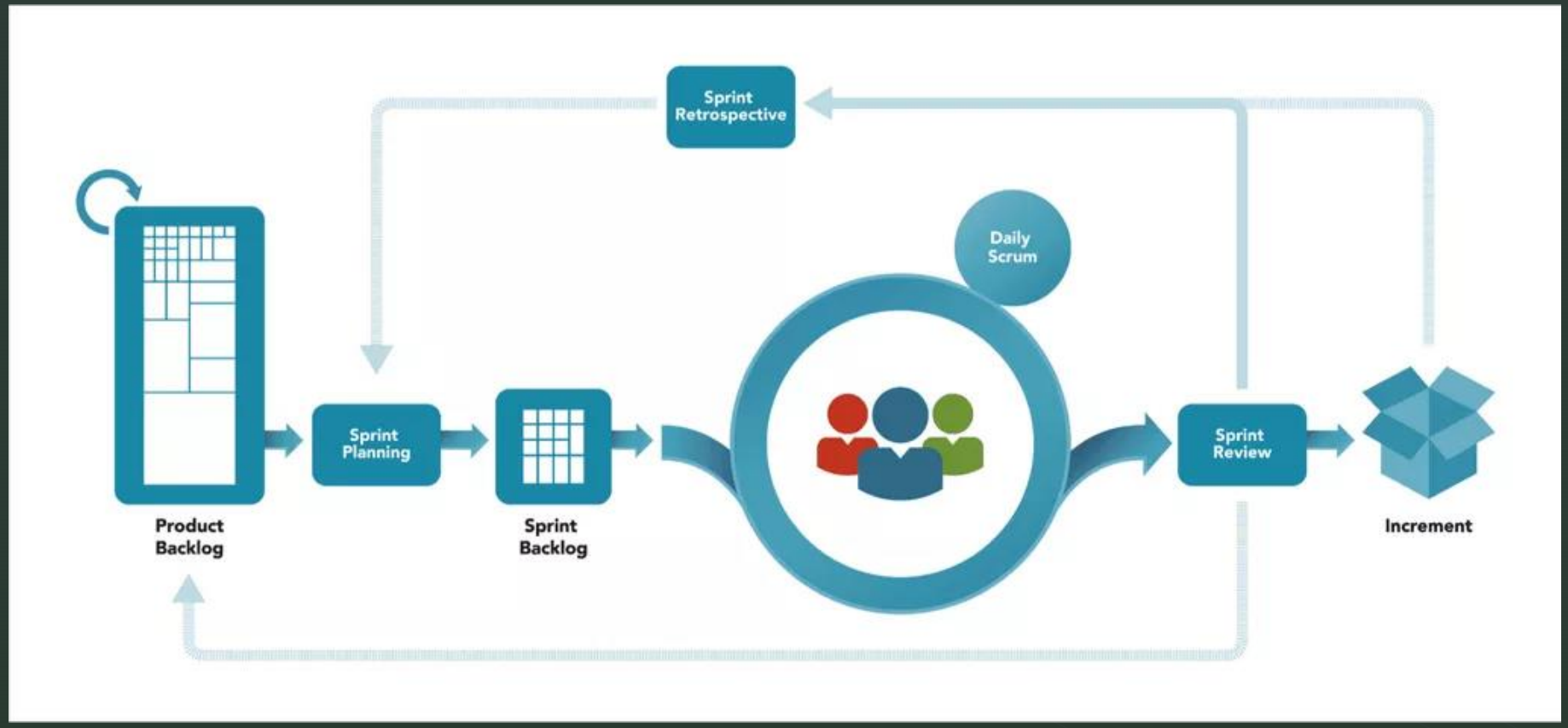
Zwinny

- plan (planowanie)
- design (projektowanie)
- develop (programowanie)
- test (testowanie)
- release (implementacja)
- feedback (informacja zwrotna)

Prototypowy

- Prototyp jest łatwy do zmiany
- Zwiększa zrozumienie programistów co do potrzeb klienta
- Pozwala klientowi zobaczyć jak mniej więcej system będzie wyglądał
- W zależności od rodzaju prototypu, może pozwalać rozpocząć szkolenie obsługi systemu po stronie klienta
- Redukcja kosztów

Scrum

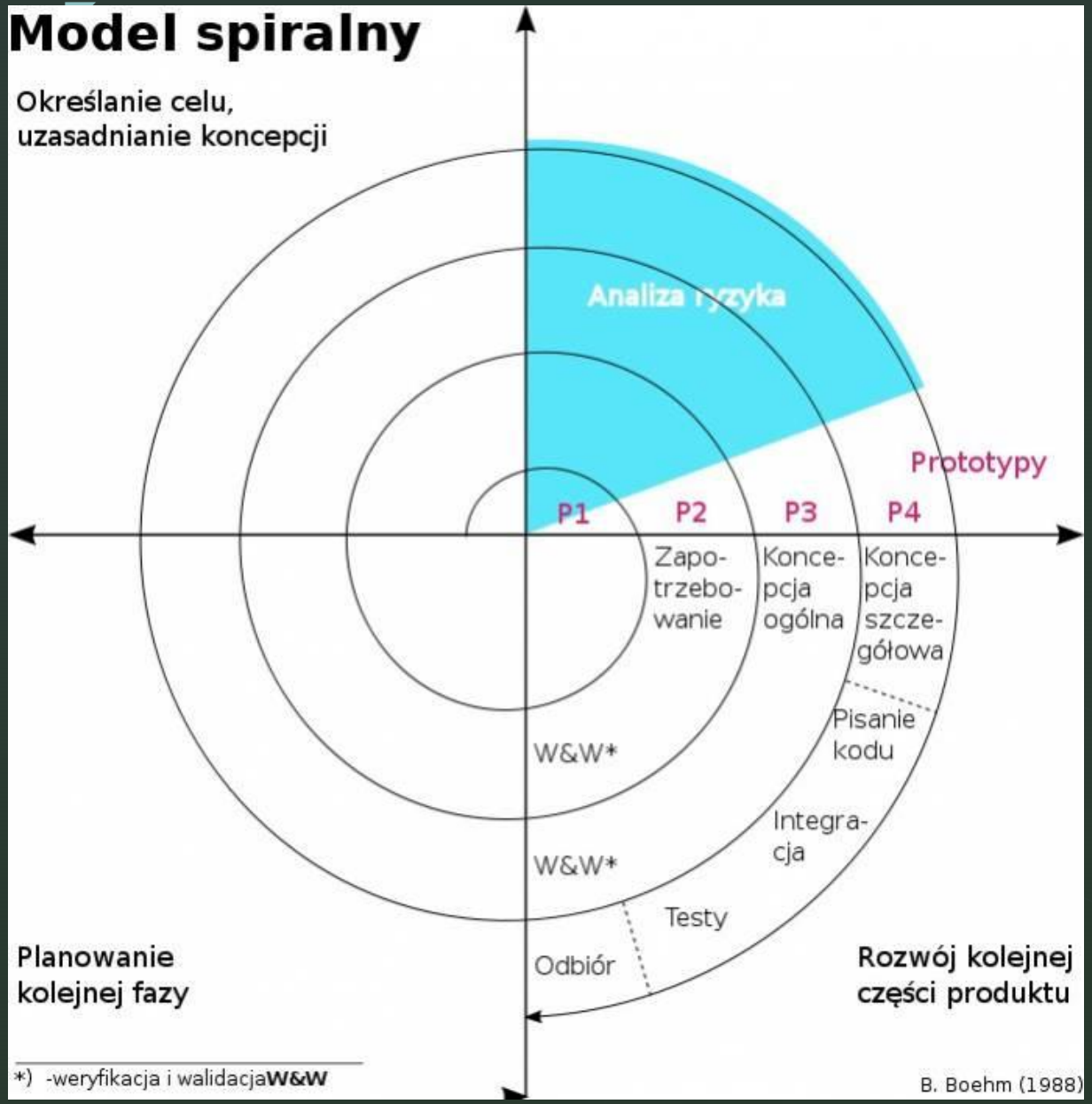


Źródło: https://i2.wp.com/www.qagile.pl/wp-content/uploads/2014/05/scrum-w-pigulce-psm-compressed.png?resize=1024%2C474&ssl=1

Ekstremalny

- Iteracyjność
- Nie projektować z góry
- Testy jednostkowe
- Ciągłe modyfikacje architektury
- Programowanie parami
- Stały kontakt z klientem

Spiralny


Model spiralny
Określanie celu,
uzasadnianie koncepcji
Analiza ryzyka
Prototypy
P1
P2
P3
P4
Zapo-
trzebo-
wanie
Konce-
pcja
ogólna
Konce-
pcja
szcze-
gółowa
Pisanie
kodu
W&W*
Integra-
cja
W&W*
Testy
Odbiór
Planowanie
kolejnej fazy
Rozwój kolejnej
części produktu
*) -weryfikacja i walidacjaW&W
B. Boehm (1988)

Komponentowy

- wymagania narzucane przez gotowe komponenty mogą być niezgodne z wymaganiami klientów
- modyfikacje kodu mogą być utrudnione przez brak kontroli nad pochodzącymi z zewnątrz komponentami
- mały koszt, głównie łączenie gotowych klocków

Materiały i odnośniki

- http://www.lomilowka.pl/upload/file/PAI/4_paradygmaty.pdf
- https://medium.com/front-end-weekly/imperative-versus-declarative-code-whats-the-difference-adc7dd6c8380
- https://www.iro.umontreal.ca/~vaucher/Agents/Jade/primer2.html
- https://press.rebus.community/programmingfundamentals/chapter/modular-programming/
- https://www.is.umk.pl/~grochu/wiki/doku.php?id=zajecia:ppz:cykl_zycia_oprogramowania
- http://zasoby.open.agh.edu.pl/~10sdczerner/page/cykl_zycia_oprogramownia.html
- https://pl.wikipedia.org/wiki/Model_przyrostowy
- https://pl.wikipedia.org/wiki/Programowanie_zwinne
- https://pl.wikipedia.org/wiki/Model_prototypowy