

# Software Craftsmanship

## Język wzorców językiem profesjonalistów

# Krótko o sobie...

## Twórca oprogramowania – całościowe podejście

- Technologie
- Architektury
- Metodyki, podejścia, najlepsze praktyki
- Usability

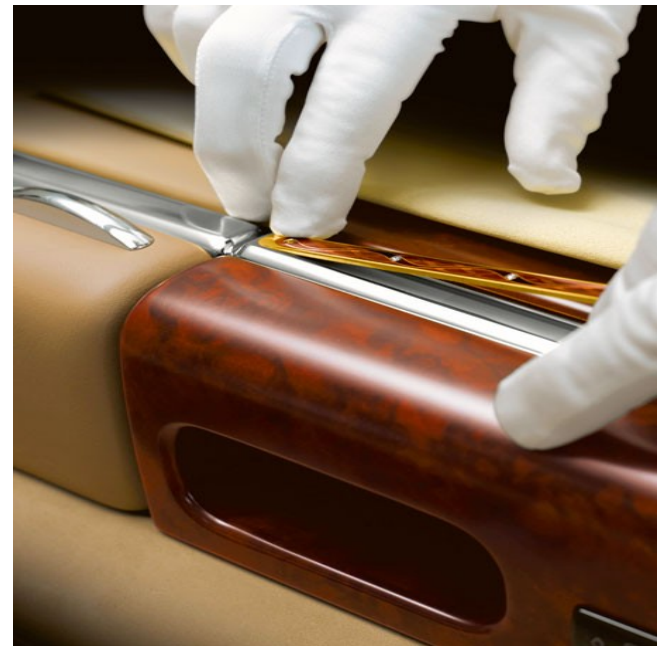
## Trener i konsultant

- Java EE
- Inżynieria oprogramowania

## Community

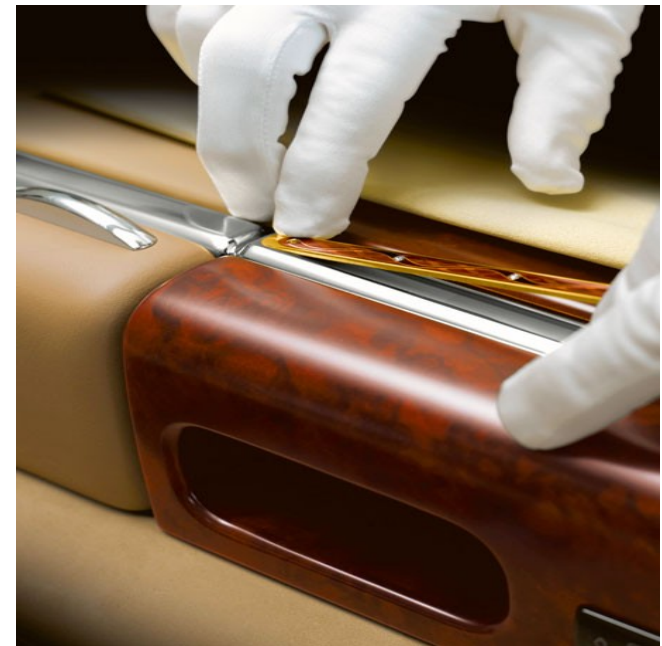
- prezes Stowarzyszenia Software Engineering Professionals Polska
- blogger, autor
- członek rady programowej 4Developers

# Craftsmanship

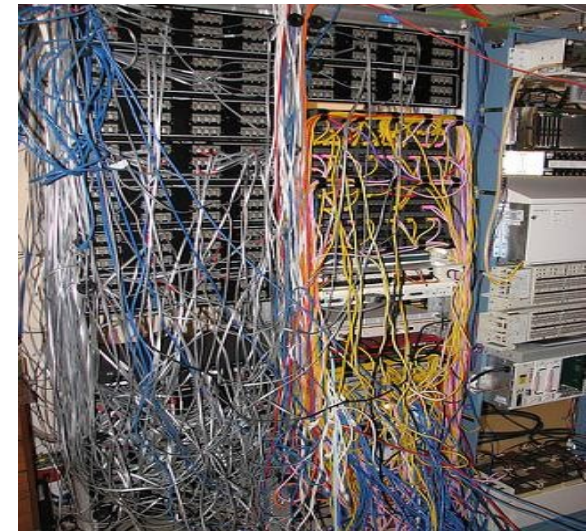
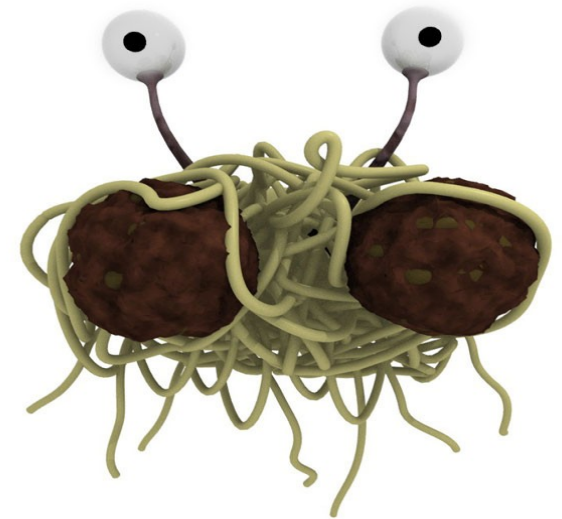
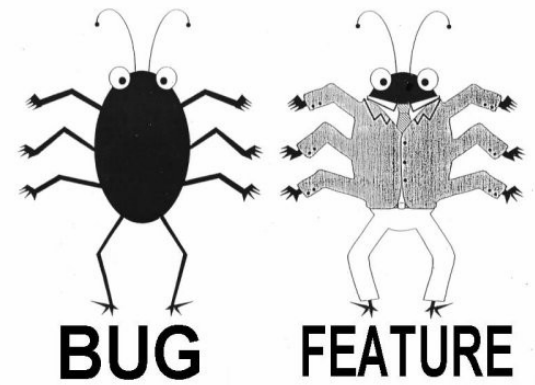


# Craftsmanship

- Profesja
  - Przygotowanie (edukacja)
  - Zasady i najlepsze praktyki
  - Wewnętrzny kodeks etyczny
  - Gwarancja wysokiego standardu
- Przykład:
  - lekarze;)
  - prawnicy;)))
  - **architekci!**



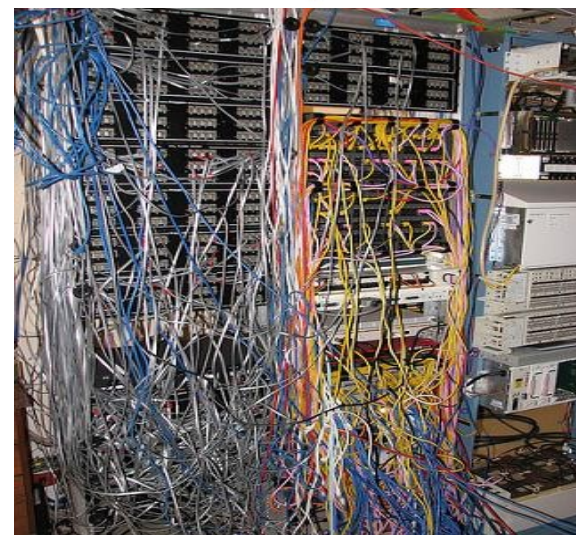
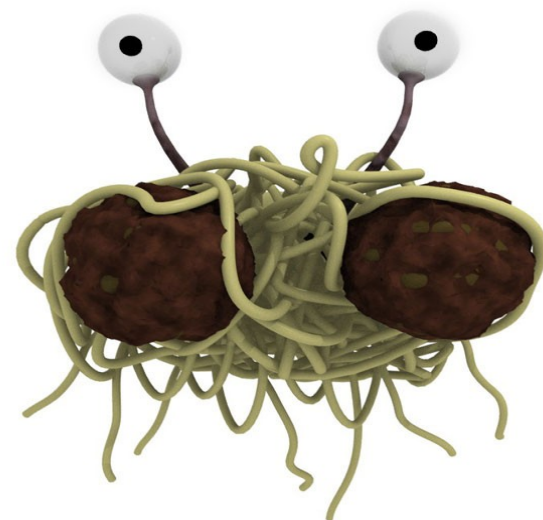
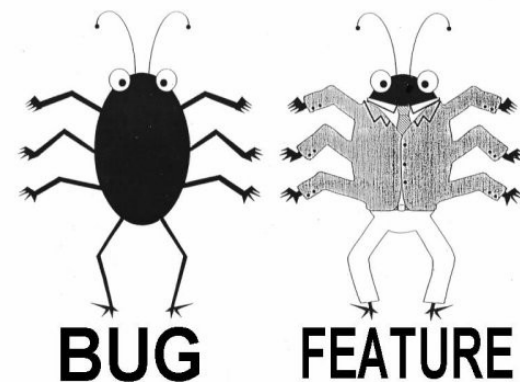
# IT – branża niedojrzała





# IT – branża niedojrzała

- Partactwo
  - **WHISKY**  
Why the Hell Isn't Somebody Koding Yet
  - **NIKE**  
Just do it!
  - **RÓB!**  
Modyfikacja RUP  
(Rational Unified Process)



**Czy wsiadłbyś do samolotu wiedząc, że...**



**...zainstalowano na nim Twój soft?**

# Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.  
Through this work we have come to value:

**Individuals and interactions** over processes and tools

**Working software** over comprehensive documentation

**Customer collaboration** over contract negotiation

**Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.



# *Manifesto for Software Craftsmanship*

Raising the bar.

As aspiring Software Craftsmen we are raising the bar of professional software development by practicing it and helping others learn the craft. Through this work we have come to value:

Not only working software,  
but also **well-crafted software**

Not only responding to change,  
but also **steadily adding value**

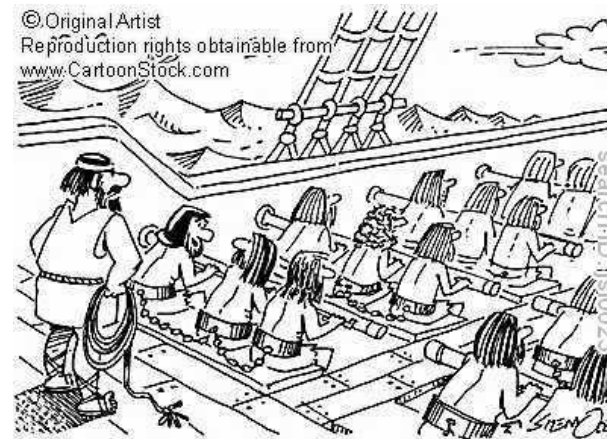
Not only individuals and interactions,  
but also **a community of professionals**

Not only customer collaboration,  
but also **productive partnerships**

That is, in pursuit of the items on the left we have found the items on the right to be indispensable.

# Craftsmanship nie jest dla mnie, ponieważ:

- Nie jest zgodny z polityką mojej firmy
- Nie jest zgodny ze strategią zarządzania mojego managera
- Nie jest w moim stylu



# Profesja w branży IT

## Nowy trend

- krytyczne spojrzenie na mainstream

## Pojęcie marki i jakości

- pożądaney przez niektórych klientów

## Naturalny proces dojrzewania

- społeczności dostawców
- części klientów



# Profesjonalista

- Nie zastawia sam na sobie pułapek
- Nie sabotuje siebie samego



# Spektrum profesjonalizmu

A landscape photograph featuring a vibrant rainbow arching across a cloudy sky. In the foreground, a dark, silhouetted tree stands on a grassy hill. A winding path leads from the bottom center towards the tree. The overall scene is bathed in a warm, golden light, suggesting either sunrise or sunset.

Usability

Wyzwania koncepcyjne

Design

Dobór narzędzia do  
problemu

Myślenie i podejście  
systemowe

Clean Code

- Uncle Bob
- potrzeby przemysłu



# Wzorce Projektowe

- Geneza w budownictwie
- Inspirowane pracami Chrisophera Alexandra



# Wzorce

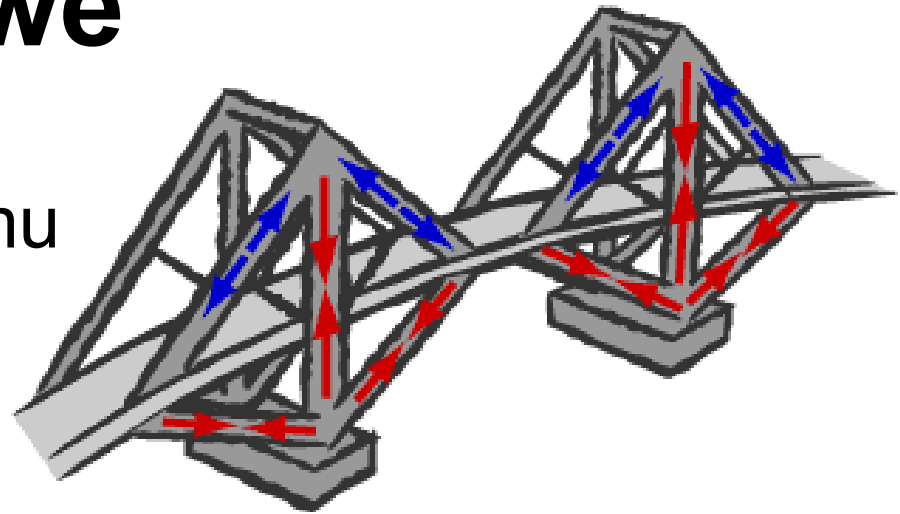
- Implementacyjne
- **Projektowe (Gang of Four)**
- Architektoniczne
  - Arch. systemu
  - Arch. aplikacji
- Integracyjne
- Specyficzne: J2EE, wielowątkowość
- Analityczne (Archetypy modeli biznesowych)





# Wzorce projektowe

- Abstrakcyjny opis problemu
  - Kontekst
  - **Siły**
- Rozwiązanie – klasy i zależności
  - Standardowe i sprawdzone
  - Motywacja i stosowalność
  - **Konsekwencje**
    - Otwarcie/zamknięcie dróg rozbudowy







„Z czasem przestajesz używać słowa 'wzorzec'  
z czasem robisz po prostu właściwą rzecz”

Jim Coplien

**Expert**

Głównie intuicja, wszystko zależy od kontekstu

**Proficient**

Szerszy kontekst, metafory, antywzorce. Pojawia się instynkt. Wiem, że nic nie wiem.

**Competent**

Nastawienie na cel, samodzielny dobór kroków. Pierwsze wzorce. Wyżej == podjęcie wysiłku poznawczego

**Advanced begginer**

Wciąż nastawienie na zadania, jednak umiejętność składania kroków. Tendencja do zawyżania.

**Novice**

Nastawienie na zadania, potrzeba okресlenia prostych kroków

**Wzorce są odkrywane a nie wymyślane**





Odżywianie	Poruszanie
------------	------------



Odżywianie	Poruszanie
------------	------------



Odżywianie	Poruszanie
------------	------------



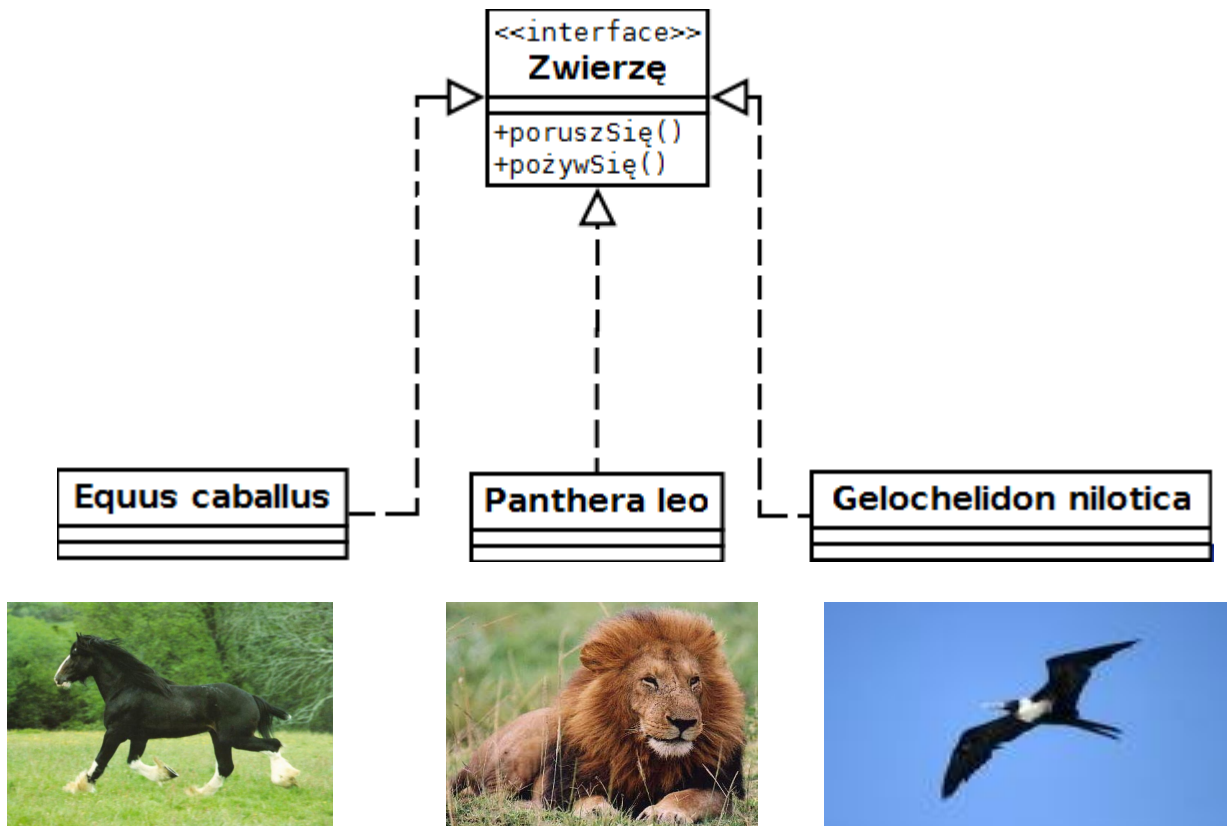
# Wymagania

- Odżywianie
  - mięsożerne
  - roślinożerne
- Poruszanie
  - biegające
  - latające
  - pływające

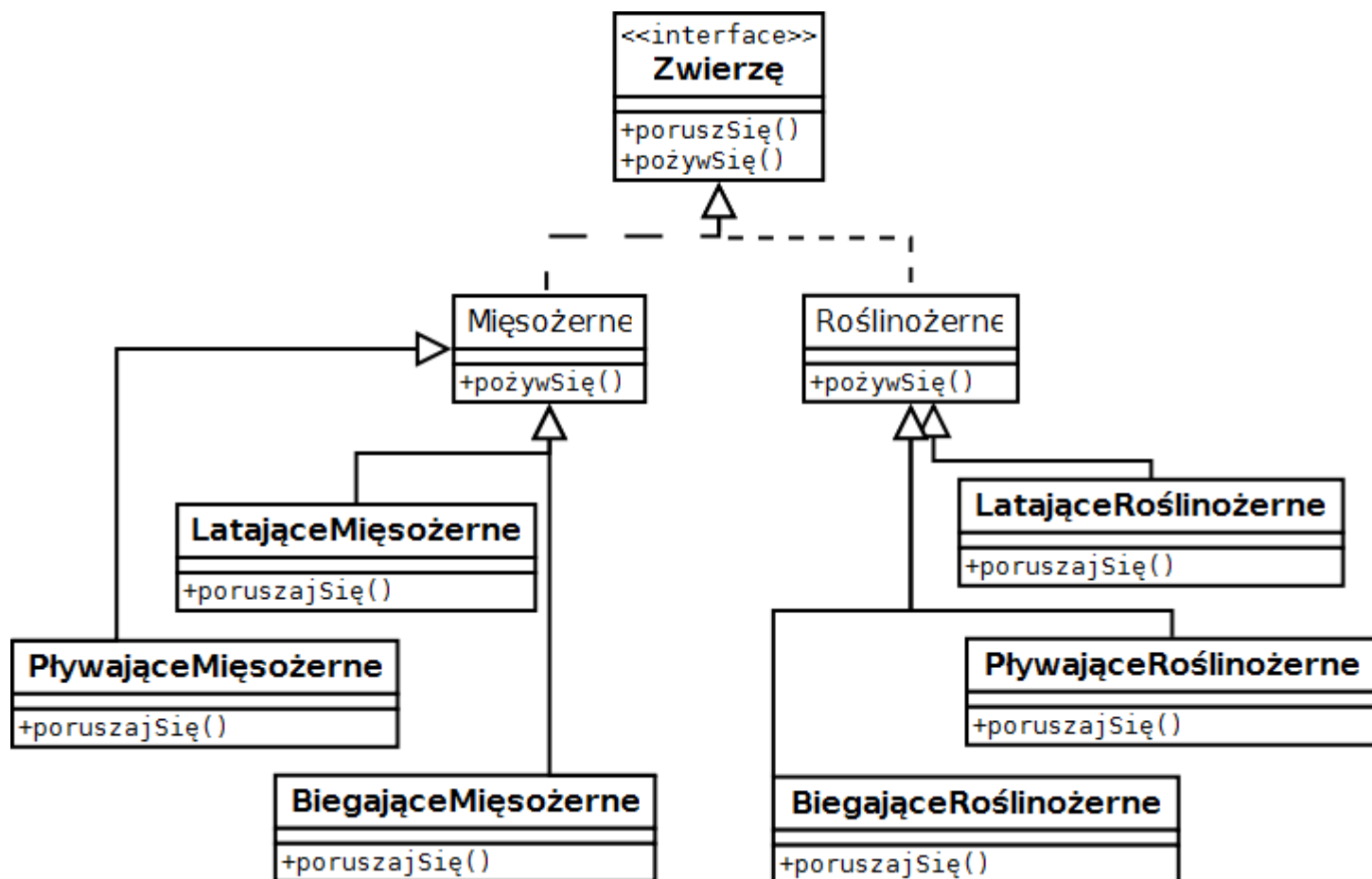
# Rozwiązanie "bo (jakoś tam) działa"

```
public class ZwierzęUtils{  
    public static void załatwSprawę(...){  
        if...  
        if...  
        ...  
    }  
}
```

# Rozwiązanie redundantne



# Rozwiązanie wybuchowe



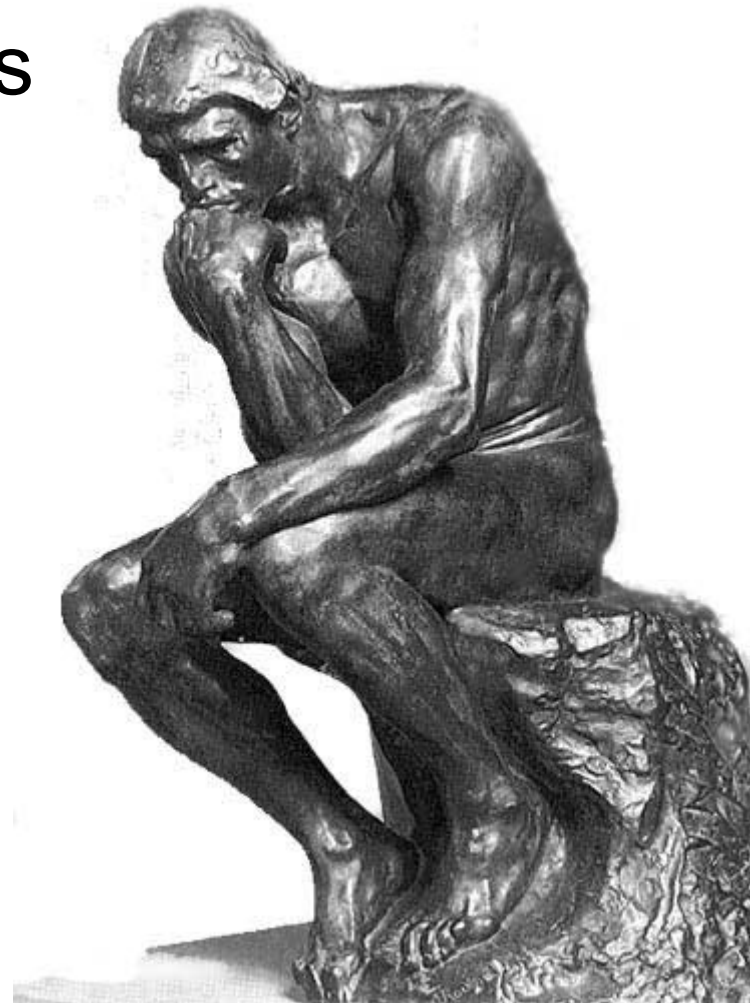


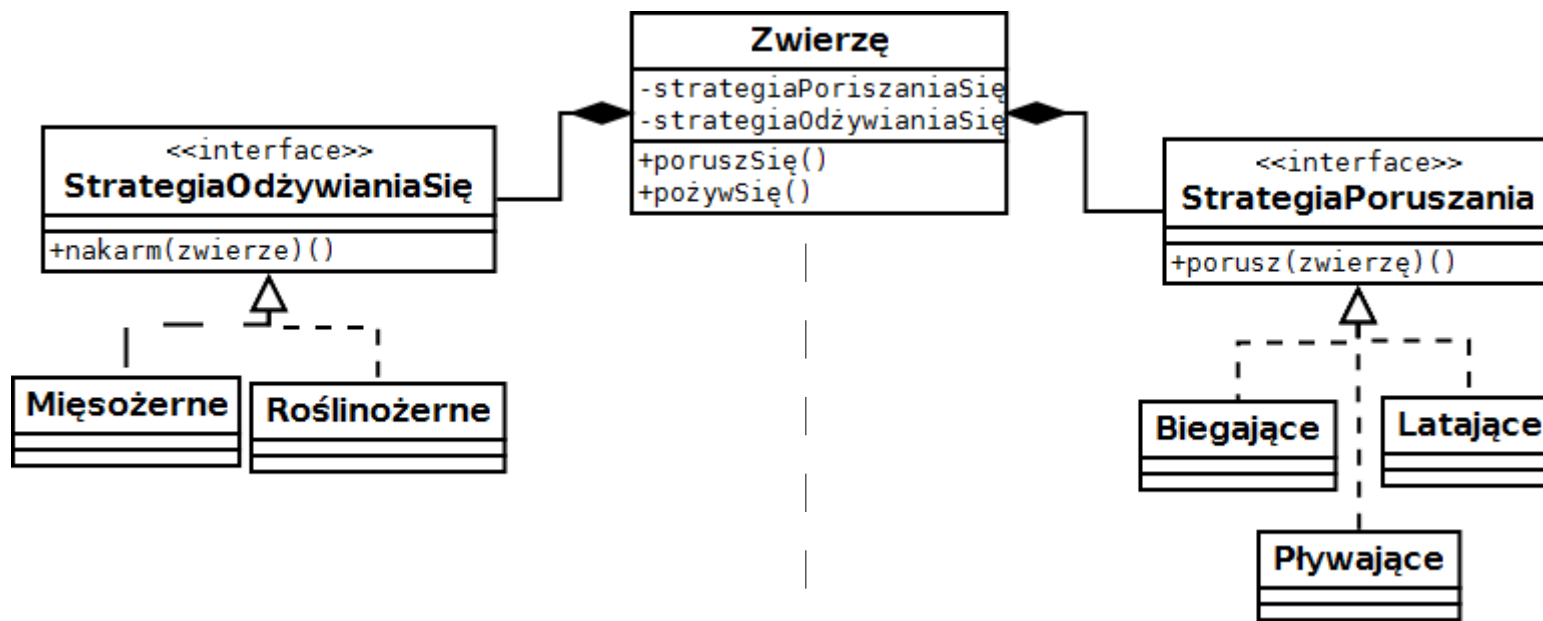
# Zmiany powodują eksplozję kombinatoryczną

- Dodanie **nowego sposobu** poruszania lub odżywiania
  - wszystkożerne
  - poruszające się lotem ślizgowym
  - lądowe pływające
- Dodanie **nowego aspektu życia**
  - oddychanie
  - rozmnażanie
  - ...

# Szukamy inspiracji

- Paradygmat OO
- **General Responsibility Assignment Software Principles**
- S.O.L.I.D.
  - Dziedziczenie – BAD
  - Agregacja – GOOD
  - Kohezja
    - Spójna odpowiedzialność
    - Jeden powód do zmiany
  - Zmienność
    - Hermetyzacja poza stabilny interfejs
    - Polimorficzne wywołania





```
public void poruszSie(){
    this.strategiaPoruszaniaSie.prousz(this);
}

public void pozywSie(){
    this.strategiaOdzywianiaSie.nakarm(this);
}
```

# Wzorzec strategii

## Motywacja

- Istnieją odmiany zachowania
- Posiadają wspólny kontrakt

## Implementacja

- Agregacja zamiast dziedziczenia
  - Dziedziczenie może się przydać w hierarchii samej strategii
- Hermetyzacja zmienności poza stabilny interfejs

## Siły

- Spójna odpowiedzialność
- Otwartość na rozbudowę bez modyfikacji

# Wzorce Projektowe w logice biznesowej Enterprise?





# Wzorce Projektowe w logice biznesowej Enterprise?

To zależy od tego gdzie szukamy...

## **Podejście mainstream (proceduralne)**

- Anemiczna encje (struktury danych bez odpowiedzialności biznesowej)
- Serwisy (paczki procedur)

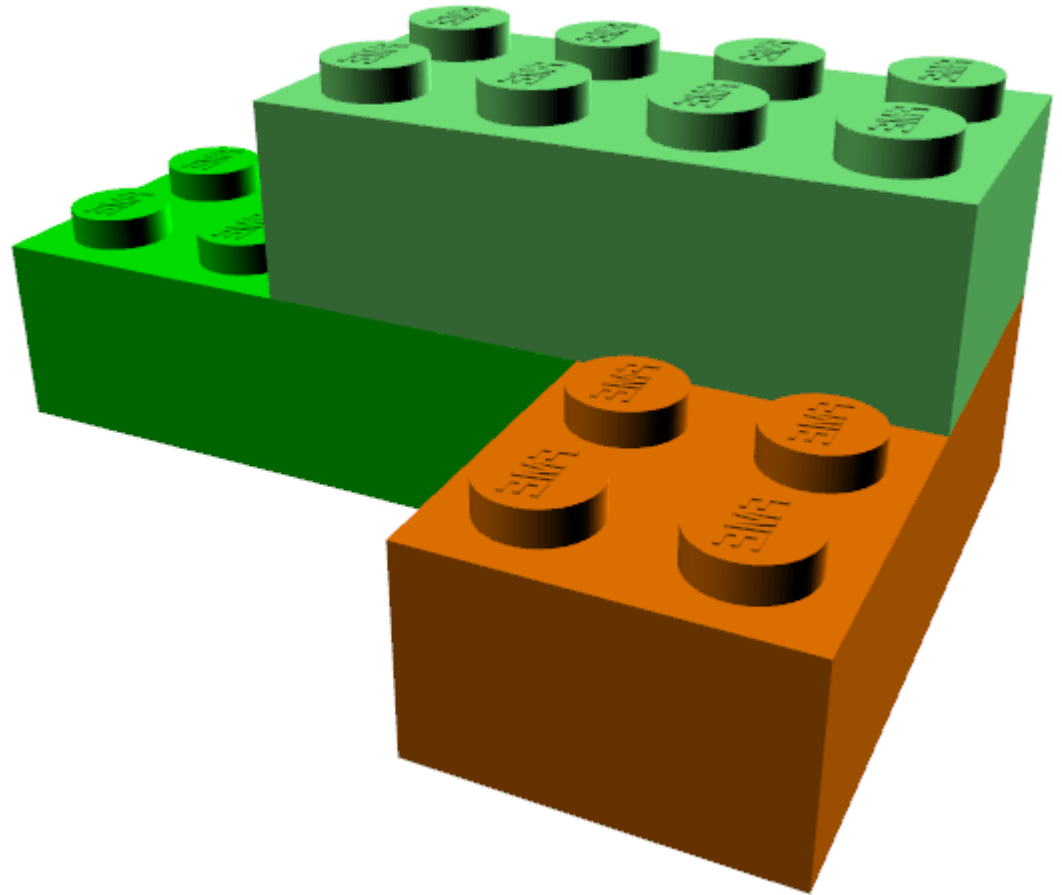
## **Przykłady w literaturze technicznej**

- Mają za zadanie ilustrowanie technikaliów (np. nowe adnotacje;)
- Nie powinny być dla profesjonalistów źródłem inspiracji odnośnie modelowania

# Domain Driven Design

## Szerszy wachlarz Building Blocks:

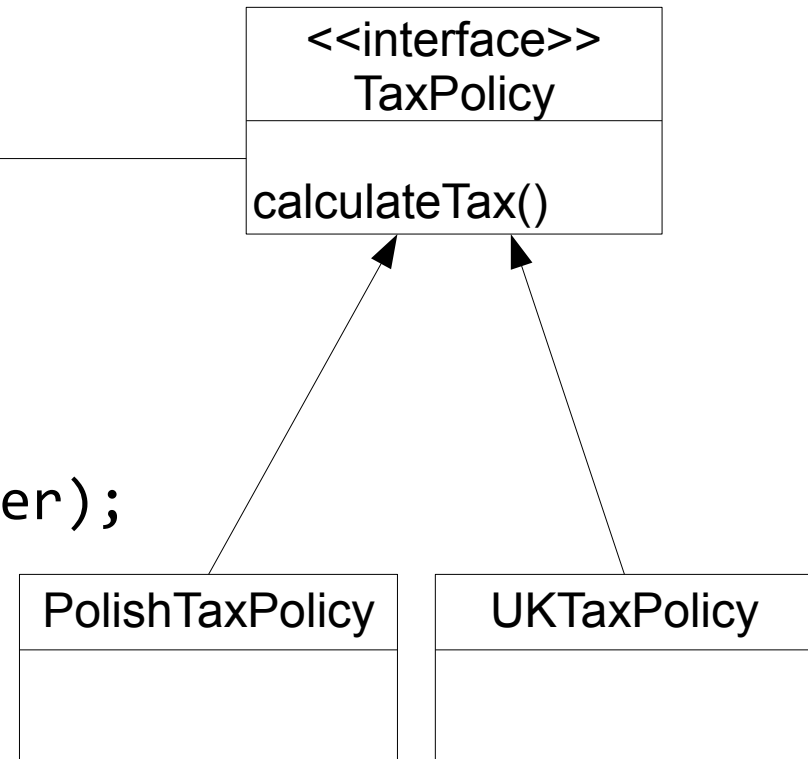
- Serwisy biznesowe
- Encje (nieanemiczne)
- Agregaty
- **Polityki – Wzorzec Strategii**
- Value Objects
- Fabryki
- Repozytoria
- Zdarzenia biznesowe



Wspólne koncepcje biznesowe pojawiające się na każdym poziomie abstrakcji:  
klient, analityk, developer

# Przykład

```
public class OrderService{  
    private TaxPolicy taxPolicy;  
    private RebatePolicy rebatePolicy;  
  
    public void submit(Order order){  
        ...  
        rebatePolicy.calculateRebate(order);  
        ...  
        taxPolicy.calculateTax(order);  
        ...  
    }  
}
```



Przykład w stylu klasycznym - w DDD encje Order posiadałby odpowiedzialność biznesową.

# Wstrzyknięcie polityki

Daj servis

Oto servis

## Kontener/Kontekst



OrderService

setPolicy

PolishTaxPolicy



# Przykład w kontekście Seam

```
@Name(„orderService”)
public class OrderService{
    @In
    private TaxPolicy taxPolicy;
    @In
    private RebatePolicy rebatePolicy;

    public void submit(Order order){
        ...
        rebatePolicy.calculateRebate(order);
        ...
        taxPolicy.calculateTax(order);
        ...
    }
}
```

# Przykład w kontekście Seam



`@Name(„taxPolicy”)`

```
public class PolishTaxPolicy implements TaxPolicy{  
    ...  
}
```



```
<component name="taxPolicy" class="x.y.PolishTaxPolicy">
```



```
public class PolicyFactories{  
    @Factory(„taxPolicy”)  
    public TaxPolicy getTaxPolicy() {  
        return ... ;  
    }  
}
```

# Przykład w kontekście Spring

**@Component**

```
public class OrderService{  
    private TaxPolicy taxPolicy;  
    private RebatePolicy rebatePolicy;
```

**@Autowired**

```
    public OrderService(TaxPolicy tp, RebatePolicy rp){  
        this.taxPolicy = tp; this.rebatePolicy=rp;  
    }
```

**@Transactional**

```
    public void submit(Order order){  
        ...  
        rebatePolicy.calculateRebate(order);  
        ...  
        taxPolicy.calculateTax(order);  
        ...  
    }  
}
```

# Przykład w kontekście Spring



`@Component(„taxPolicy”)`

```
public class PolishTaxPolicy implements TaxPolicy{  
    ...  
}
```



```
<bean id="taxPolicy" class="x.y.PolishTaxPolicy">
```



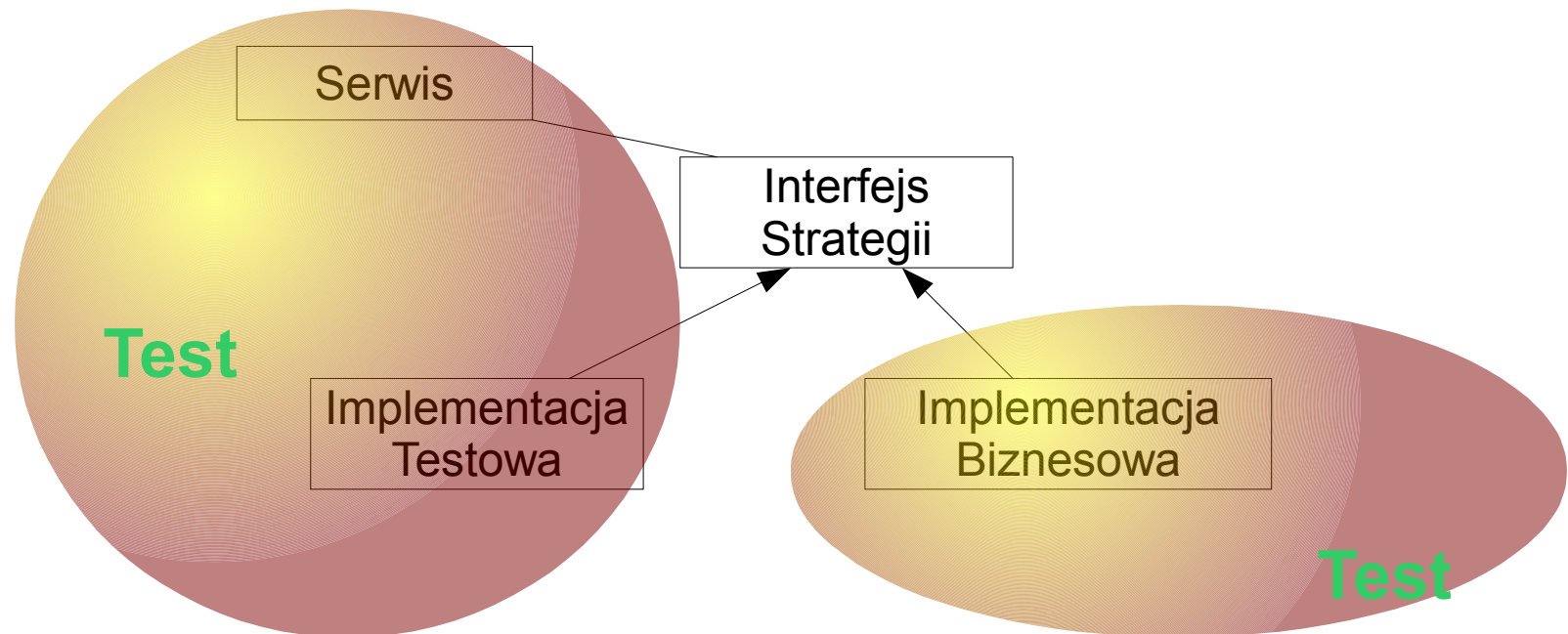
`@Configuration`

```
public class PolicyFactories{  
    @Bean  
    public TaxPolicy taxPolicy() {  
        return ... ;  
    }  
}
```

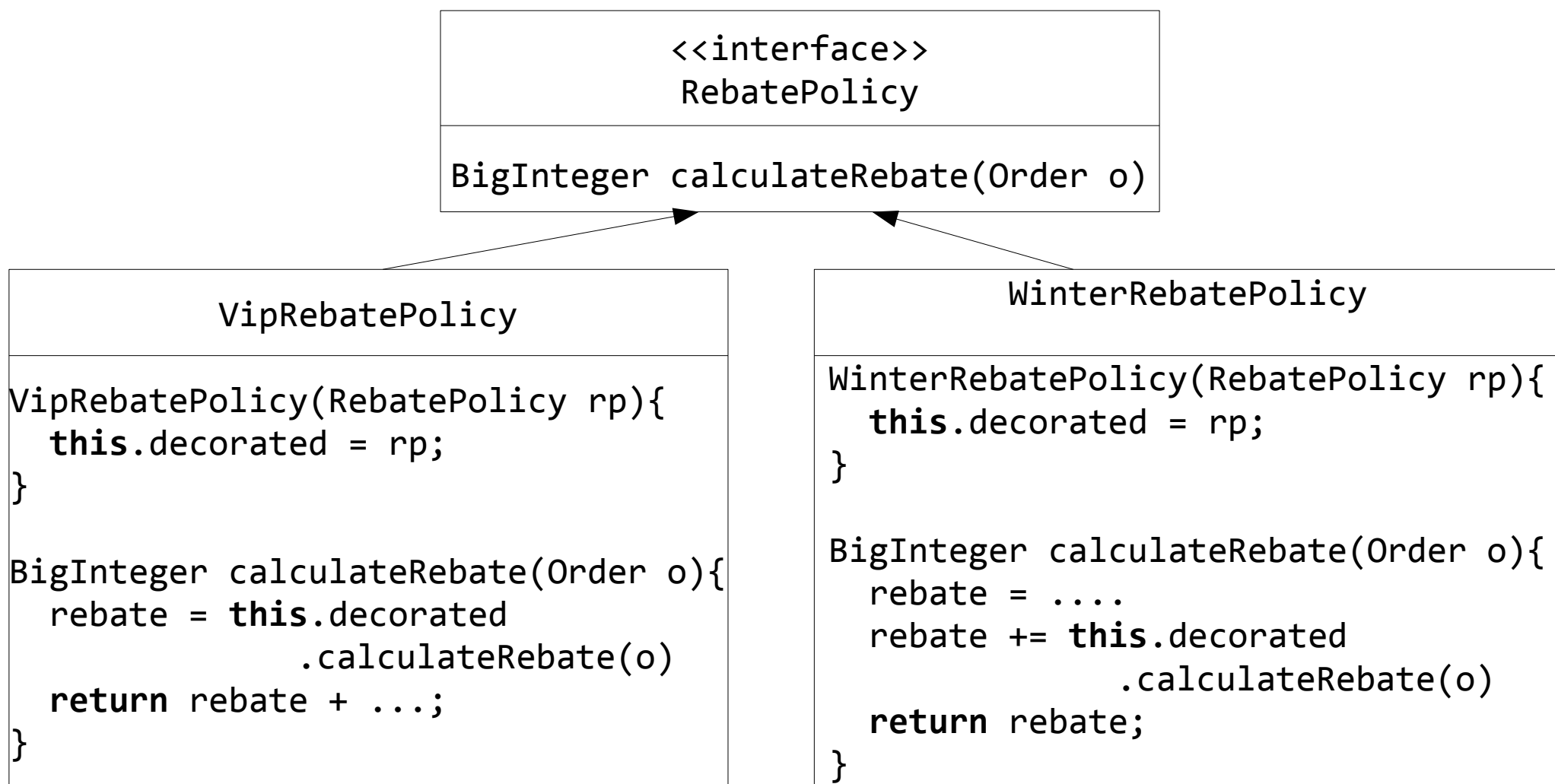


# Rozbudowa i Testability

- Rozbudowa o nowe klasy bez modyfikacji istniejących
- Składowe wzorca charakteryzują się wysoką kohezją
  - Testy jednostkowe klas o spójnej odpowiedzialności
  - Wstrzyknięcie Mock/Stub na czas testów jednostkowych

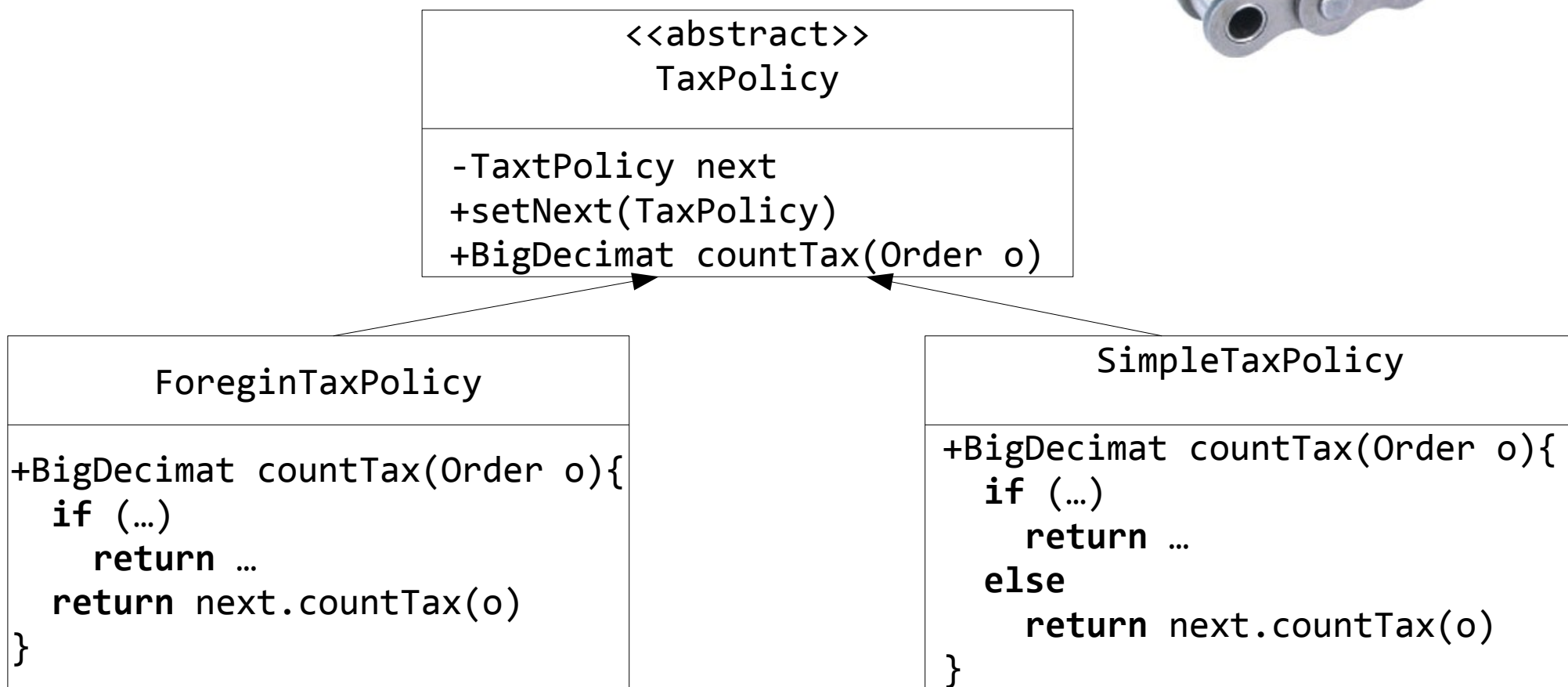


# Dekorowanie złożonej logiki



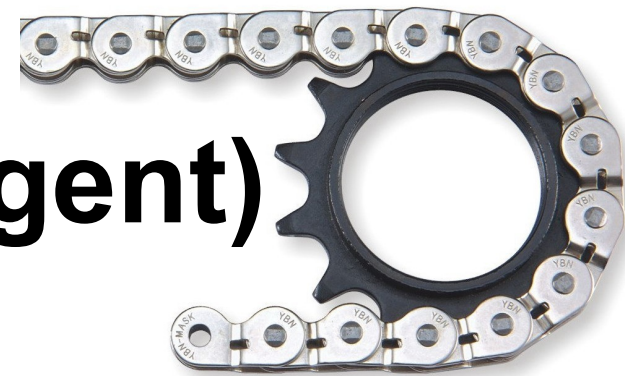
```
RebateCounter rc = new VipRebateCounter(new WinterRebateCounter());
rc.countRebate(order);
```

# Łańcuch odpowiedzialności



```
TaxPolicy tp = new ForeginTaxPolicy();
tp.setNext(new SimpleTaxPolicy());
...
tp.countTax(order);
```

# Łańcuch – modyfikacja (agent)



```
<<interface>>  
TaxHandler
```

```
+boolean canHandle(Order o)  
+BigDecimal countTax(Order o)
```

```
public class TaxCounter implements TaxPolicy{  
    private List<TaxHandler> handlers  
  
    public BigDecimal count(Order o){  
        for (TaxHandler th : handlers)  
            if (th.canHandle(o))  
                return th.countTax(o);  
  
        return new BigDecimal(0);  
    }  
}
```

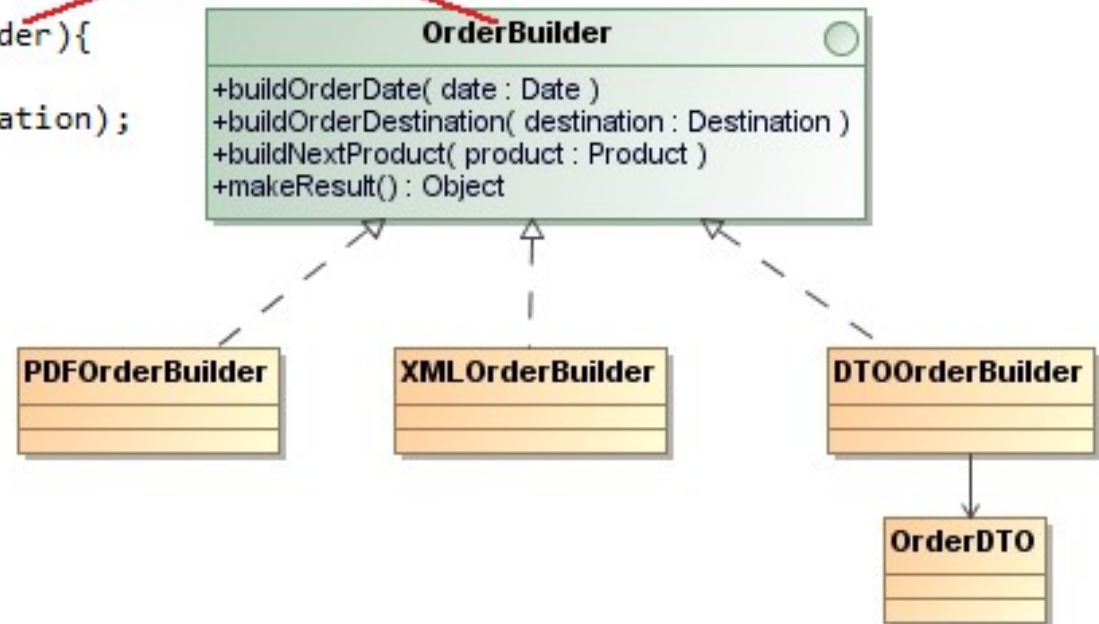


# Budowniczy



```
public class Order{  
    private Date date;  
    private List<Product> products;  
    private Destination destination;
```

```
    public Object export(OrderBuilder builder){  
        builder.buildOrderDate(date);  
        builder.buildOrderDestination(destination);  
  
        for (Product p : products){  
            builder.buildNextProduct(p);  
        }  
  
        return b.makeResult();  
    }  
  
    //metody biznesowe  
}
```



Kierownik budowy

Konkretni budowniczkowie

# Ogólny wzorzec

## Zmiana klasycznego podejścia:

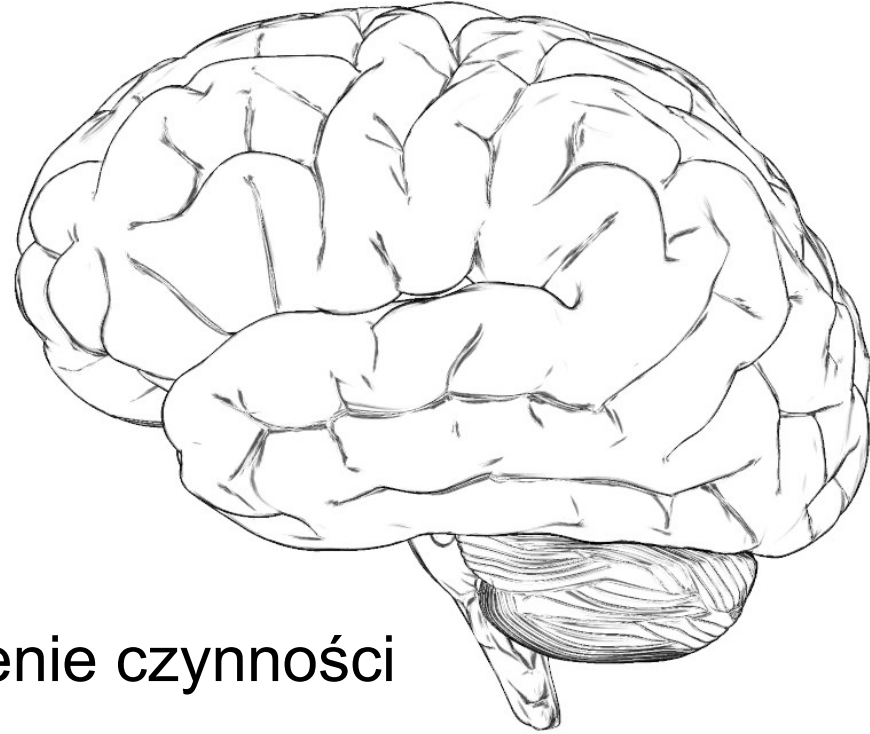
- rzeczowniki – klasy
- czasowniki - metody

## Czynność jest obiektem!

- metoda jedynie inicjuje uruchomienie czynności

## Potencjał do stosowania technik OO dla czynności

- polimorficzne wykonanie
- reużycie



# Uwaga aby nie przesadzić!

```
public interface ProblemSolver{  
    public Solution solve(Problem p)  
}
```



# Wzorce – słownictwo branżowe

- Wyższy poziom abstrakcji
- Rodzaj standaryzacji kodu - łatwe poruszanie się po frameworkach i bibliotekach
- Kod samodokumentujący się (na poziomie koncepcji)

Zaimplementuj to, proszę, jako  
**fabrykę dekorowanych strategii**  
wstrzykiwanych do serwisów

Koleżko, nie musisz mi tego  
dwa razy powtarzać!  
mówisz... masz!





# Wzorce – wsparcie dla Metafory Systemu





# Rozszerzony katalog wzorców

## Kreacyjne:

- Abstract factory
- Builder
- Factory method
- Lazy initialization
- Multiton
- Object pool
- Prototype
- Singleton

## Strukturalne:

- Adapter (Wrapper)
- Bridge
- Composite
- Decorator
- Facade
- Flyweight
- Proxy

## Behawioralne:

- Chain of responsibility
- Command
- Interpreter
- Iterator
- Mediator
- Memento
- Null object (Guardian)
- Observer (publisher/subscriber)
- Blackboard (uogólniony Observer)
- Specification
- State
- Strategy
- Template method
- Visitor

# Mity na temat wzorców projektowych





# Mity na temat wzorców projektowych

**Przejmują stery i odbierają inwencję programistom**

- wzorce to jedynie abstrakcyjny kierunek do rozwiązania
- wzorce nadają formę, nie ingerując w treść



# Mity na temat wzorców projektowych

## Przejmują stery i odbierają inwencję programistom

- wzorce to jedynie abstrakcyjny kierunek do rozwiązania
- wzorce nadają formę, nie ingerując w treść

## Wprowadzają niepotrzebną komplikację kodu

- wzorce nie zawsze są **zasadne** (przerost formy nad treścią)
- złożoność właściwa/złożoność przypadkowa





# Mity na temat wzorców projektowych

## Przejmują stery i odbierają inwencję programistom

- wzorce to jedynie abstrakcyjny kierunek do rozwiązania
- wzorce nadają formę, nie ingerując w treść

## Wprowadzają niepotrzebną komplikację kodu

- wzorce nie zawsze są **zasadne** (przerost formy nad treścią)
- złożoność właściwa/złożoność przypadkowa

## Powodują wydłużenie czasu developmentu

- ilość kodu biznesowego nie zwiększa się (ew zmniejsza)
- dodajemy jedynie więcej „pudełek”
- **oszczędzamy** podczas rozbudowy



# Mity na temat wzorców projektowych

## **Przejmują stery i odbierają inwencję programistom**

- wzorce to jedynie abstrakcyjny kierunek do rozwiązania
- wzorce nadają formę, nie ingerując w treść

## **Wprowadzają niepotrzebną komplikację kodu**

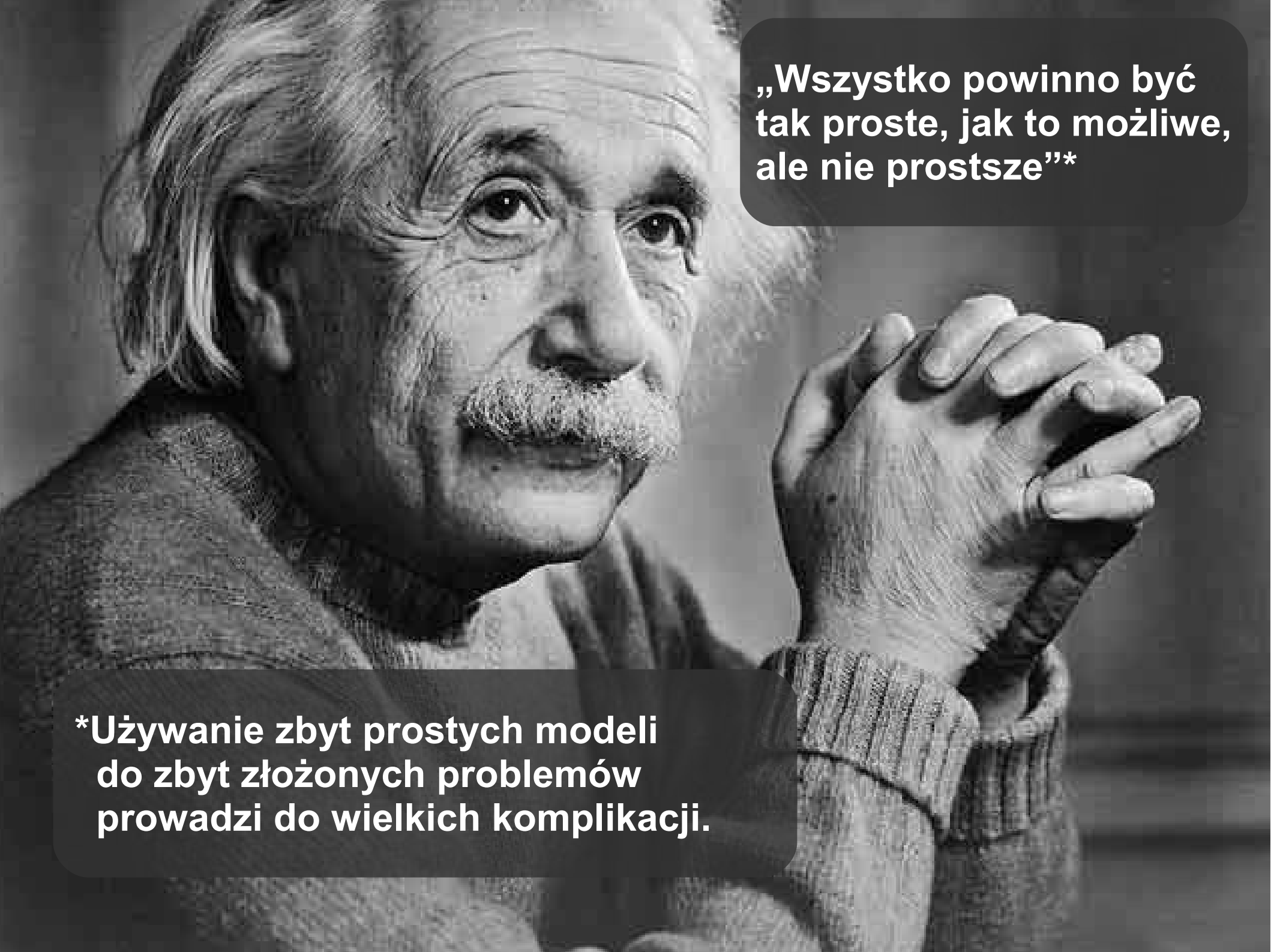
- wzorce nie zawsze są **zasadne** (przerost formy nad treścią)
- złożoność właściwa/złożoność przypadkowa

## **Powodują wydłużenie czasu developmentu**

- ilość kodu biznesowego nie zwiększa się (ew zmniejsza)
- dodajemy jedynie więcej „pudełek”
- **oszczędzamy** podczas rozbudowy

## **Ich potrzeba wynika z niedostatków języków „wąsatych”**

- TAK!

A black and white portrait of Albert Einstein, showing him from the chest up. He has his characteristic wild hair and a mustache. His hands are clasped together in front of him, with his fingers interlaced. He is looking slightly to the right of the camera with a thoughtful expression.

**„Wszystko powinno być  
tak proste, jak to możliwe,  
ale nie prostsze”\***

**\*Używanie zbyt prostych modeli  
do zbyt złożonych problemów  
prowadzi do wielkich komplikacji.**

## Dziękuję za uwagę

więcej...



<http://art-of-software.blogspot.com>  
[slawomir.sobotka@bottega.com.pl](mailto:slawomir.sobotka@bottega.com.pl)



# Photo Credits

- <http://www.jadegalore.com/html/galleria/img/craftsmanship5.jpg>
- <http://runningagile.files.wordpress.com/2009/02/craftsmanship.jpg>
- [http://www.maybachusa.com/graphics/level\\_one/57/features/maybach\\_craftsmanship.jpg](http://www.maybachusa.com/graphics/level_one/57/features/maybach_craftsmanship.jpg)
- [http://www.goiam.org/uploadedImages/Pick\\_A\\_Fight/China\\_Trade/7589321.jpg](http://www.goiam.org/uploadedImages/Pick_A_Fight/China_Trade/7589321.jpg)
- [http://www1.assumption.edu/users/McClymer/his261/Spiting\\_Himself.gif](http://www1.assumption.edu/users/McClymer/his261/Spiting_Himself.gif)
- <http://files.myopera.com/sprogger/albums/180133/SeaMine.jpg>
- <http://dominicambrose.files.wordpress.com/2009/05/thebighouse.jpg>
- <http://www.mcescher.com/>
- <http://www.laputan.org/images/pictures/gof-car.jpg>
- <http://www.fmepedia.com/index.php/LegoBrickCreator>
- [http://ifitsgotanengine.com/wp-content/uploads/2009/06/motorcycle\\_chain.jpg](http://ifitsgotanengine.com/wp-content/uploads/2009/06/motorcycle_chain.jpg)
- [http://www.global-trade.com.tw/images/Product/BFbicycle226560chain\(1\).jpg](http://www.global-trade.com.tw/images/Product/BFbicycle226560chain(1).jpg)
- [http://www.grafpak.pl/img/karty\\_menu/menu\\_A4\\_4d.jpg](http://www.grafpak.pl/img/karty_menu/menu_A4_4d.jpg)
- <http://www.cs.princeton.edu/gfx/proj/sugcon/models/brain.png>
- [http://images.fanpop.com/images/image\\_uploads/The-IT-Crowd-the-it-crowd-231621\\_640\\_352.jpg](http://images.fanpop.com/images/image_uploads/The-IT-Crowd-the-it-crowd-231621_640_352.jpg)
- [http://englishrussia.com/images/mig\\_factory/1.jpg](http://englishrussia.com/images/mig_factory/1.jpg)
- <http://iwassaying.net/wp-content/uploads/2009/11/highway.jpg>
- [http://img.alibaba.com/photo/11675866/Linksys\\_wireless\\_Router.jpg](http://img.alibaba.com/photo/11675866/Linksys_wireless_Router.jpg)
- <http://www.pbs.org/wgbh/buildingbig/images/bridge/basics/trussbridge.gif>
- [http://www.technochitlins.com/archives/rainbow\\_elam\\_2.jpg](http://www.technochitlins.com/archives/rainbow_elam_2.jpg)
- <http://www.aktywnysmyk.pl/images/puky/z2/red-detale/Z2-red-boczne-kolka-tyl-2.jpg>