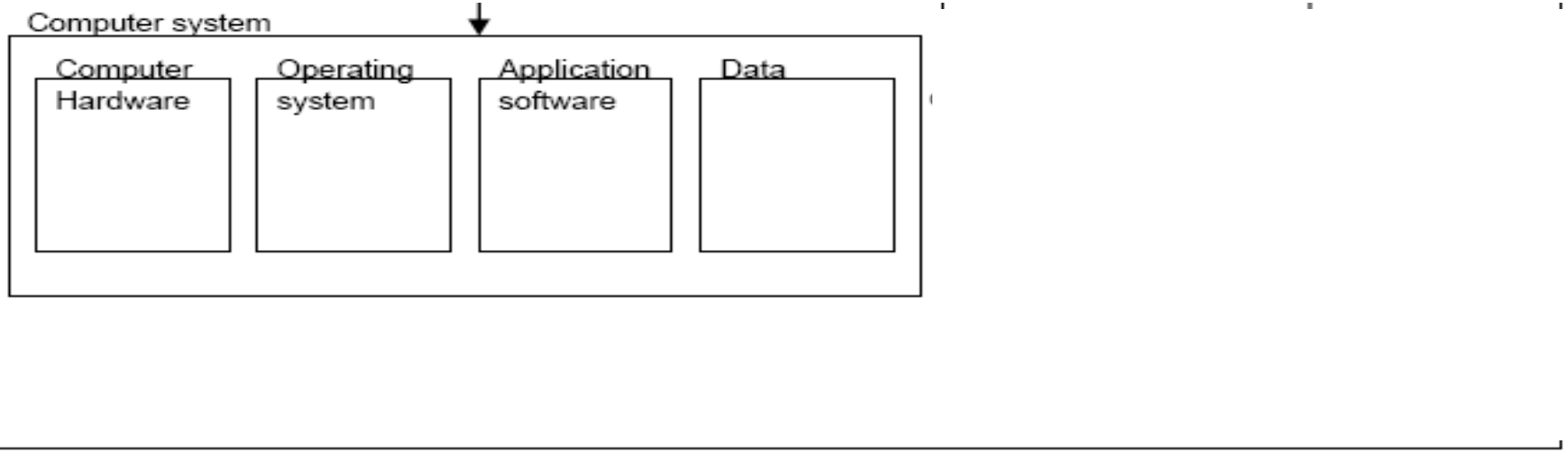
Modelowanie i analiza biznesowa

Prof. Zbigniew Huzar
Katedra Informatyki
Wydział Informatyki i Zarządzania

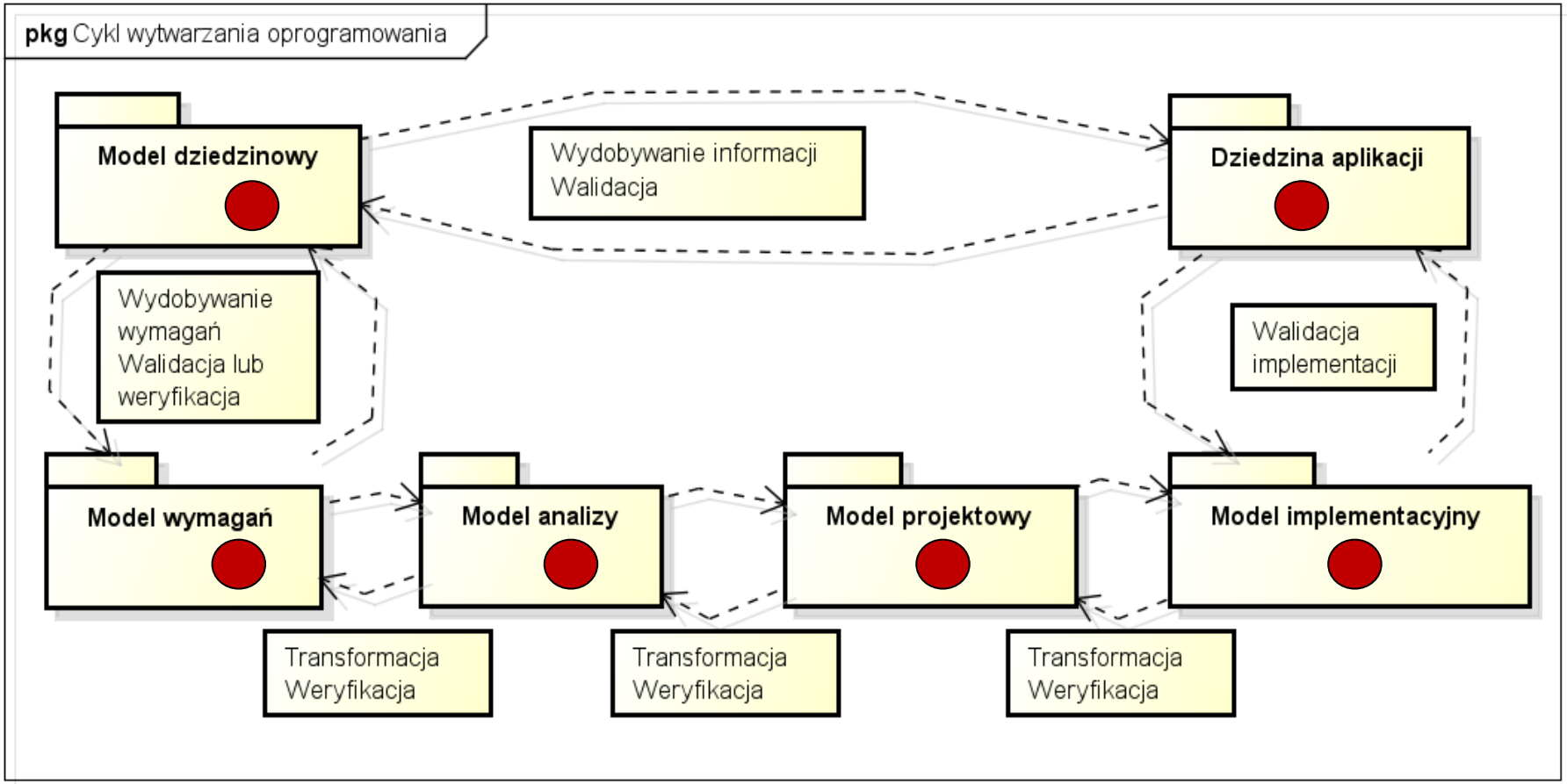
System informatyczny

- Aspekt statyczny:
  - Sprzęt – węzły, kanały komunikacji
  - Oprogramowanie – komponenty posadowione na węzłach
- Aspekt dynamiczny:
  - Interakcje pomiędzy otoczeniem a systemem przez dobrze zdefiniowane interfejsy
- Cel – gromadzenie i przetwarzanie danych zgodnie z oczekiwaniami użytkowników

System informatyczny



Cykl wytwarzania systemów oprogramowania

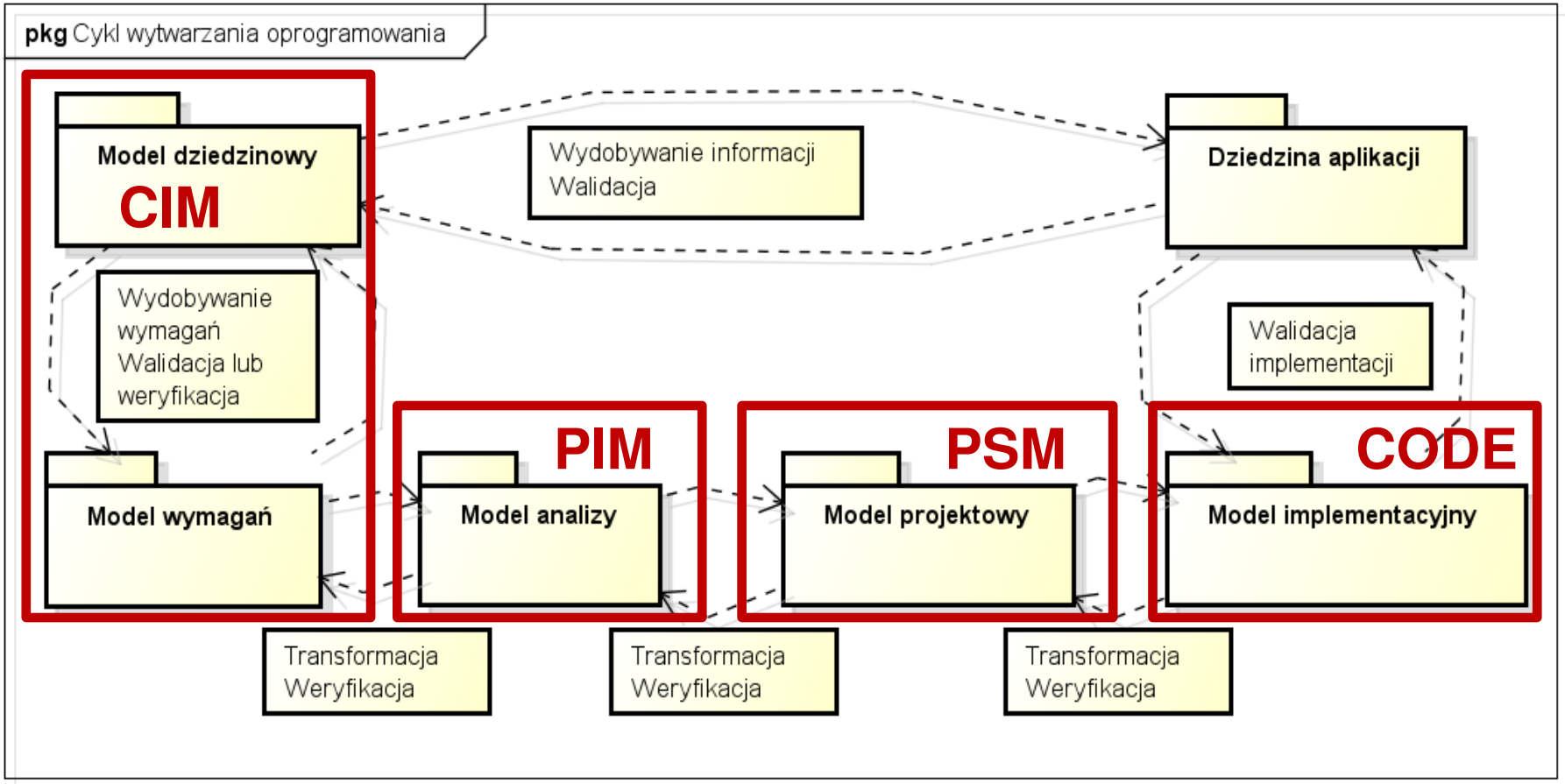


Paradygmaty wytwarzania

- Podejście obiektowe
- Podejście oparte na modelowaniu

  MDA – Model Driven Architecture (OMG)
  - CIM – Computer Independent Model
  - PIM – Platform Independent Model
  - PSM – Platform Specific Model
  - Kod (źródłowy)

MDA a cykl klasyczny



Modelowanie

- Modelowanie – proces budowy modeli
- Model – abstrakcyjna (uproszczona) wizja pewnego rzeczywistego lub wyobrażanego bytu
- Model zależy od przyjętego celu i perspektywy modelowania
- Perspektywa (wybrane istotne właściwości) wynika z celu modelowania
- Reprezentacja modelu
  - tekstowa – tekst formalny/nieformalny
  - graficzna
  - fizyczna

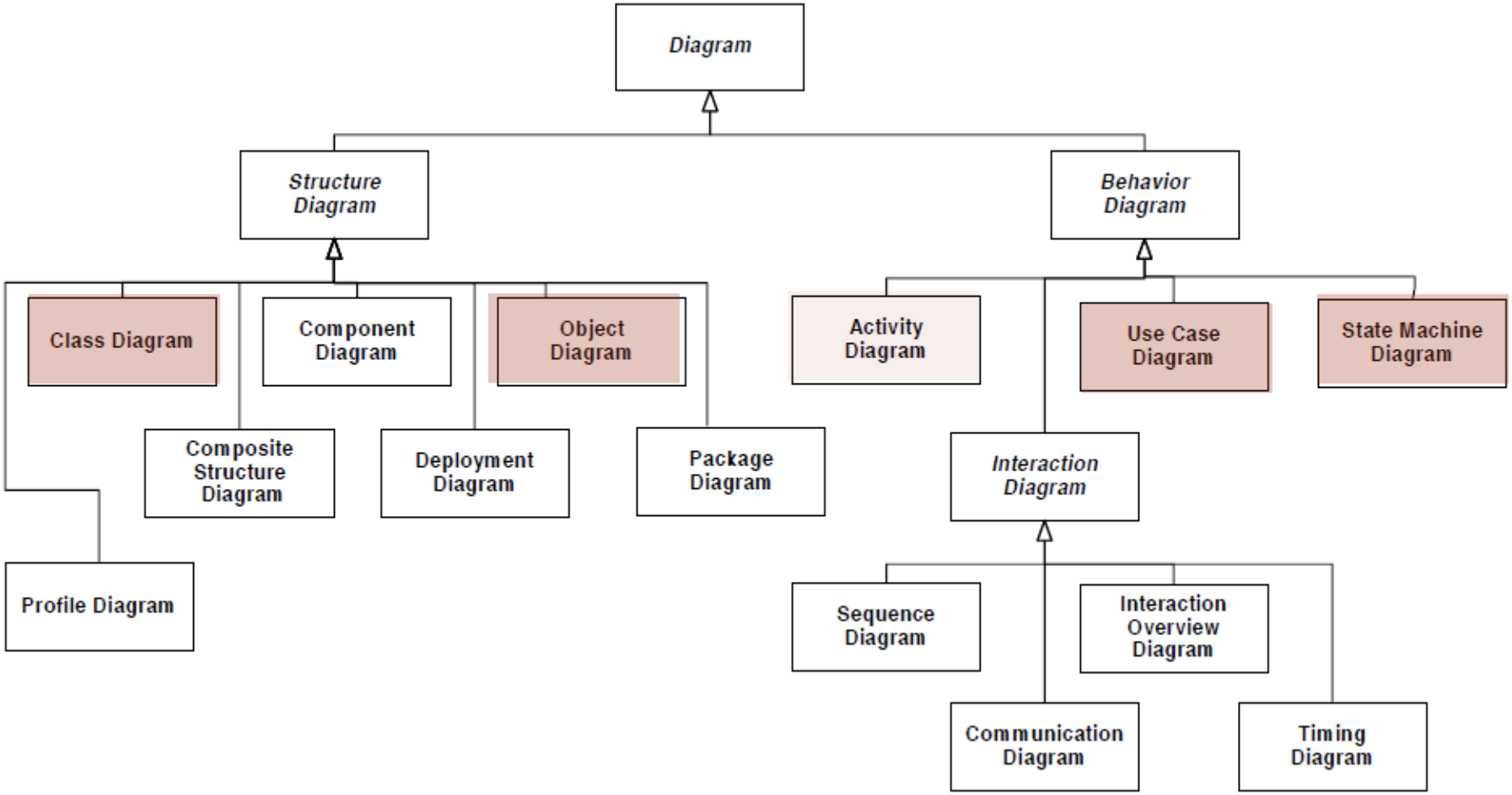
UML – modele

- Model – zbiór diagramów reprezentujących statyczny i dynamiczny aspekt modelowanego bytu
- Diagram – graf skierowany z etykietowanymi wierzchołkami i łukami

UML – modele

- Jawnie nie wskazuje, ale dopuszcza modelowanie z perspektywy:
  - analizy dziedzinowej (biznesowej)
  - analizy systemowej
  - projektowej
  - implementacyjnej

UML – diagramy



Elementy modelowania

- Dwie kategorie elementów modelowania w UML
- Elementy reprezentujące (modelujące) byty zbiorowe (grupowe) – są to tzw. klasyfikatory
  - np. klasa może reprezentować zbiór osób
- Elementy reprezentujące (modelujące) byty indywidualne (pojedyncze)
  - np. obiekt może reprezentować daną osobę

Platon (427-347 p.n.e) – idee i cienie idei (klasy i obiekty)

Zbiór – określenie

- **Zbiór** – nieuporządkowany zestaw różnych obiektów, czy też kolekcja niepowtarzających się komponentów bez wyróżnionej kolejności.
- Jest jednoznacznie wyznaczony przez jego składowe nazywane jego *elementami* (tzn. istnieje tylko jeden zbiór złożony z zadanych elementów); każdy element może należeć do danego zbioru bądź nie (element nie może należeć do zbioru np. „dwukrotnie”.
- **Wielozbiór** – uogólnienie pojęcia zbioru; dany element może występować wiele razy.

Zbiór – sposoby definiowania

- Definicja intensjonalna
  - Jawne wyliczenie elementów, np.
    $X = \{x_1, x_2, \ldots, x_n\}$
  - Konstrukcja elementów (rekursja)
    1. Założenie: $3, 5 \in Y$
    2. Jeżeli $x, y \in Y$, to także $x+y \in Y$
- Definicja ekstensjonalna

  $Z = \{z \mid W(z)\}$

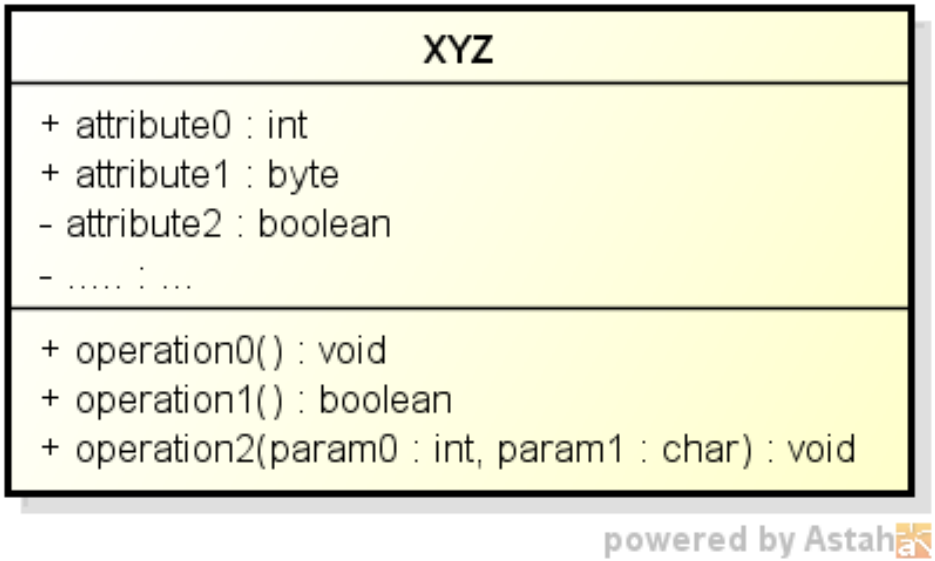
  np. $Z = \{z \mid z \in Nat \wedge$ „z dzieli się przez 7”$\}$

Diagram klas

- Wierzchołki:
  - klasy
- Łuki:
  - asocjacje, generalizacje, zależności
- Wierzchołki-łuki:
  - klasa asocjacji
- Ograniczenia
  - standardowe
  - definiowane

Klasa nieinterpretowana

- Składnia

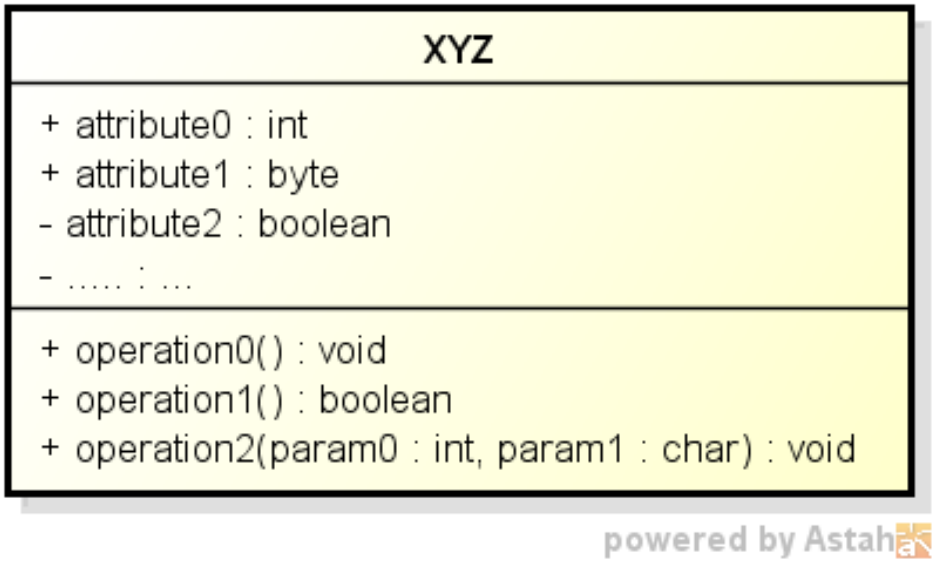


- Znaczenie klasy nieinterpretowanej
  - wzorzec do generacji obiektów

Uwaga:

Pojęcie klasy występuje również w filozofii i matematyce.

Klasa nieinterpretowana

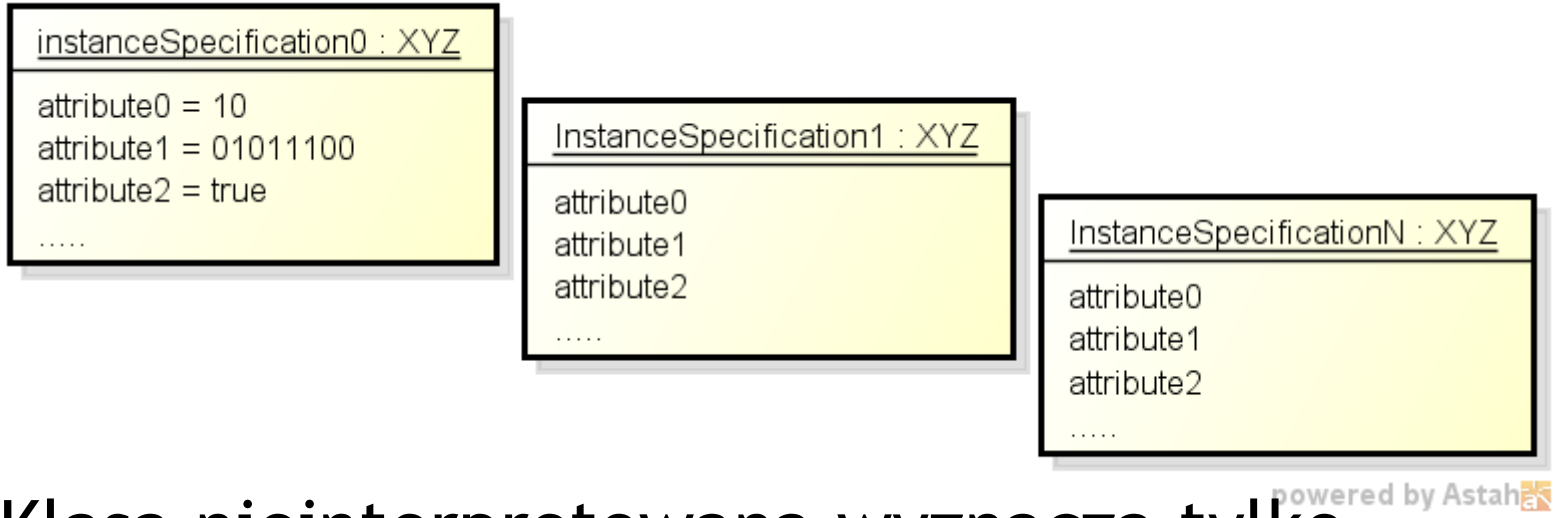


Oznaczenie:

Extent(XYZ)

jest zbiorem wszystkich możliwych do wygenerowania obiektów klasy XYZ, czyli tych bytów, które mają dokładnie te same własności, jakie określa definicja klasy

Klasa i jej instancje



Klasa nieinterpretowana wyznacza tylko

– zbiór swoich potencjalnych instancji

Extent(XYZ) (ekstensja klasy)

Klasa X jest abstrakcyjna, gdy Extent(X) = ∅

Interpretacja klasy

Co klasa reprezentuje (modeluje)?

Oznaczenie:

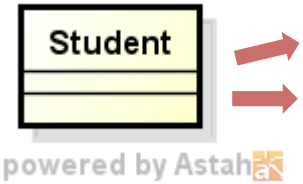
Sem(XYZ)

określa **zbiór** elementów w modelowanym wycinku rzeczywistości reprezentowany przez klasę XYZ.

Interpretacja klasy

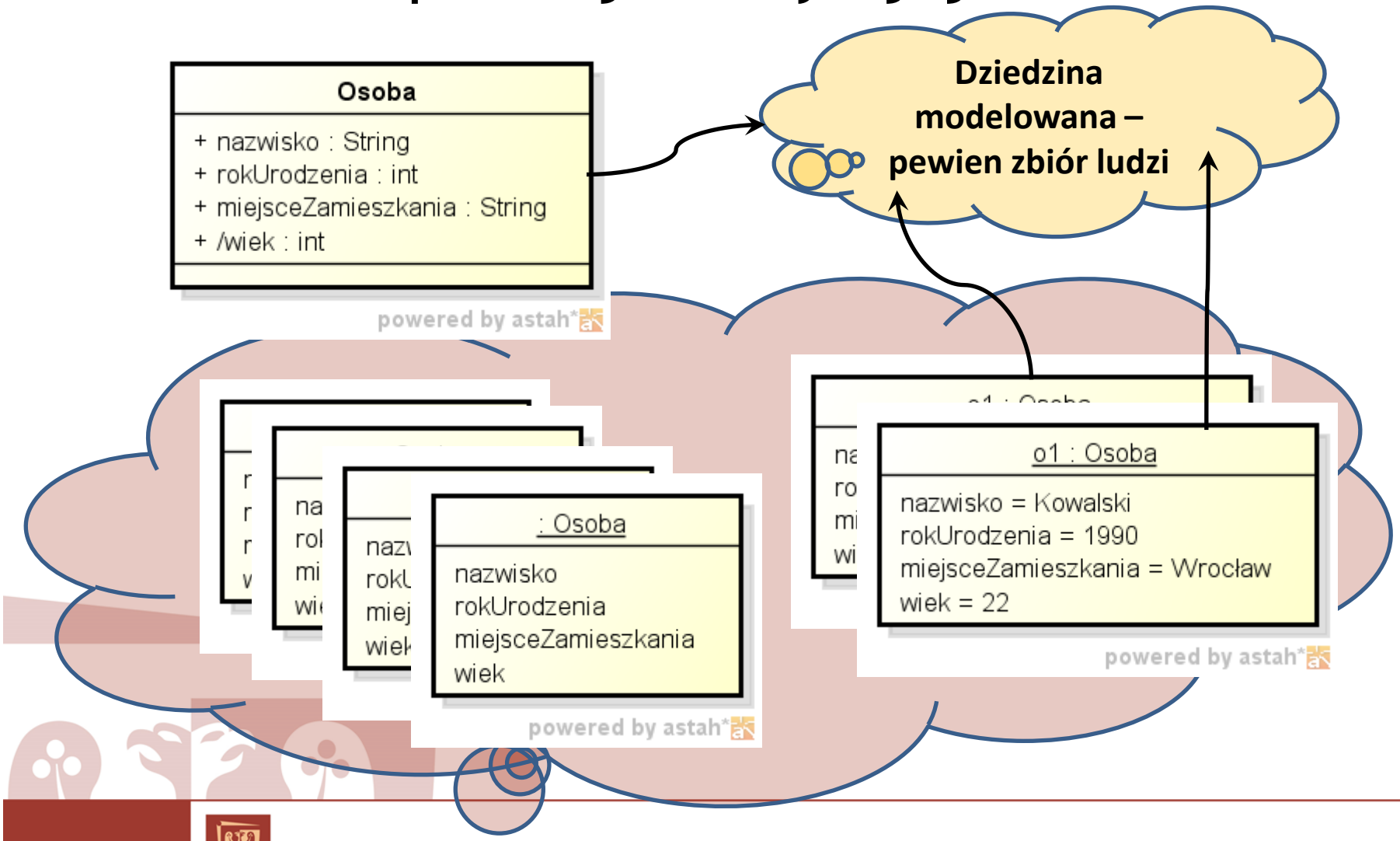
- Przykłady

Sem(Osoba) = Zbiór ludzi na świecie
Sem(Osoba) = Zbiór ludzi w Polsce
Sem(Osoba) = Zbiór ludzi w Polsce, w XIX wieku.
……



Sem(Student) = Zbiór studentów w UE
Sem(Student) = Zbiór studentów PWr
……..

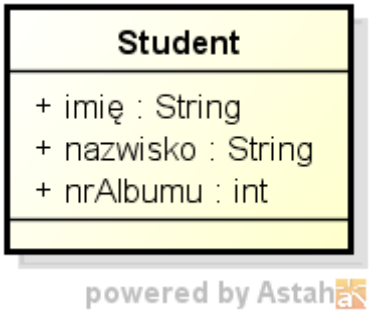
Interpretacja klasy i jej obiektów



Interpretacja klasy

- Jak określać atrybuty klasy?
- Przykład

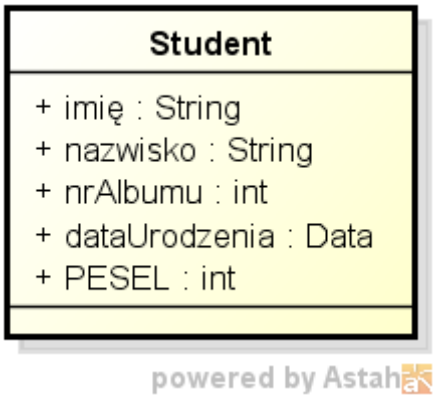
Niech reprezentuje zbiór studentów, którzy rozpoczęli studia w PWr w 2000 roku.



Czy jest to właściwy zestaw atrybutów?
-- pozwala odróżnić studentów
Jaki zestaw atrybutów będzie potrzebny dla systemu dziekanatowego?
Jaki zestaw atrybutów będzie potrzebny dla aplikacji samorządu studenckiego?
-- zależy od celu modelowania

Interpretacja klasy

- Ograniczenia dla atrybutów
- Przykłady



Niech reprezentuje zbiór studentów, którzy rozpoczęli studia w PWr w 2000 roku.

{dataUrodzenia < 1990}
{dataUrodzenia traktowana jako ciąg jest podciągiem PESEL}

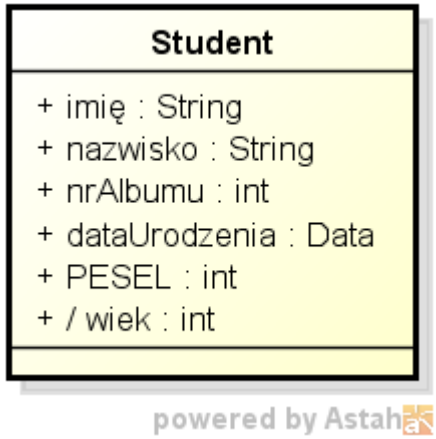
Własności klasy

- Wewnętrzne
  - atrybuty
  - operacje
  - metody
- Zewnętrzne
  - powiązania z innymi klasami

Reprezentacja własności klasy w OCL

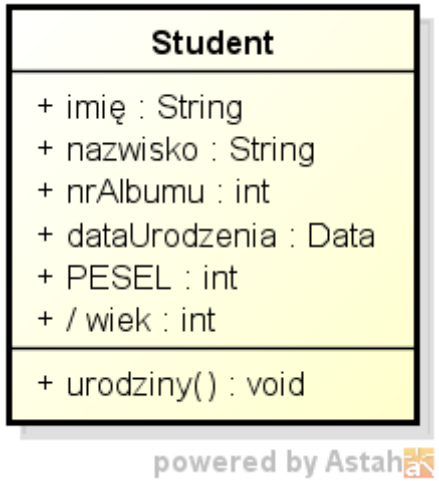
- Notacja OCL (Object Constraint Language)

  self – reprezentacja dowolnego obiektu danej klasy




```
context Student
    self.imię              -- typ: String
    self.nazwisko          -- typ: String
    self.nrAlbumu          -- typ: int
    self.dataUrodzenia     -- typ: Data
    self.wiek              -- typ: int
```

Reprezentacja ograniczeń klasy w OCL



context Student **inv:**

self.wiek ≥ 18 and self.wiek ≤ 70

context Student::urodziny()

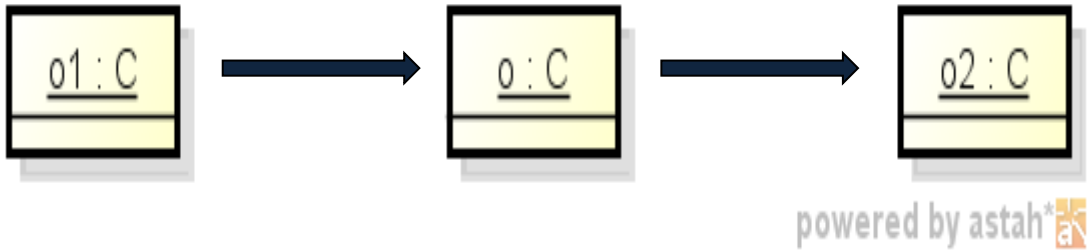
post self.wiek = self.wiek@pre + 1

Obiekt

- Byt o skończonym czasie istnienia
  (należy odróżniać czas życia obiektu od czasu życia bytu modelowanego przez dany obiekt)
- Odróżnialność i niezależność obiektów
- Enkapsulacja – ukrycie szczegółów wewnętrznych dotyczących struktury i zachowania
- Hermetyzacja – rozdzielenie interfejsu od jego realizacji

Obiekt

- Jednostka usługowa
  - świadczenie usług innym obiektom
  - korzystanie z usług innych obiektów



Obiekt

- Życie obiektu – ciąg stanów

  s_1 -> s_2 -> s_3 -> … -> s_n -> …

- Stan obiektu określony przez
  - wartościowanie atrybutów
  - powiązania z innymi obiektami
  - stopień zaawansowania wykonywanych operacji
- Zmiana stanu
  - zdarzenia odbioru wywołanie operacji lub odbioru sygnału

Obiekt

- Komunikacja pomiędzy obiektami
  - wywoływanie operacji
    - tryb synchroniczny
    - tryb asynchroniczny
  - przesyłanie sygnałów
    - tryb asynchroniczny

Obiekt a modelowany byt

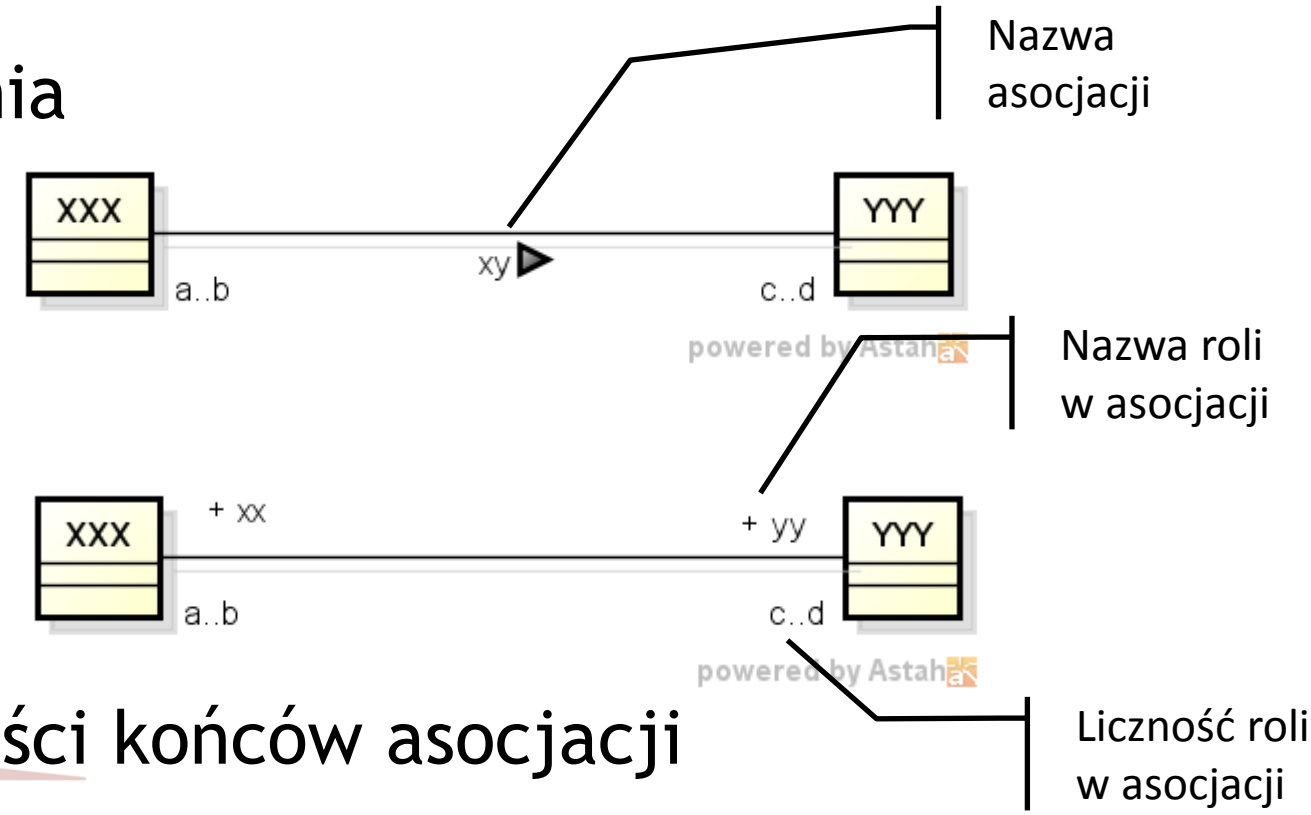
- Obiekt o : C powinien reprezentować (modelować) dokładnie jeden byt rzeczywisty z dziedziny interpretacji (modelowania) klasy C, czyli Sem(C)
- Oznacza to, że zbiór atrybutów klasy powinien być taki, aby wartościowanie atrybutów jednoznacznie określało konkretny byt rzeczywisty
- Zbiór interpretowanych obiektów klasy C jest podzbiorem ekstencji (uniwersum) klasy C, czyli zbioru Extent(C)

Diagram klas

- Wierzchołki:
  - klasy, interfejsy, aktorzy
- Łuki:
  - asocjacje, generalizacje, zależności
- Wierzchołki-łuki:
  - klasa asocjacji
- Ograniczenia
  - standardowe
  - specyficzne

Asocjacja – związek pomiędzy klasami

Składnia



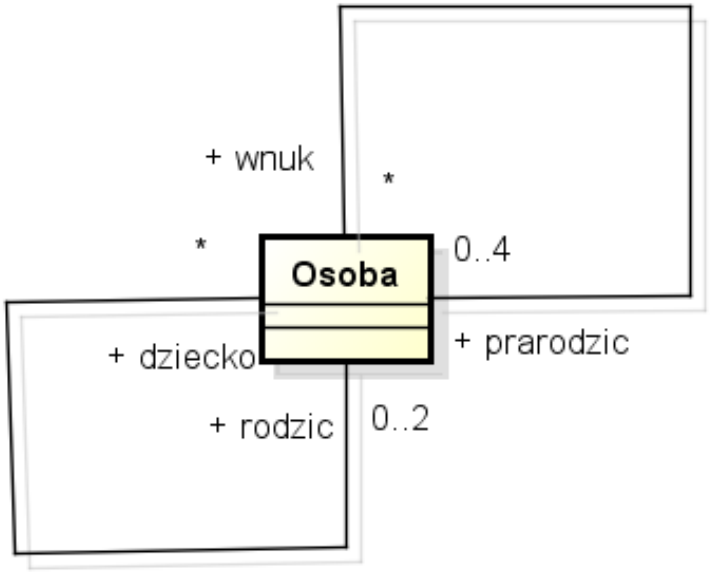
Liczności końców asocjacji

$a..b \subseteq Nat$

$c..d \subseteq Nat$

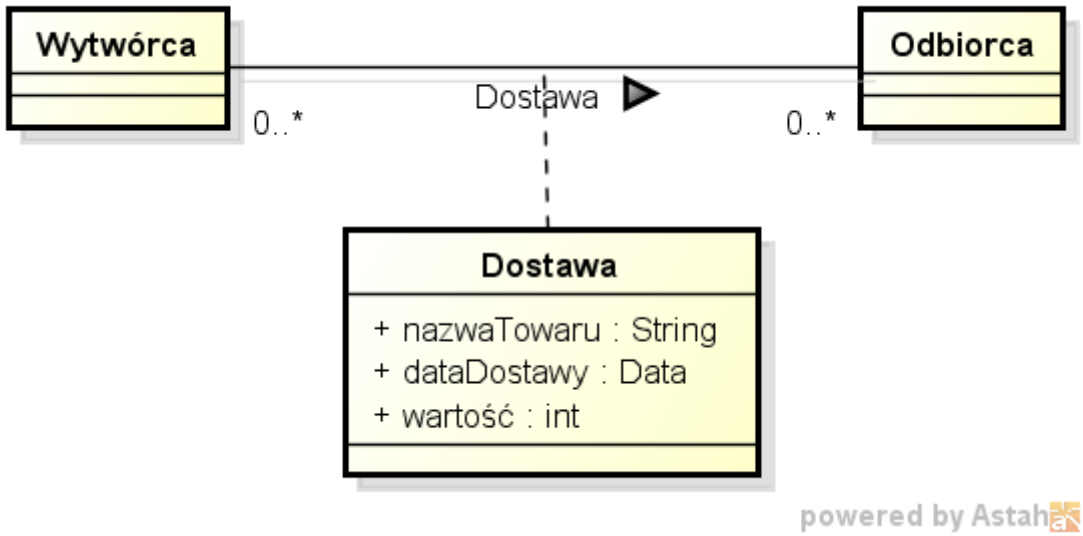
Składanie asocjacji – przykład

- Asocjacja ‘wnuk-prarodzic’ jest złożeniem asocjacji ‘dziecko-rodzic’ z ‘dziecko-rodzic’



Klasa asocjacji

Składnia



Klasa asocjacji

- Semantyka klasy asocjacji:
  - zbiór powiązanych par (instancji asocjacji)

    <w:Wytwórca, o:Odbiorca>

    z przypisanym każdej parze rekordem wartości typu (obiektem – instancją klasy):

    nazwa towaru : String

    nr partii : int

    dataDostawy : Data

Klasa asocjacji

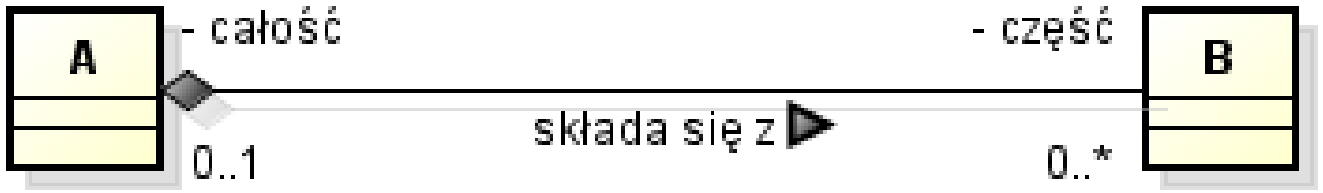
- Instancja klasy asocjacji może być rozumiana jako para:

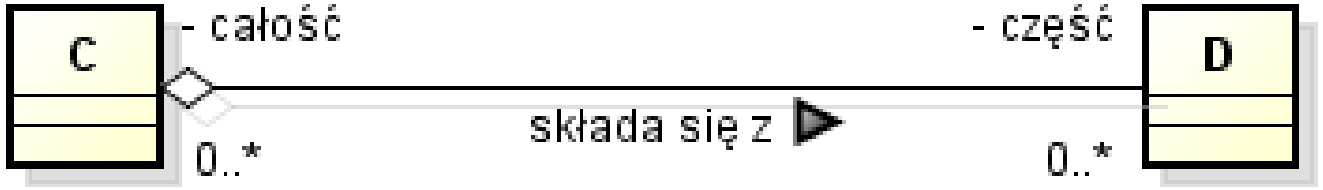
  <<w:Wytwórca, o:Odbiorca>, d: Dostawca>

  zaś semantyka klasy asocjacji jako zbiór takich par.

Kompozycja i agregacja

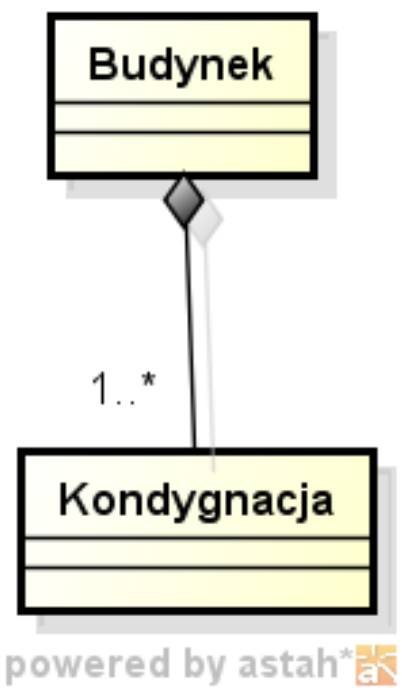
- Asocjacje typu „część-całość”

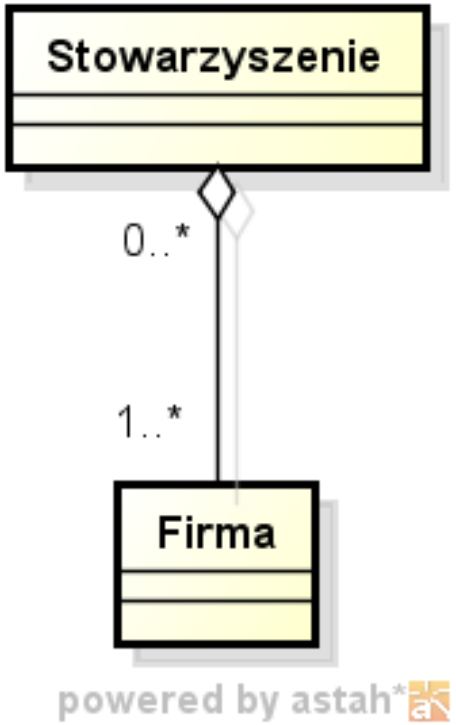




Kompozycja i agregacja

- Przykłady



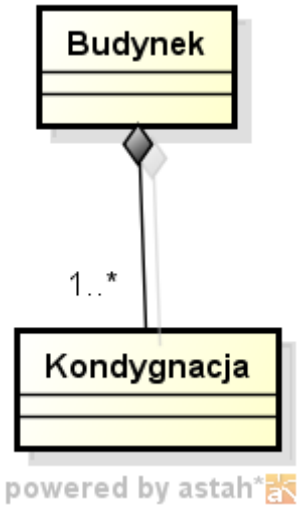


- Własności relacji
  - przeciwsymetria
  - przechodniość

Kompozycja

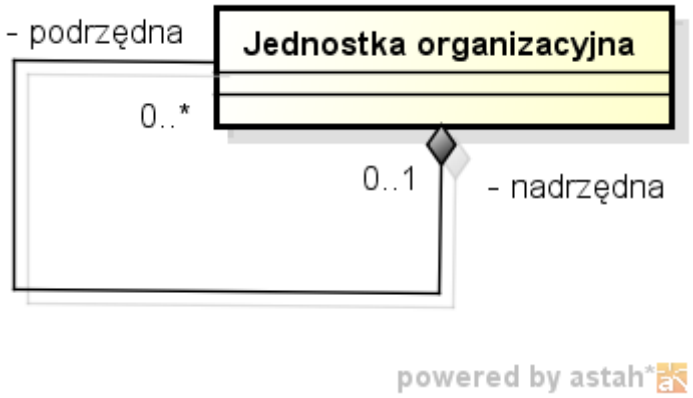
Ograniczenia

- obiekt składowy jest elementem co najwyżej jednego obiektu złożonego
- czas życia obiektu składowego jest fragmentem czasu życia obiektu złożonego



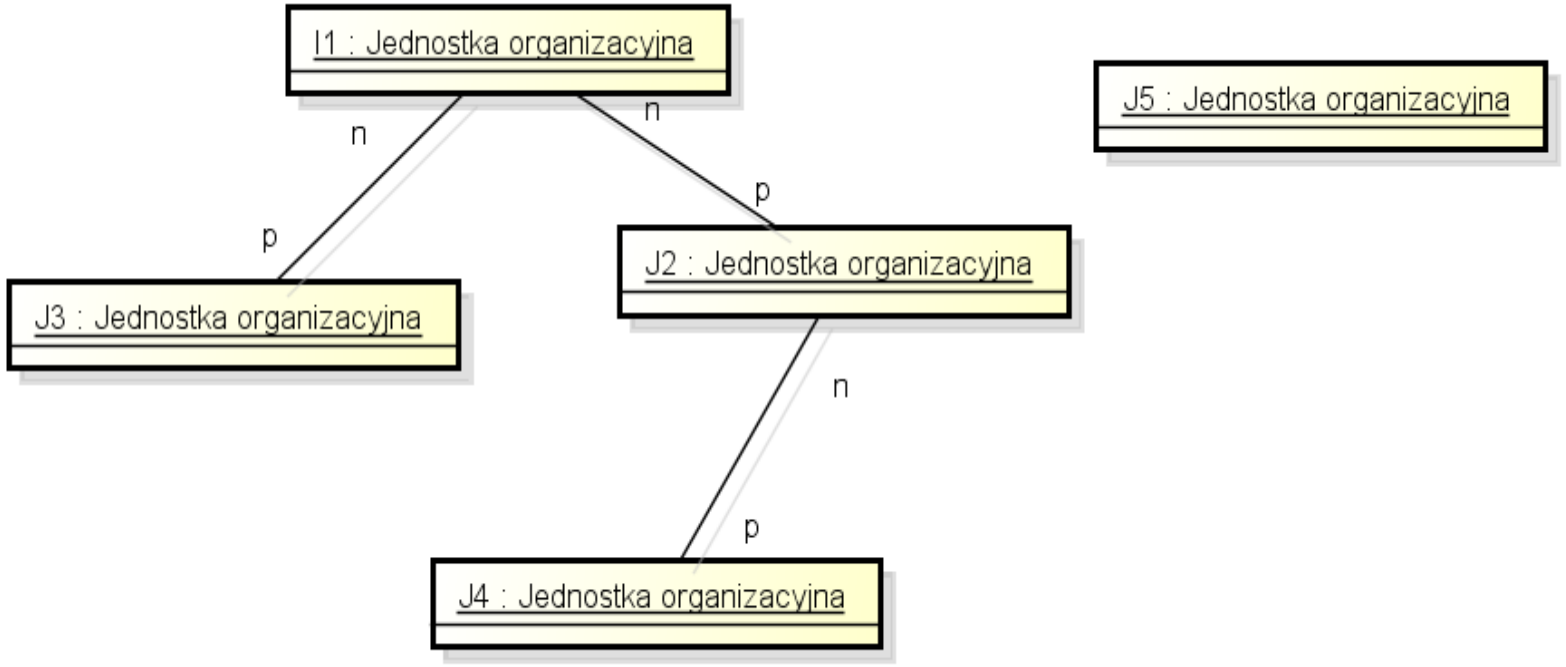
Kompozycja

Struktura hierarchiczna



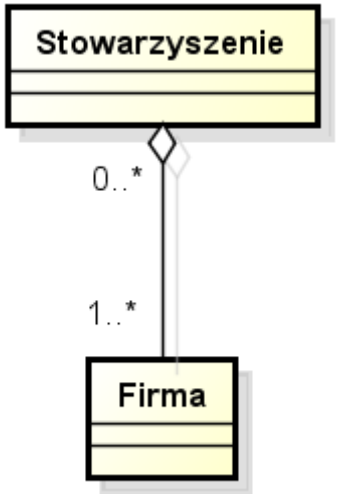
Kompozycja

Rekursja – struktura hierarchiczna

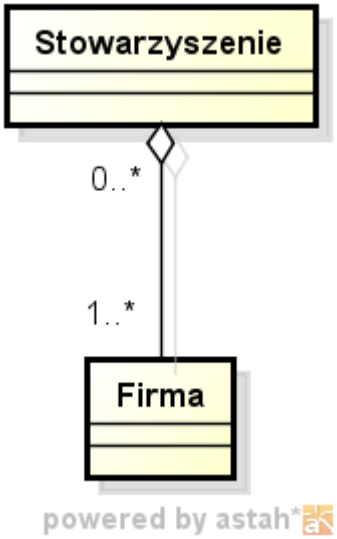


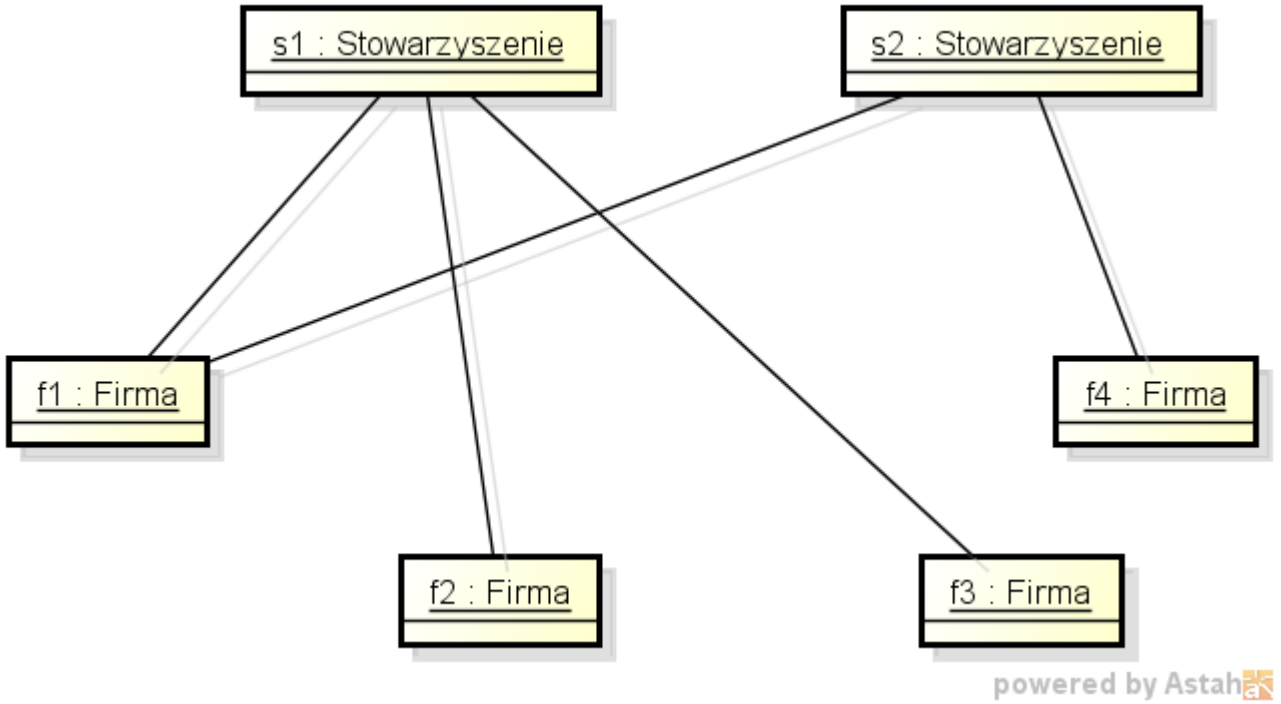
Agregacja

- Obiekty klasy składowej mogą należeć do (być częściami) wielu obiektów klasy złożonej
- Czas życia obiektów klasy składowej jest niezależny od czasu życia obiektów, do których należą



Agregacja


Stowarzyszenie
0..*
1..*
Firma
powered by astah*

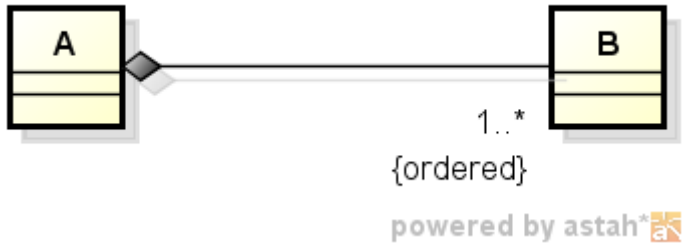

s1 : Stowarzyszenie
s2 : Stowarzyszenie
f1 : Firma
f4 : Firma
f2 : Firma
f3 : Firma
powered by Astah

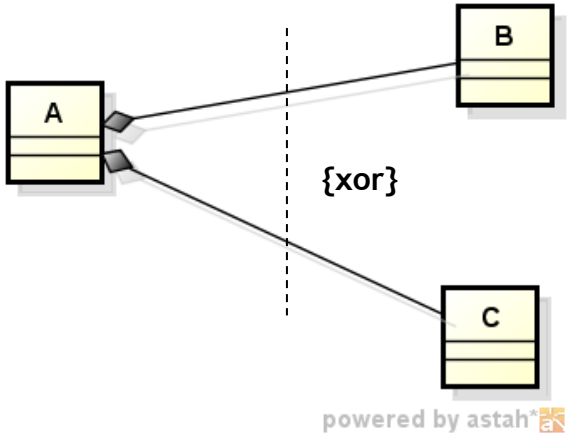
Własności asocjacji

Nawigowalność

Widzialność końców – jak dla atrybutów i operacji

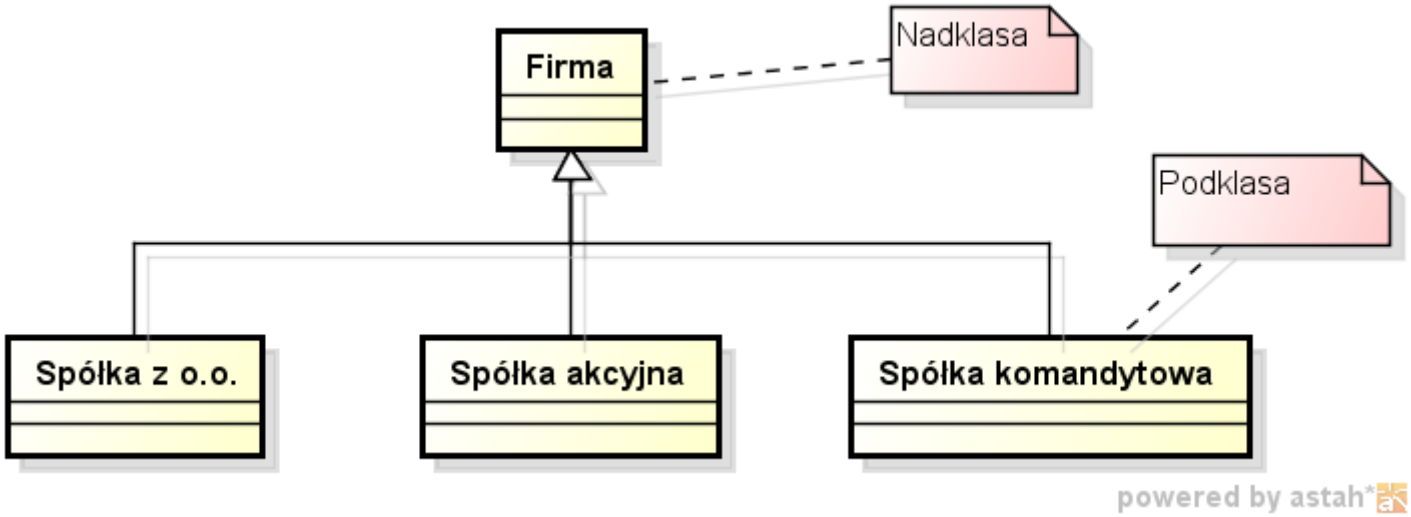
Ograniczenia





Generalizacja

- Składnia



Generalizacja – semantyka

- Relacja taksonomiczna (klasyfikująca) pomiędzy klasami (ogólnie klasyfikatorami), a nie pomiędzy obiektami klas (!)
  - przeciwzwrotna
  - przeciwsymetryczna
  - przechodnia

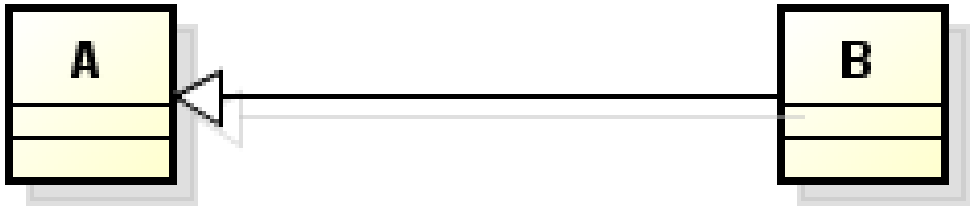
Generalizacja – semantyka

- Postulat zastępowalność – Barbara Liskov (MIT):

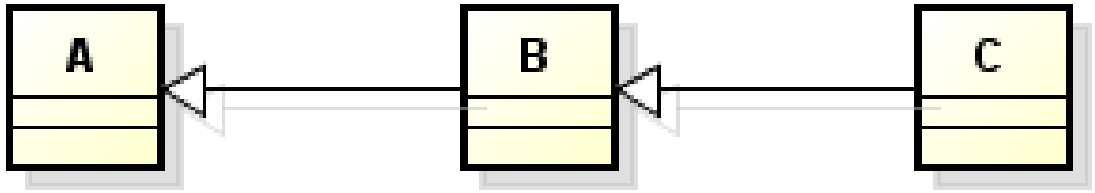
  Jeśli zmienna lub parametr ma deklarację v : X, to instancja każdego typu Y, która jest potomkiem (dziedzicem) X, może być przypisana v.

Generalizacja – semantyka

Jeżeli



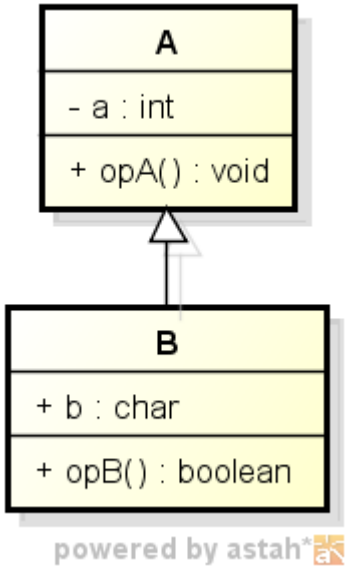
to obiekt klasy B nie jest obiektem typu A, ale jest obiektem rodzaju A.

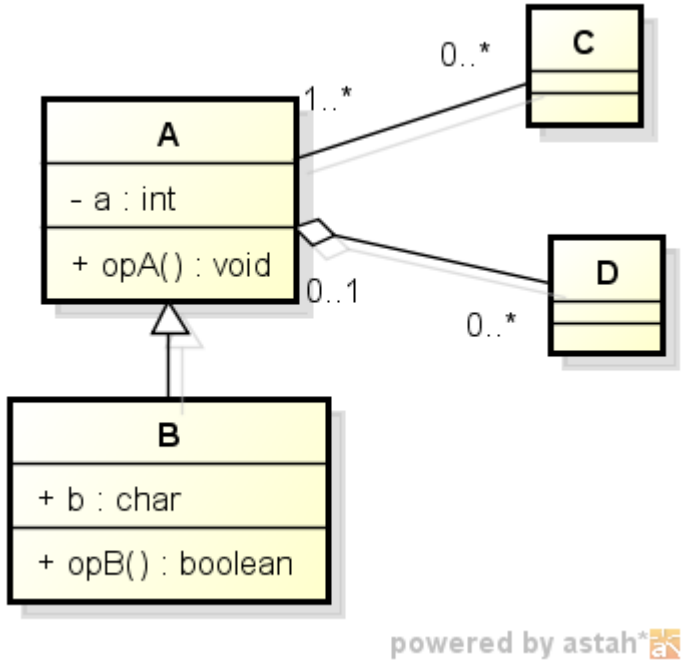


Obiekt klasy C jest też obiektem rodzaju B i rodzaju A.

Generalizacja – dziedziczenie

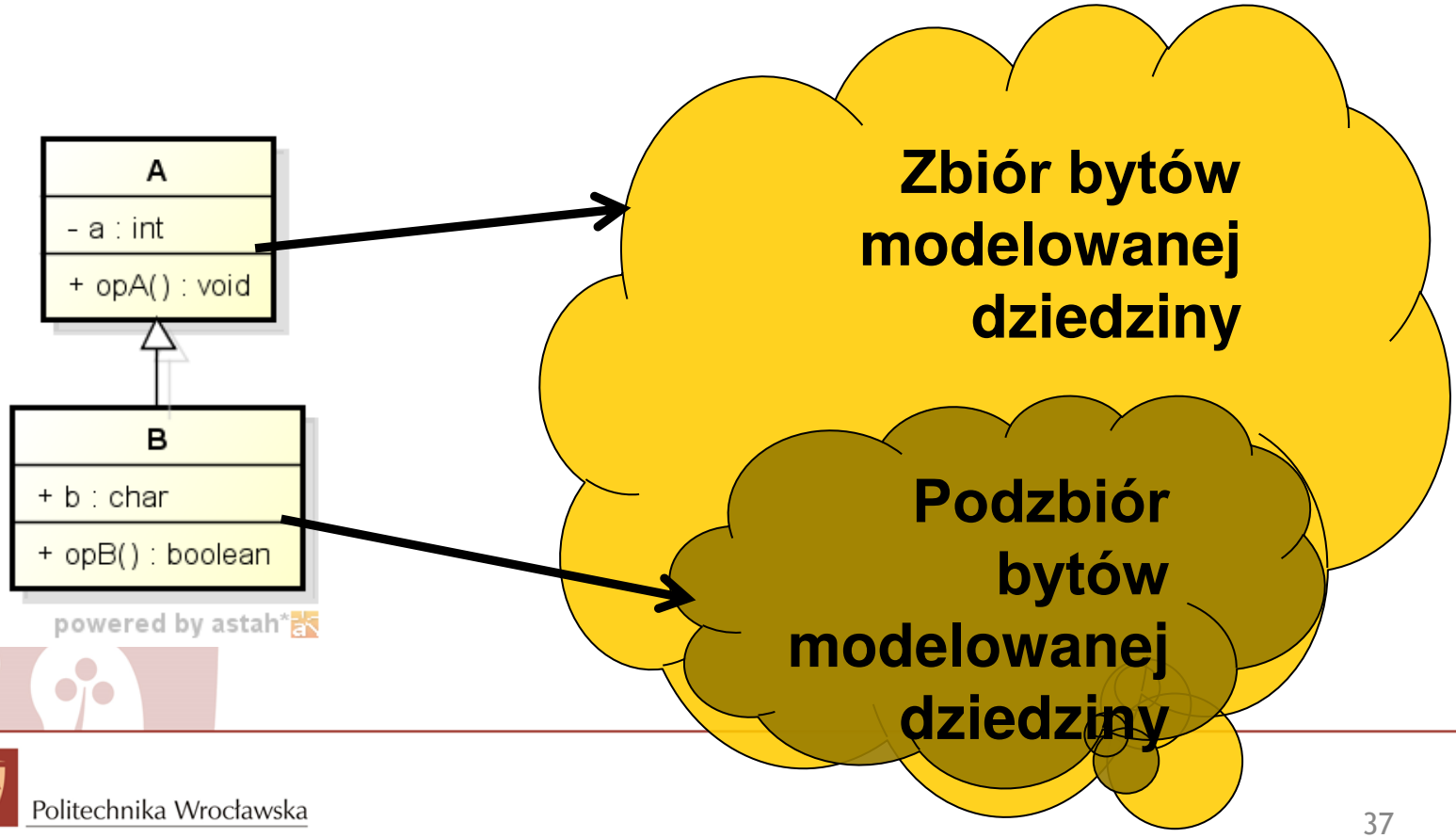
- Własności wewnętrzne i zewnętrzne





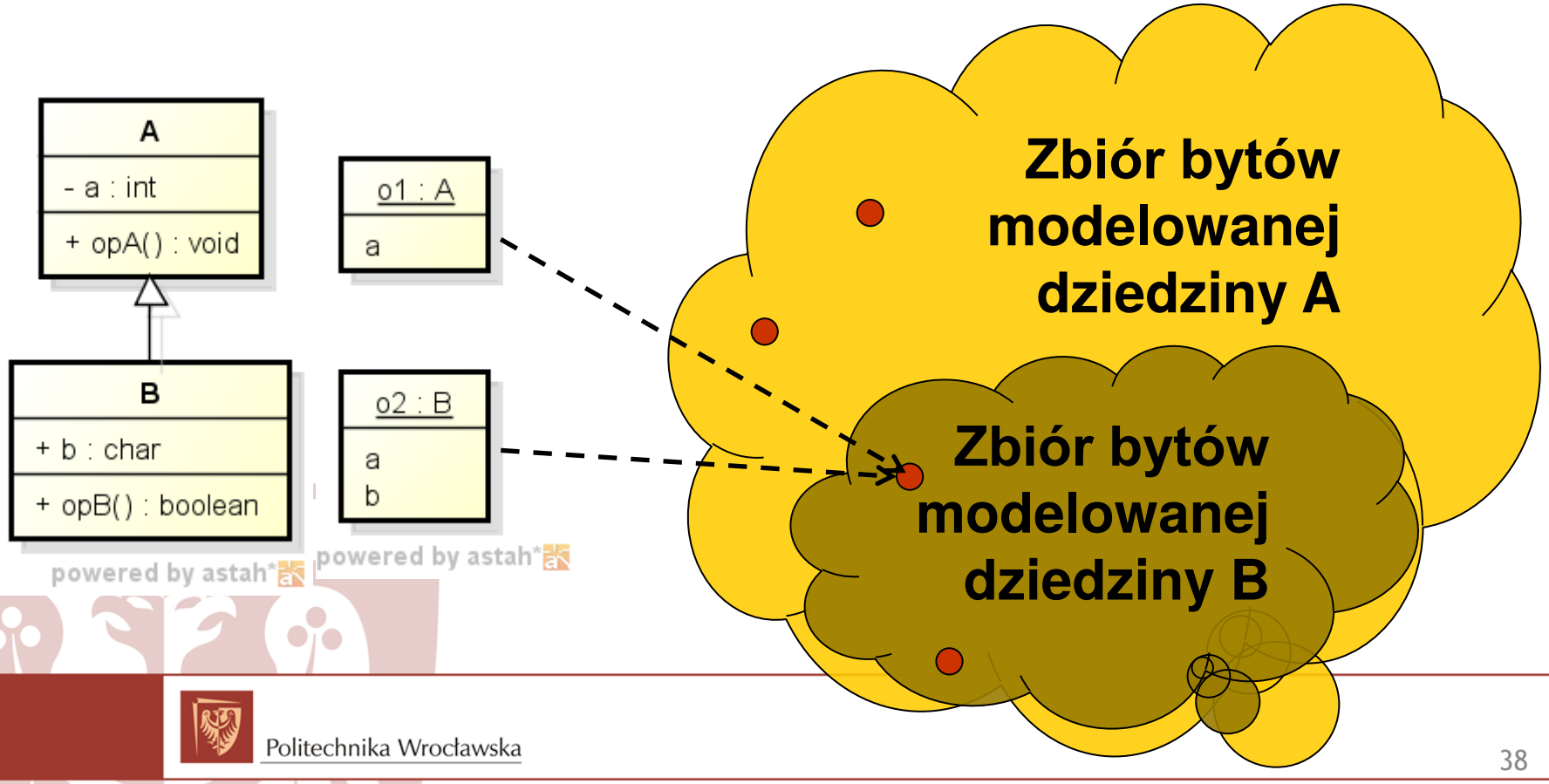
Generalizacja – zgodność dziedzinowa

- Zawieranie modelowanych dziedzin



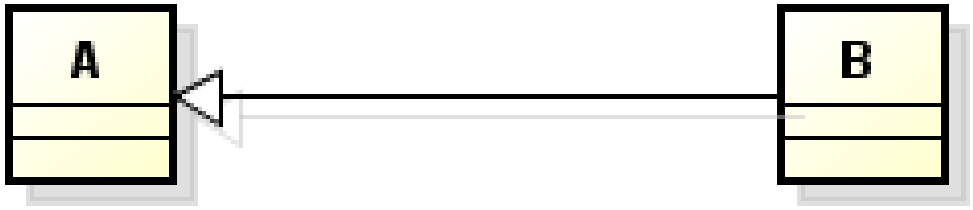
Generalizacja – poziom dziedzinowy

- Zgodność wartości wspólnych atrybutów



Generalizacja

Jeżeli



to

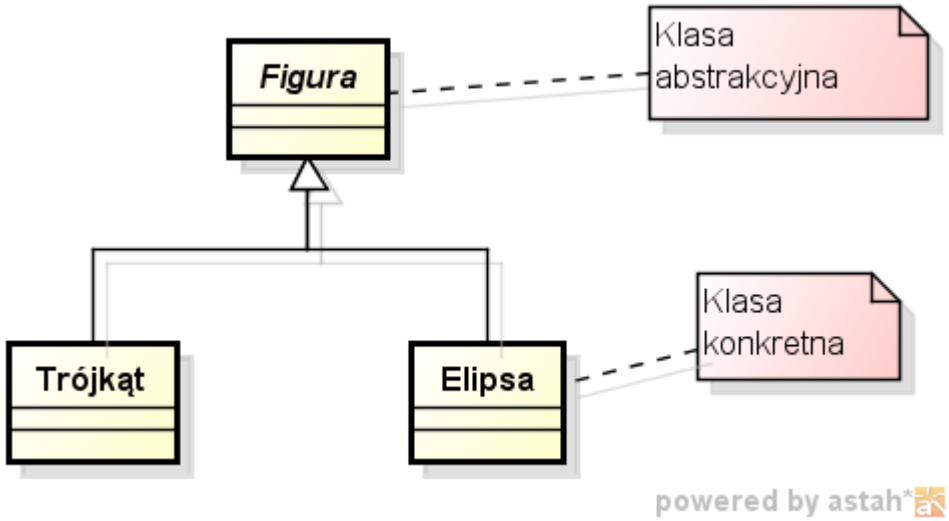
$$Extent(A) \cap Extent(B) = \varnothing \quad (!)$$

ale

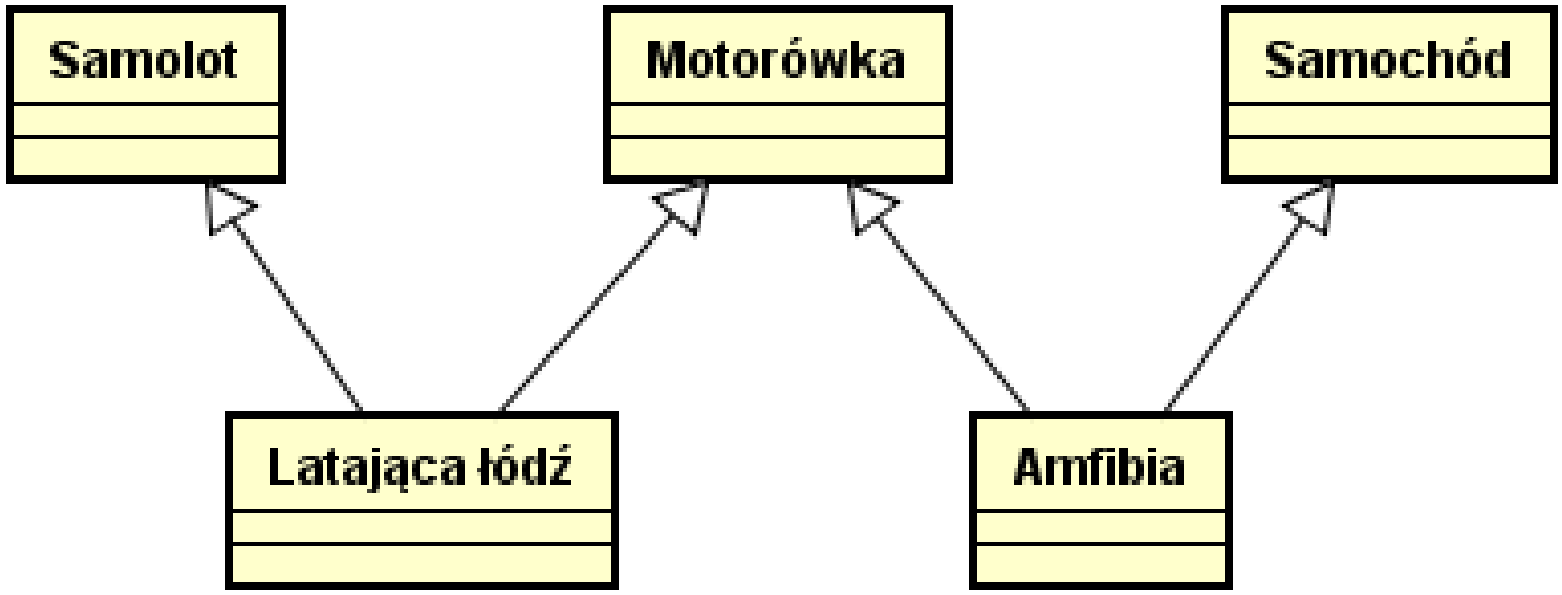
$$Sem(B) \subseteq Sem(A)$$

Generalizacja

Klasa abstrakcyjna – bezpośrednio nieinstantowalna

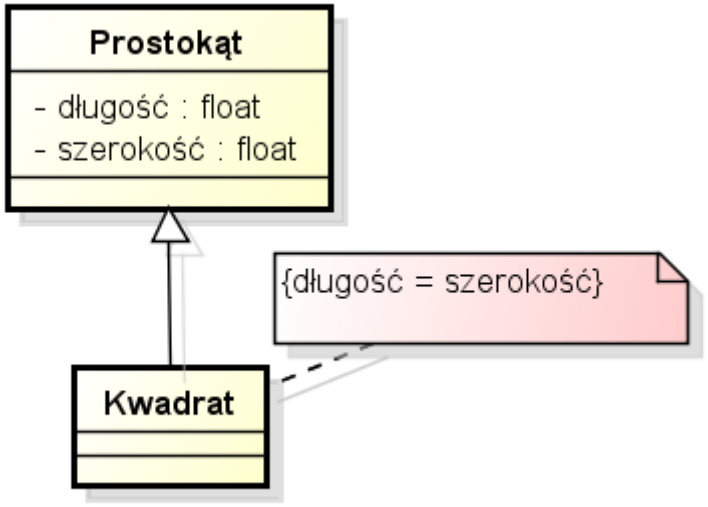


Generalizacja - wielodziedziczenie



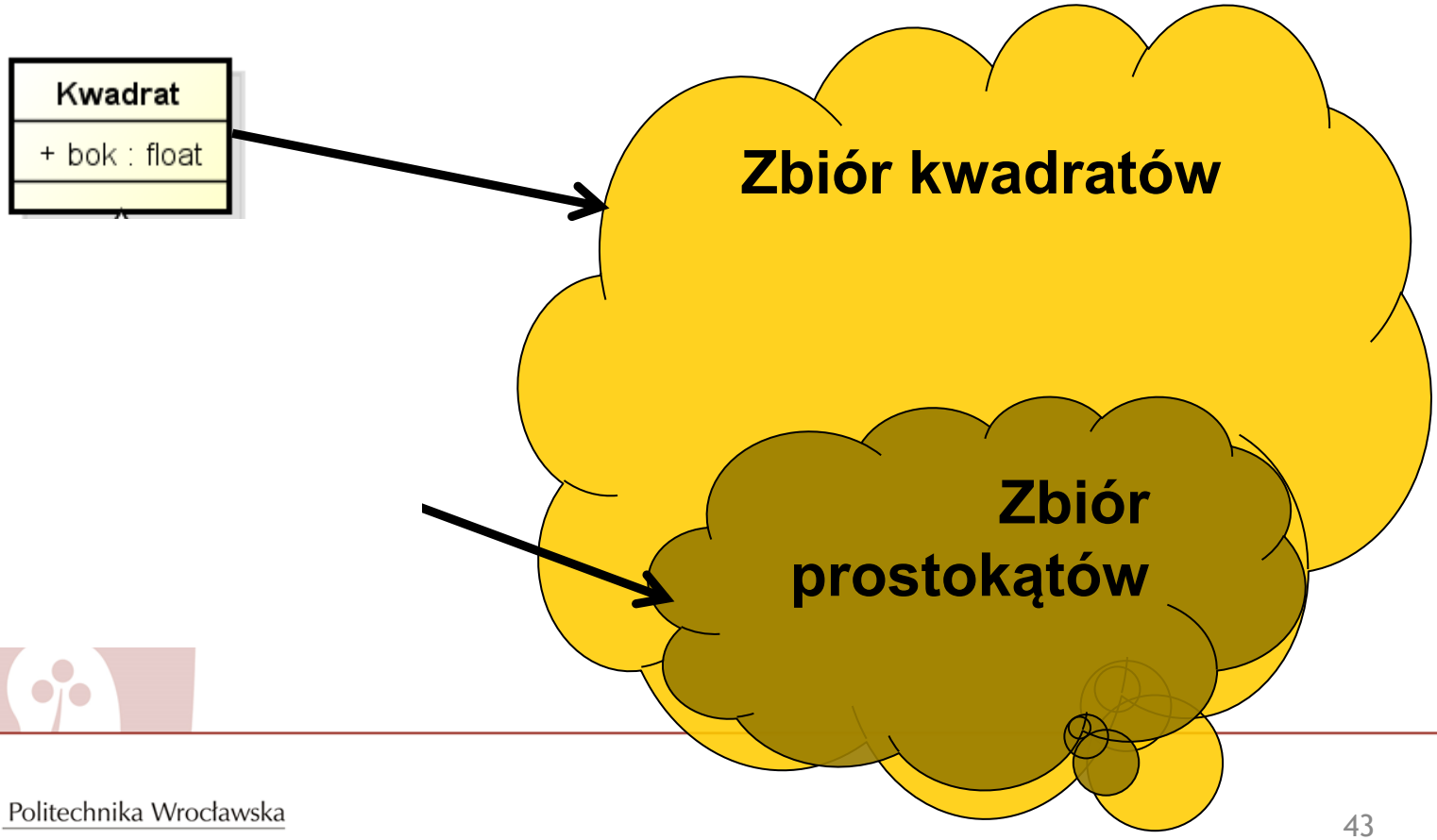
Definiowanie podklas

- Dodawanie nowych własności (atrybutów i operacji)
- Nakładanie ograniczeń na nadklasę



Definiowanie podklas

- Przykład bezmyślności



Obiekty – typy i rodzaje

Obiekt k1 : Kwadrat

jest **typu** Kwadrat

jednocześnie jest **rodzaju**:

- Prostokąt
- Czworokąt
- Figura

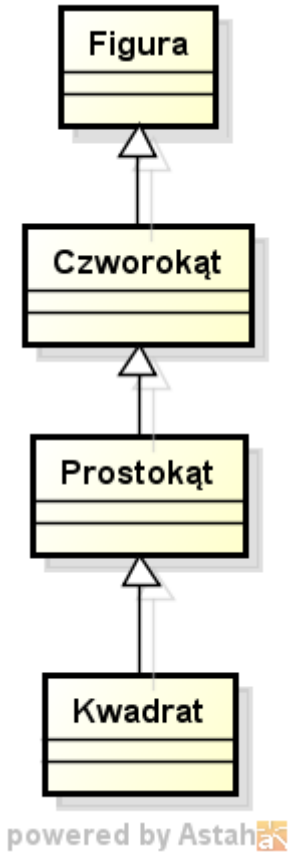


Diagram klas – podsumowanie

- Diagram klas wymaga informacji uzupełniającej w postaci:
  - definicji znaczenia (słownik pojęć)
    - każdej klasy
    - każdego jej atrybutu
    - każdej asocjacji
  - definicji ograniczeń dotyczących oddzielnie lub łącznie
    - klasy, atrybutu, asocjacji

Diagram klas – podsumowanie

- Diagram klas – obrazuje zbiorowości oraz związki w obrębie i pomiędzy zbiorowościami
- Diagram obiektów – instancja diagramu klas; obrazuje konkretny, skończony zbiór bytów (elementów zbiorowości) oraz powiązania pomiędzy tymi bytami w danej chwili

Diagram klas – podsumowanie

- Zalety:
  - diagram statyczny o dużej sile ekspresji; dogodny do specyfikacji modelu konceptualnego baz danych
  - bliski obiektowemu językowi programowania
  - mechanizm dziedziczenia wspomaga ponowne użycie i podnosi jakość
  - umożliwia definiowanie architektury (dekompozycja na elementy składowe - systemy, podsystemy)

Diagram klas – podsumowanie

- Wady i pułapki:
  - brak dynamiki; brak pojęcia czasu
  - nieodróżnianie poziomów abstrakcji; należy odróżniać diagramy klas stanowiące model wycinka rzeczywistości (model biznesowy) od diagramów modelujących projekt lub implementację systemu informatycznego

Diagramy klas - podsumowanie

Zasady konstruowania biznesowych diagramów klas

- Identyfikacja klas
- Identyfikacja związków pomiędzy klasami
  - generalizacja
  - asocjacje
- Identyfikacja atrybutów
  - zapewnienie rozróżnialności obiektów
  - atrybuty opcjonalne
- Identyfikacja ograniczeń

OCL – Object Constraint Language

- Składowa standardu UML
- Język specyfikacji warunków i ograniczeń (a nie programowania)
- Zmodyfikowana wersja klasycznego języka predykatów – zasadnicze modyfikacje:
  - typizacja wyrażeń
  - specyficzna notacja
  - konstrukcje (operacje) pseudo-programistyczne

OCL

Dwa obszary zastosowania:

- definiowane ograniczeń przy budowie modeli, np.
  - definiowanie niezmienników na diagramach klas
  - definiowanie warunków wstępnych i końcowych dla operacji (specyfikacja metod)
- definiowanie składni kontekstowej języka UML

OCL – self

- Wyrażenia są definiowane w kontekście instancji danego klasyfikatora, np.
  - obiektu w przypadku klasy,
  - powiązania w przypadku asocjacji, …
- self – słowo kluczowe jako referencja do danej instancji

OCL – wyrażenia

Nieformalnie – dwie postaci wyrażeń:

- wyrażenia ‘kropkowe’ (nawigacyjne), odpowiedniki termów, np.

  self.nazwisko
  self.dataUrodzenia

- wyrażenia ‘strzałkowe’, odpowiedniki formuł i termów, np.

  self.małżeństwo -> notEmpty()
  self.pracownicy -> size()

OCL – wyrażenia ‘kropkowe’

context Osoba:

self.nazwisko -- String

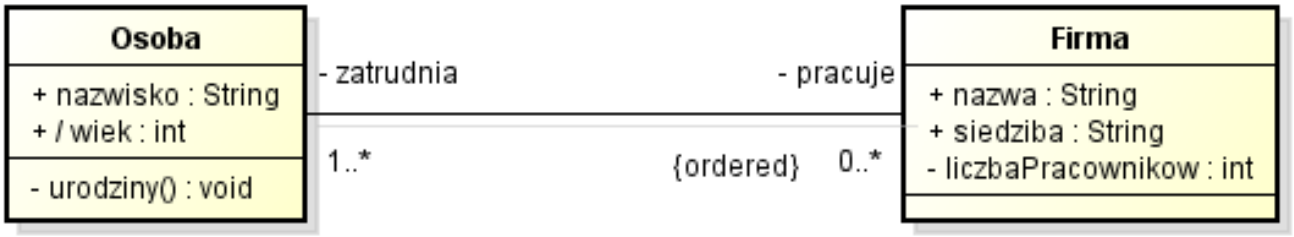
self.pracuje -- Firma

self.pracuje.nazwa -- String

OCL – wyrażenia ‘kropkowe’

context Osoba:

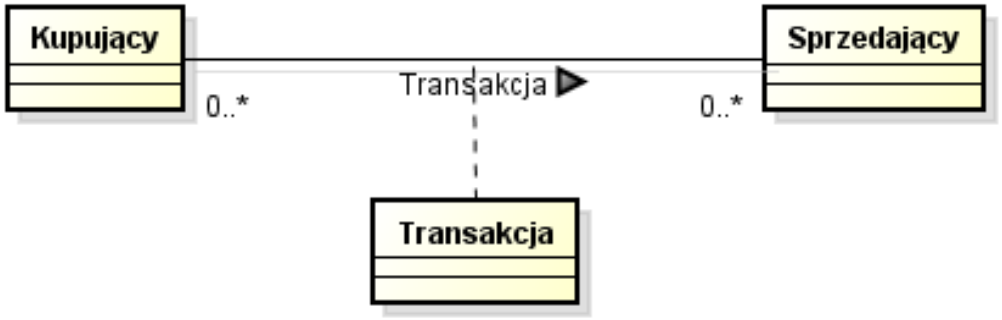
self.pracuje -- Set(Firma)



self.pracuje -- Sequence(Firma)

OCL – wyrażenia ‘kropkowe’

context Kupujący:

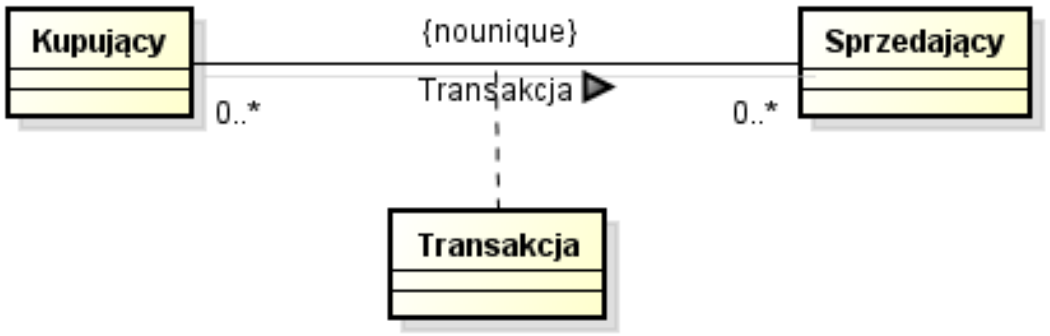


self.sprzedający -- Set(Sprzedający)

self.transakcja -- Set(Transakcja)

OCL – wyrażenia ‘kropkowe’

context Kupujący:



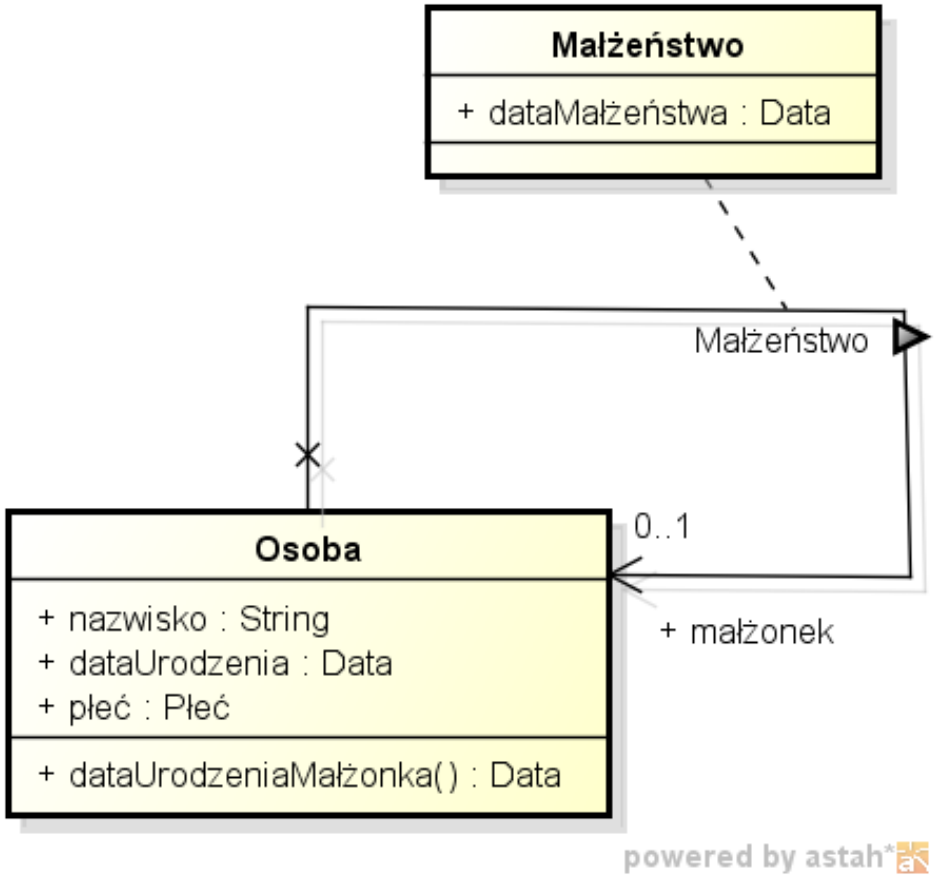
self.sprzedający -- Bag(Sprzedający)

self.transakcja -- Set(Transakcja)

OCL – schematy definiowania operacji

- **pre, post** warunek wstępny/końcowy
- **post** warunek końcowy
- **pre, body** warunek wstępny/treść
- **body** treść operacji

OCL – przykład



OCL – przykład definiowania operacji

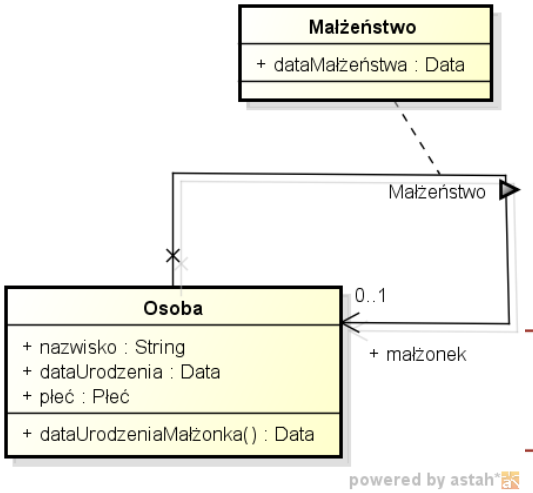
Przykład definicji operacji:

context Osoba ::

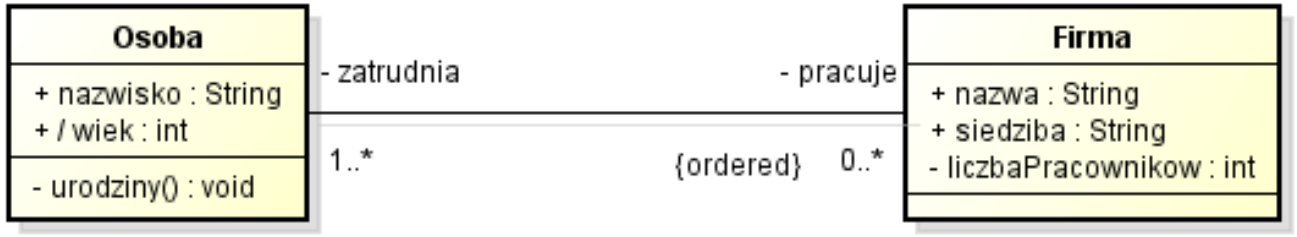
dataUrodzeniaMałżonka() : Data

pre self.małżeństwo -> notEmpty()

body self.małżonek.dataUrodzenia



OCL – przykład definiowania operacji



Przykład definicji operacji:

context Firma ::

zatrudnienieOsoby(o : Osoba) : void

post self.zatrudnia =

self.zatrudnia@pre -> including(o)

Modelowanie i analiza biznesowa

Prof. Zbigniew Huzar
Katedra Informatyki
Wydział Informatyki i Zarządzania

OCL – Object Constraint Language

- Składowa standardu UML
- Język specyfikacji warunków i ograniczeń (a nie programowania)
- Zmodyfikowana wersja klasycznego języka predykatów – zasadnicze modyfikacje:
  - typizacja wyrażeń
  - specyficzna notacja
  - konstrukcje (operacje) pseudo-programistyczne

OCL

Dwa obszary zastosowania:

- definiowane ograniczeń przy budowie modeli, np.
  - definiowanie niezmienników na diagramach klas
  - definiowanie warunków wstępnych i końcowych dla operacji (specyfikacja metod)
- definiowanie składni kontekstowej języka UML

OCL – składnia

- Słowa kluczowe

  and, attr, context, def, derive, else, endif,
  endpackage, if, implies, in, init, inv,
  let, not, oper, or, package,
  post, pre, then, xor

- Lokalne oznaczenia:

  let … in …

- Definicje

  def …

OCL – typy

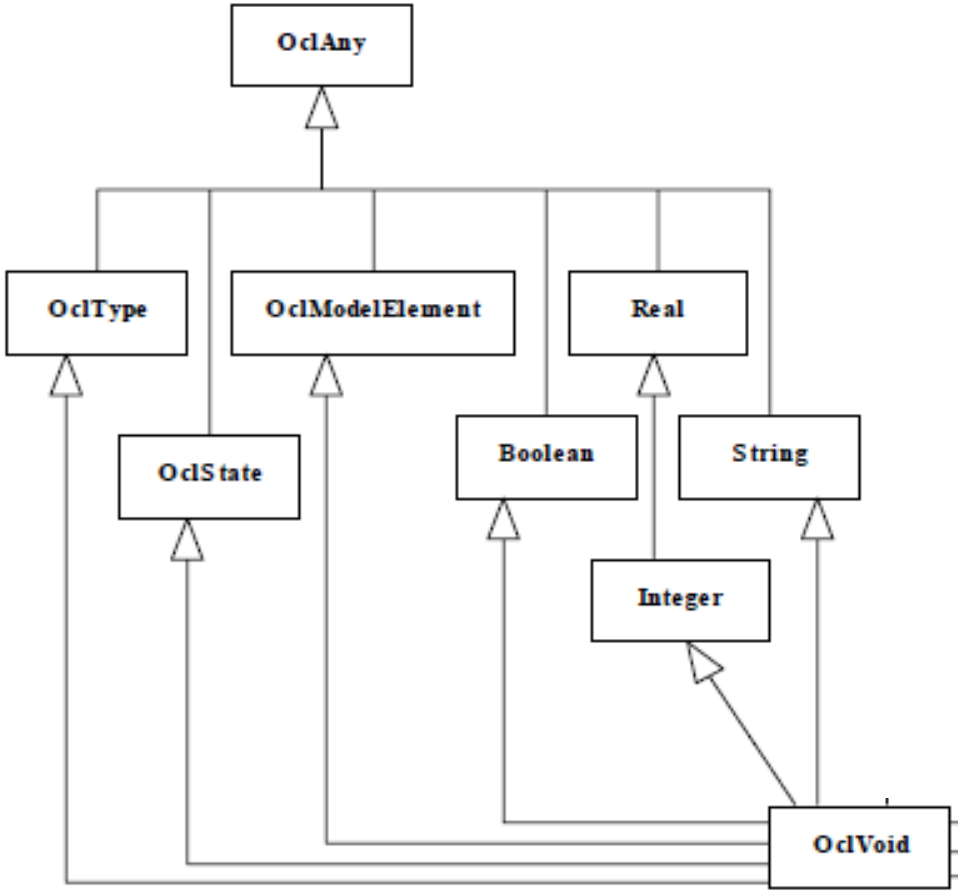
Typy proste

- Typy wbudowane
- Typy wyliczeniowe
- Klasyfikatory występujące w modelach stanowią typy modelu (OclType)
- …

Typy złożone

- Krotki (rekordy)
- Kolekcje (wielozbiory, zbiory, sekwencje)

OCL - typy

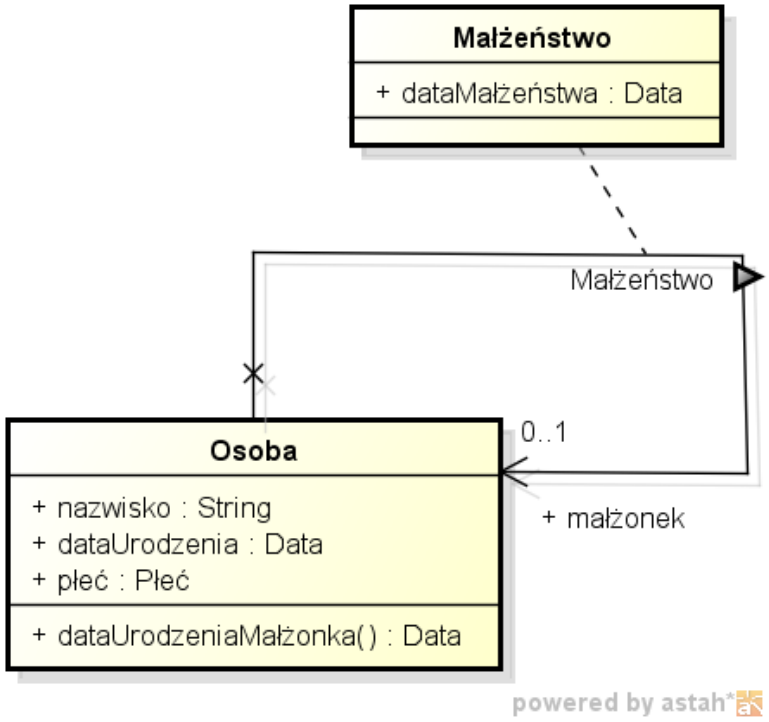

OclAny
OclType
OclModelElement
Real
OclState
Boolean
String
Integer
OclVoid

OCL – typy bazowe

- Typ danych = zbiór wartości + zbiór operacji
- Typy bazowe (wbudowane)
  - Boolean
  - Integer (pełny zakres)
  - Real (pełny zakres)
  - String

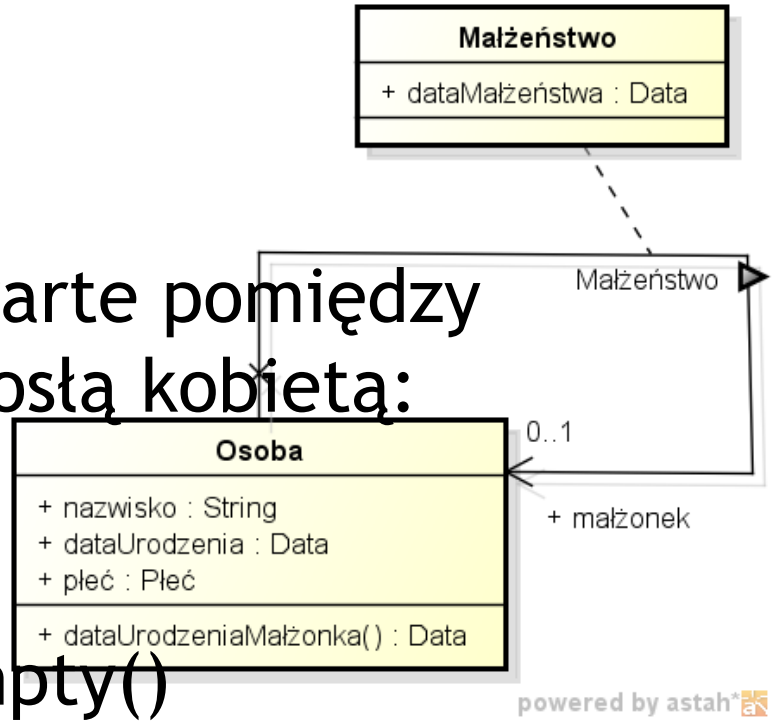
Przykład ograniczeń

- Małżeństwo może być zawarte pomiędzy dorosłym mężczyzną i dorosłą kobietą



Przykład ograniczeń

Małżeństwo może być zawarte pomiędzy dorosłym mężczyzną i dorosłą kobietą:



context Osoba **inv:**

self.małżonek -> notEmpty()

implies self.płeć <> self.małżonek.płeć

and (self.małżeństwo.data –
self.rokUrodzenia) > 18

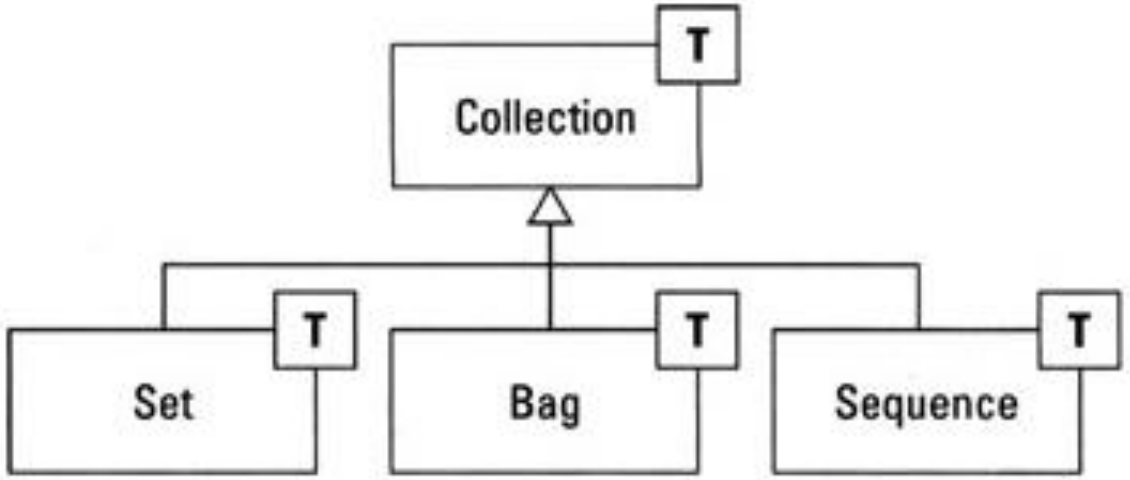
and (self.małżeństwo.data –
self.małżonek.rokUrodzenia) > 18

OCL – typy kolekcyjne

Collection – typ abstrakcyjny parametryzowany dowolnym innym typem

Podtypy

- *Set*
- *Bag*
- *Sequence*



OCL – typy kolekcyjne

- Literały – reprezentacja wartości danego typu
  - Set{1, 5, 3}

    Set{czerwony, biały, żołty}
  - Bag{1, 5, 5, 3}

    Bag{1, 5, 3}
  - Sequence{1, 5, 2, 8}

    Sequence{1..10}

OCL – typ rekordowy

Literał

Tuple{nazwisko: String = ‘Jan’, wiek: Integer = 10}

jest równoważny literałom:

Tuple{nazwisko = ‘Jan’, wiek = 10}

Tuple{wiek = 10, nazwisko = ‘Jan’}

OCL – inne typy

- OclType – typ wyliczeniowy; każdemu elementowi-klasyfikatorowi w modelu UML odpowiada literał wyliczeniowy.
- Dla modelu z poprzedniego diagramu klas:

  Osoba, *Firma*, *Zatrudnienie*, *Zwolnienie*, …

  są wartościami typu OclType

OCL – zgodność typów

- Postulat Liskov

  Niech Φ (x) będzie dowiedzioną własnością o obiektach x typu T. Wówczas własność Φ (y) musi być prawdziwa dla obiektów y typu S, gdzie S jest podtypem typu T.

- Typ A jest zgodny z typem B (A jest podtypem B) jeśli instancja typu A może być wstawiona w dowolne miejsce, w którym jest spodziewana instancja typu B.

OCL – zgodność typów

- Zasady zgodności:
  - Każdy typ jest zgodny z każdym swoim nadtypem; zgodność typów jest przechodnia.
  - Typ Collection(Typ1) jest zgodny z typem Collection(Typ2), jeśli Typ1 jest zgodny z Typ2.
  - Podobnie Set(Typ1)/Set(Typ2), Sequence(Typ1)/Sequence(Typ2), Bag(Typ1)/Bag(Typ2)

OCL – wybrane często stosowane operacje

Operacja	Zwracana wartość
size()	Liczba elementów kolekcji
includes(*obiekt*)	True, gdy *obiekt* należy do kolekcji, false w p.p.
union(*kolekcja*)	Suma mnogościowa dwóch kolekcji
intersection(*kolekcja*)	Przekrój mnogościowy dwóch kolekcji
asSet(*kolekcja*)	Zbiór zawierający wszystkie elementy *kolekcji*
notEmpty()	True, gdy kolekcja jest niepusta, false w p.p.
sum()	Suma wartości elementów kolekcji dla której jest określone dodawanie

OCL – operacje na typach kolektywnych

Operacje podstawowe:

- select
- collect
- forAll
- exists
- iterate

OCL – operacja select

- Składnia

  collection ->

  select(*v* | *boolean-expression-with-v*)

  – *collection* – dowolne wyrażenie OCL typu kolekcyjnego
  – *v* – zmienna (iterator) reprezentuje referencje do obiektów z kolekcji
  – *boolean-expression-with-v* – wyrażenie określające warunek przynależności do podkolekcji
- Semantyka – definicja podkolekcji

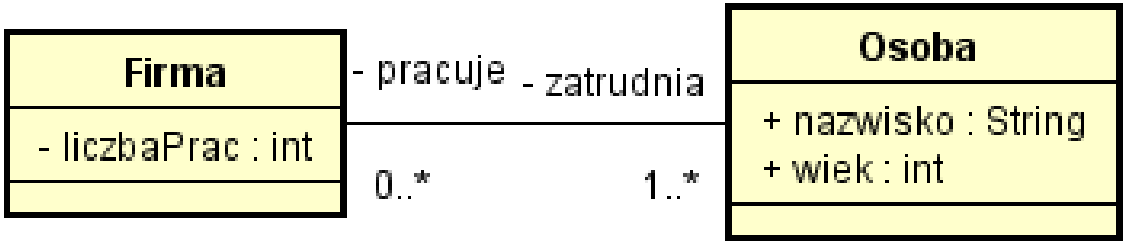
OCL – operacja select

- Przykład

context Firma

self.zatrudnia ->

select(o | o.wiek > 30)



OCL – operacja collect

- Składnia

  collection ->

  collect(v | *expression-with-v*)
- Semantyka – definicja nowej kolekcji na podstawie danej kolekcji

OCL – operacja collect

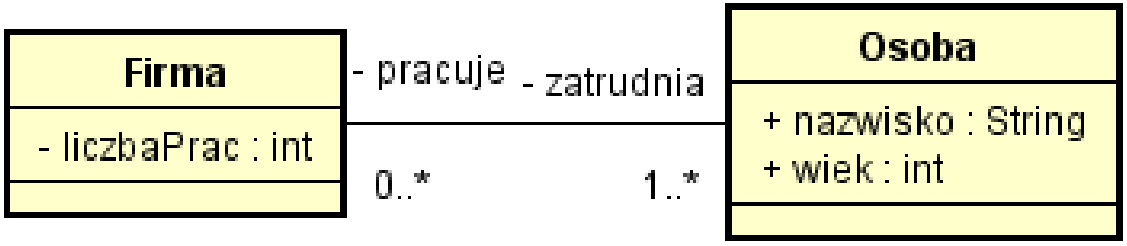
- Przykład

context Firma

self. zatrudnia ->

collect(o | o.wiek)

-- Bag(int)



OCL – operacja forAll

- Składnia

  collection ->

  forAll(v | *boolean-expression-with-v*)

- Semantyka – odpowiednik tradycyjnej notacji:

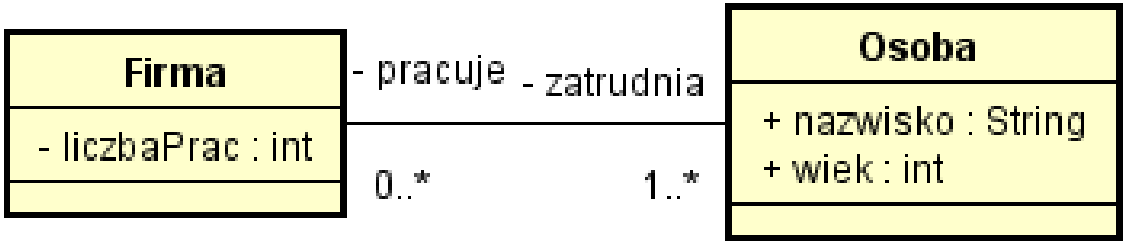
  $\forall v \in collection \bullet boolean\text{-}expression\text{-}with\text{-}v$

OCL – operacja forAll

- Przykład

  context Firma

  self.zatrudnia -> forAll(o | o.wiek > 17)



OCL – operacja exists

- Składnia

  collection ->

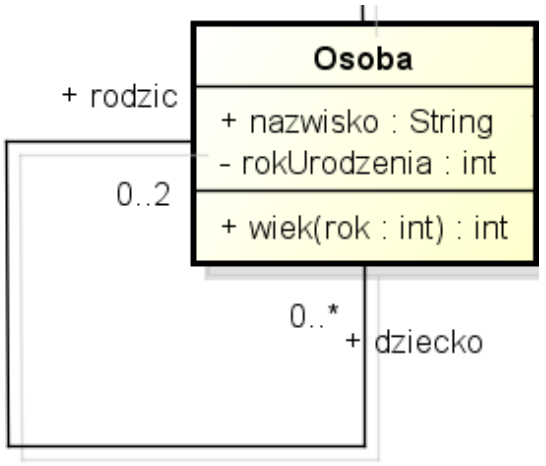
  exists(v | *boolean-expression-with-v*)

- Semantyka – odpowiednik tradycyjnej notacji:

  $\exists v \in collection \bullet boolean\text{-}expression\text{-}with\text{-}v$

Przykład

Dzieci są młodsze od rodziców



context Osoba **inv:**

self.dziecko -> notEmpty()

implies self.dziecko ->
forAll(c|c.rokUrodzenia >
self.rokUrodzenia)

OCL – operacja iterate

- Składnia

  collection->

  iterate(*elem* : *Type-1*;

  acc : *Type-2* = *expression* |

  expression-with-elem-and-acc)

  – zmienna *elem* jest iteratorem typu jak wyrażenie *collection*,
  – zmienna *acc* jest akumulatorem,
  – *expression* jest początkową wartością akumulatora,
  – *expression-with-elem-and-acc* wyrażeniem typu *Type-2*, która modyfikuje bieżącą wartość akumulatora

- Semantyka – wyznaczenie pewnej wartości typu *Type-2*

OCL – operacja iterate

- Semantyka operacyjna
  1. Akumulator *acc* otrzymuje wartość początkową określoną przez *expression*
  2. Iterator *elem* przegląda wszystkie elementy kolekcji, i dla każdego elementu oblicza
     expression-with-elem-and-acc
  3. Po każdym obliczeniu *expression-with-elem-and-acc*, jego wartość jest przypisywana do *acc*
  4. Wynikiem operacji jest końcowa wartość *acc*

OCL – operacja iterate

- Przykłady

  Bag{2, 3, 1, 2, 5, 5} ->

  iterate(e : Integer; a : Integer = 0 | a + e)

  – Co jest wynikiem obliczenia?

  Set{1, 2} ->

  iterate(e : Integer; a : Integer =0 | a*a + e)

  – Co teraz jest wynikiem obliczenia?

OCL - inne operacje

Typ	Operator
Collection	size() : Integer, includes(object : T) : Boolean , excludes(object : T) : Boolean, count(object : T) : Integer, includesAll(c2 : Collection(T)) : Boolean, excludesAll(c2 : Collection(T)) : Boolean, isEmpty() : Boolean, notEmpty() : Boolean, sum() : T, product(c2: Collection(T2)) : Set(Tuple(first: T, second: T2))

OCL - inne operacje

Typ	Operator
Set	union(s : Set(T)) : Set(T), union(bag : Bag(T)) : Bag(T), intersection(s : Set(T)) : Set(T), intersection(bag : Bag(T)) : Set(T), including(object : T) : Set(T), excluding(object : T) : Set(T), flatten() : Set(T2), asSet() : Set(T), asOrderedSet() : OrderedSet(T), asSequence() : Sequence(T), asBag() : Bag(T)

OCL - inne operacje

Typ	Operator
Bag	union(s : Set(T)) : Bag(T), union(bag : Bag(T)) : Bag(T), intersection(s : Set(T)) : Set(T), intersection(bag : Bag(T)) : Bag(T), including(object : T) : Bag(T), excluding(object : T) : Bag(T), count(object : T) : Integer, flatten() : Bag(T2), asSet() : Set(T), asOrderedSet() : OrderedSet(T), asSequence() : Sequence(T), asBag() : Bag(T)

OCL - inne operacje

Typ	Operator
Sequence	count(object : T) : Integer, union(s : Sequence(T)) : Sequence(T), flatten() : Sequence(T2), insertAt(index : Integer, object : T) : Sequence(T), subSequence(lower : Integer, upper : Integer) : Sequence(T), first() : T, last() : T including(object : T) : Sequence(T), asSet() : Set(T), asSequence() : Sequence(T), asBag() : Bag(T)

OCL – przykład funkcji pomocniczej

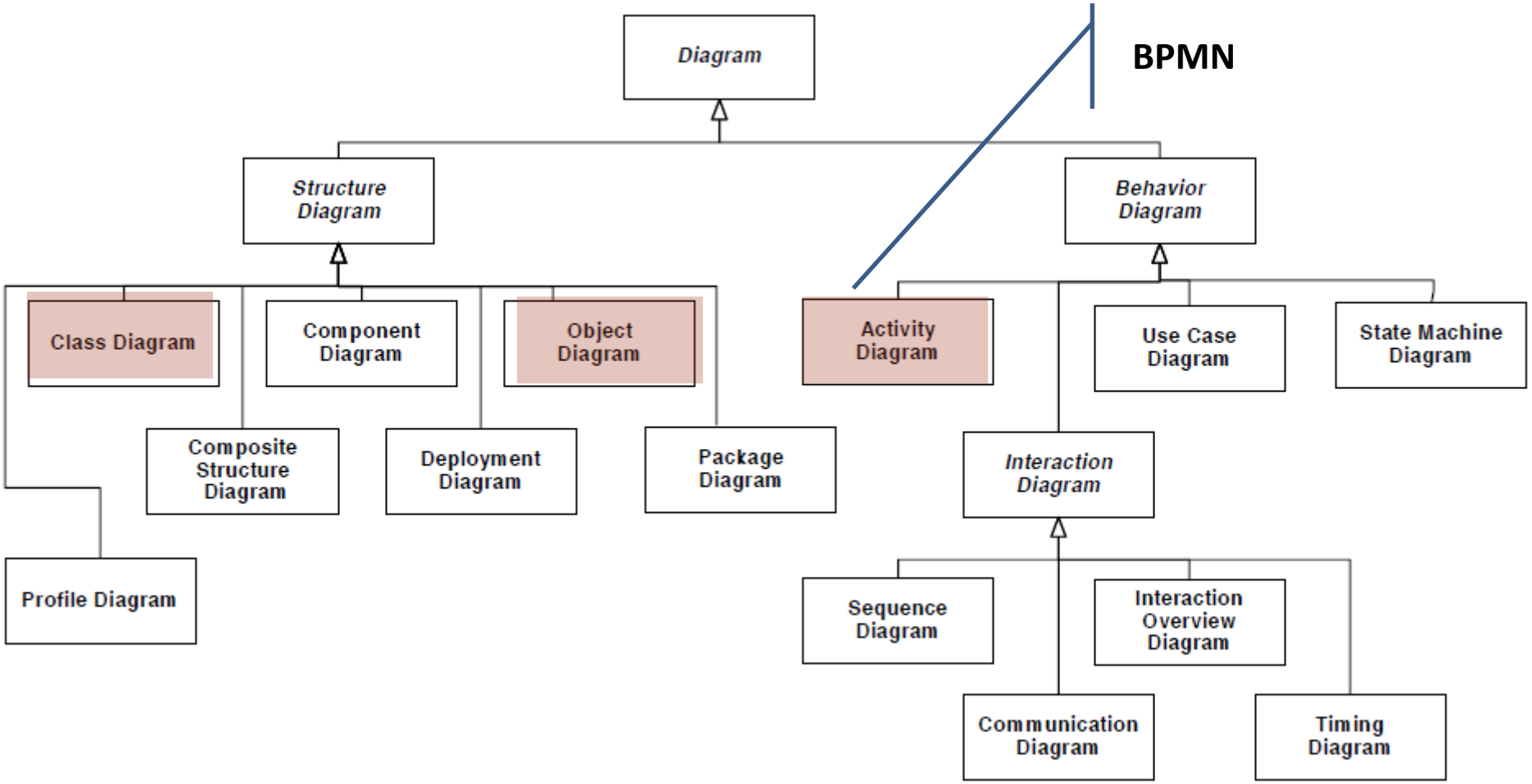
- Jednostka nie może bezpośrednio lub pośrednio być swoją podjednostką

OCL – przykład definicji rekursywnej

```
context Jednostka inv:
        not self.składowe() -> includes(self)


def
Jednostka::składowe() : Set(Jednostka)
body
  self.podjednostka ->
        union(self.podjednostka ->
              collect(j | j.składowe())
```

UML – diagramy



Business Process Model and Notation

- Object Management Group
  - 2008, wersja BPMN 1.0
  - 2011, wersja BPMN 2.0
- Standard modelowania procesów biznesowych (usług biznesowych)

Business Process Model and Notation

- Usługa biznesowa stanowi zestaw konsumowalnych, niematerialnych korzyści:
  - dostarczana przez odpowiedzialnego dostawcę na zlecenie autoryzowanego konsumenta
  - realizowana w wyniku aktywności różnych systemów technicznych
  - wykorzystana i konsumowana przez zlecającego konsumenta.

Business Process Model and Notation

- Ukierunkowany na analityków biznesowych, architektów systemowych i twórców oprogramowania
- Jeden rodzaj diagramów BPD (Business Process Diagram) ze szczególnymi podrodzajami
  - Diagramy kolaboracji (diagram konwersacji)
  - Diagramy standardowych procesów (diagramy orkiestracji)
  - Diagramy choreografii

Modelowanie w BPMN

- Model BPMN jako zbiór diagramów procesów jest:
  - konieczny,
  - ale niewystarczający

  do definiowania wymagań przyszłego systemu informatycznego!
- Diagram procesów BPMN to odpowiednik diagramu biznesowych przypadków użycia UML

Modelowanie w BPMN

- Metafora modelowania
  - Struktura organizacyjna firmy/organizacji
  - Cele biznesowe przypisane jednostkom organizacyjnym
  - Zasoby przeznaczone do realizacji celów
  - Sposób opisu realizacji celów oparty o opis procesów:
    - Deskryptywny – reguły biznesowe
    - Operacyjny – proceduralny
    - Mieszany – przepływ prac (workflow)

Modelowanie w BPMN

- Opis procesu BPMN
  - zbiór zadań (elementarnych aktywności)
  - częściowy porządek w zbiorze zadań

    Częściowy porządek w sensie ścisłym – relacja antysymetryczna i przechodnia
    - Przepływ sterowania
    - Przepływ danych

Modelowanie w BPMN

- Proces biznesowy – przepływ czynności w organizacji celem wykonania określonej pracy (usługi)
- Uczestnicy procesów biznesowych – rodzaje procesów:
  - Prywatne (firma)
  - Publiczne (otoczenie firmy)
  - Współpracy/kolaboracji (colaboration)

Modelowanie w BPMN

- Procesy biznesowe można:
  - grupować w pule reprezentujące uczestników; wyznaczać granice dla przepływu sterowania
  - dekomponować na tory obrazujące obszary odpowiedzialności (akcje na różnych torach tego samego procesu może łączyć przepływ sterowania)
  - łączyć w grupy współdziałania – kolaboracje

Modelowanie w BPMN

- Uczestnicy, kolaboracje

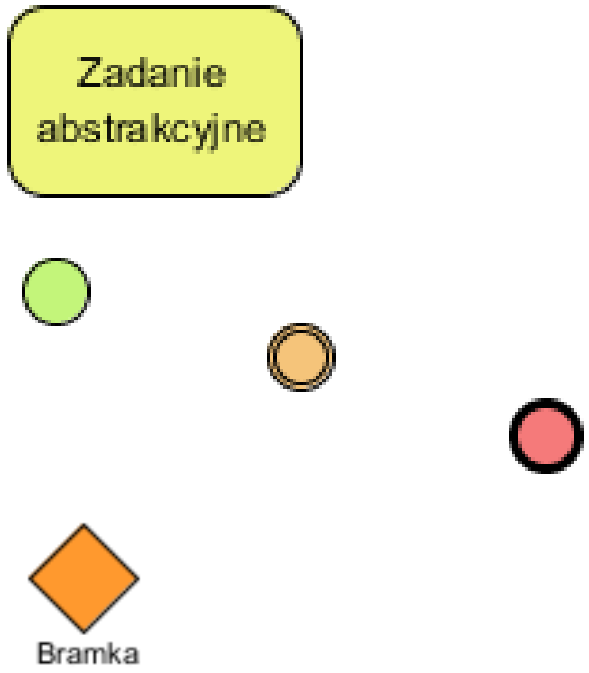
Diagram procesów - składnia

Etykietowany graf skierowany:

- Wierzchołki:
  - Aktywności (zadania, podprocesy)
  - Zdarzenia
  - Bramki
- Łuki:
  - Przepływy sterowania
  - Przepływy danych

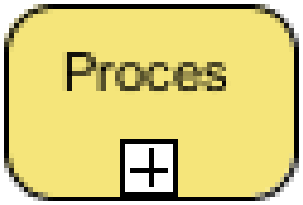
Elementy składowe modelu procesu

- Zadania
- Zdarzenia
  - Początkowe
  - Pośrednie
  - Końcowe
- Bramki
- Przepływy
  - Sterowania
  - Danych



Elementy składowe procesu

- Proces
  - Zagnieżdża zadania
  - Zagnieżdża podprocesy



- Wywołanie procesu

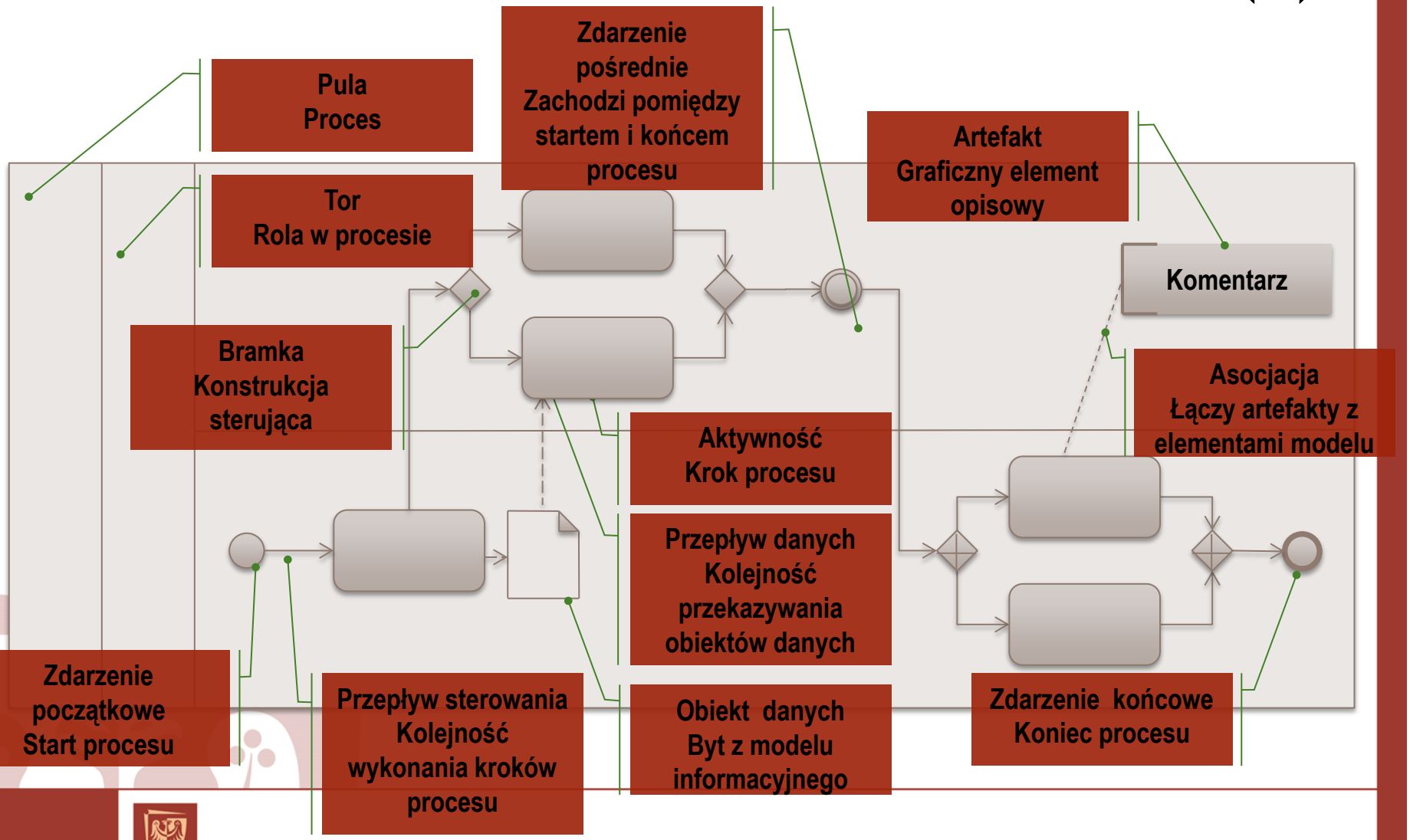
Modelowanie w BPMN

- Procesy składają się z:
  - podprocesów
  - zadań (akcji)
- Mogą być przedstawiane na tle obszarów odpowiedzialności (torów)

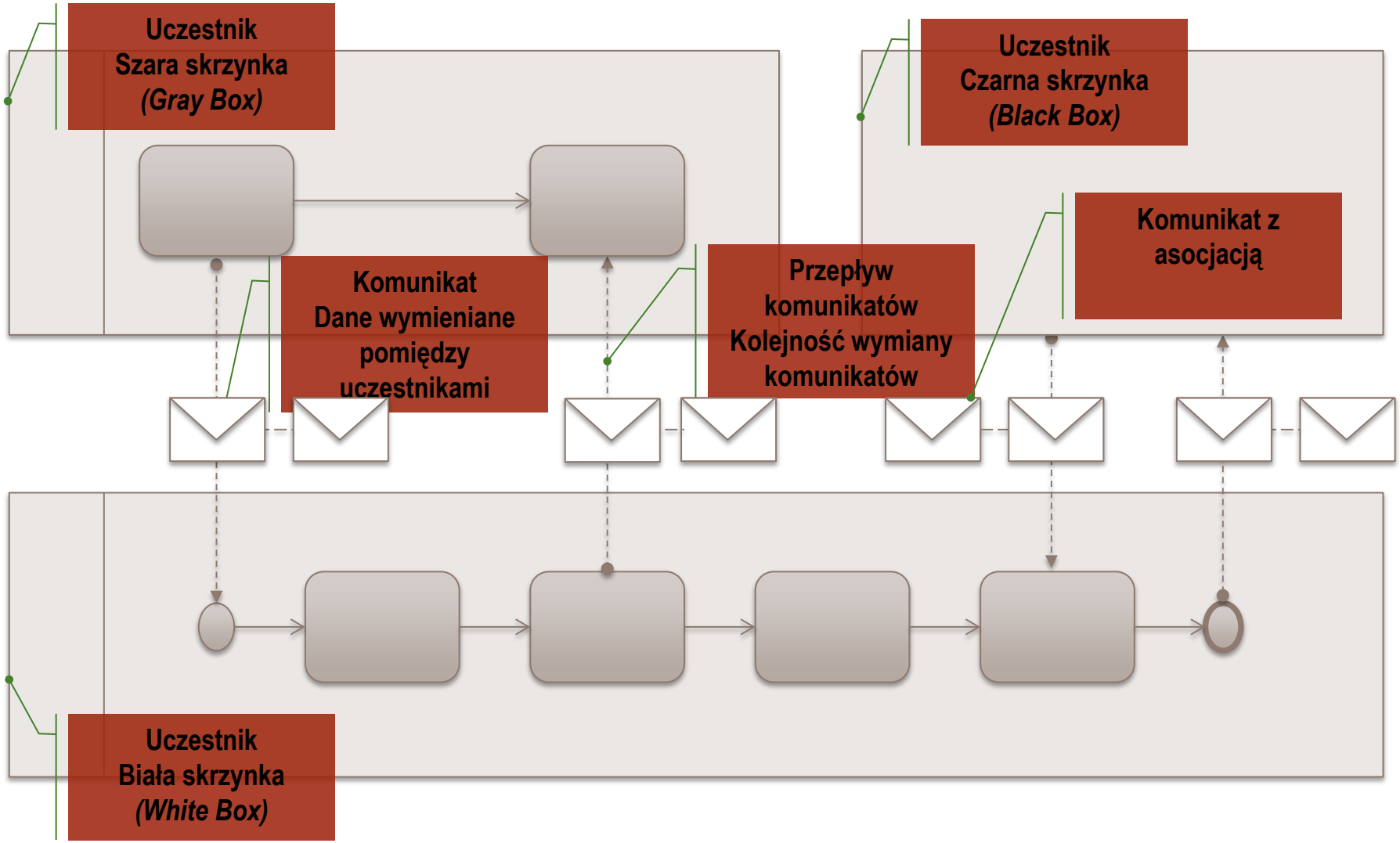
Modelowanie w BPMN

- Wykonanie procesu
  - Proces jest inicjowany zajściem zdarzenia rozpoczynającego
  - Rozpoczęcie realizacji zadania jest uwarunkowane przepływem sterowania i przepływem danych na wejściu, albo zajściem odpowiednich zdarzeń
  - Wykonanie zadania określa przepływ sterowania i przepływ danych na wyjściu,
  - Zakończenie procesu wyznacza zajście zdarzenia końcowego

Modelowanie w BPMN – składnia (1)



Modelowanie w BPMN - składnia (2)



Modelowanie w BPMN - semantyka

Opis usługi

- Ciąg interakcji z otoczeniem
- Ciąg interakcji wewnętrznych

Business Process Model and Notation

- Ukierunkowany na analityków biznesowych, architektów systemowych i twórców oprogramowania
- Jeden rodzaj diagramów BPD (Business Process Diagram) ze szczególnymi podrodzajami
  - Diagramy kolaboracji (diagram konwersacji)
  - Diagramy standardowych procesów (diagramy orkiestracji)
  - Diagramy choreografii

Modelowanie w BPMN

- Opis procesu BPMN
  - zbiór zadań (elementarnych aktywności)
  - częściowy porządek w zbiorze zadań

    Częściowy porządek w sensie ścisłym – relacja antysymetryczna i przechodnia
    - Przepływ sterowania
    - Przepływ danych

Modelowanie w BPMN

- Proces biznesowy – przepływ czynności w organizacji celem wykonania określonej pracy (usługi)
- Uczestnicy procesów biznesowych – rodzaje procesów:
  - Prywatne (firma)
  - Publiczne (otoczenie firmy)
  - Współpracy/kolaboracji (colaboration)

Diagram procesów - składnia

Etykietowany graf skierowany:

- Wierzchołki:
  - Aktywności (zadania, podprocesy)
  - Zdarzenia
  - Bramki
- Łuki:
  - Przepływy sterowania
  - Przepływy danych

Zadania

Serwisowe *(Serwise Task)*
Wywołuje zautomatyzowaną aplikację, np usługę Web Service.

Wysłania komunikatu *(Send Task)*
Wysyła komunikat i kończy się.

Odbioru komunikatu *(Receive Task)*
Oczekuje na komunikat, po odebraniu komunikatu kończy się.

Użytkownika *(User Task)*
Wykonywane przez użytkownika z wykorzystaniem systemów IT.

Manualne *(Manual Task)*
Wykonywana przez użytkownika manualnie, bez wykorzystania systemów IT.

Reguła biznesowa *(Business Rule Task)*
Wykonywane przez silnik reguł biznesowych.

Skryptowe *(Script Task)*
Wykonywane przez silnik procesów biznesowych.

BPMN – OMG

„Podstawowym celem języka BPMN jest dostarczenie notacji, która jest łatwo zrozumiała dla wszystkich uczestników projektu dotyczącego procesów biznesowych, od analityków biznesowych, którzy tworzą wstępne projekty procesów, do programistów odpowiedzialnych za wdrożenie technologii, na platformie na której będą wdrożone te procesy, a również dla ludzi biznesu, którzy będą zarządzać i monitorować wdrożone procesy. Tak więc, BPMN umożliwia wypełnienie luki pomiędzy projektowaniem procesów biznesowych, a ich wdrożeniem”.

BPMN – podsumowanie

- Zalety
  - Wspólny język (półformalny):
    - użytkowników biznesowych,
    - analityków-informatyków
  - Zasadniczo jeden rodzaj diagramu – duża siła ekspresji
  - Sformalizowanie notacji umożliwia automatyczne wygenerowanie kodu BPEL-WS (automatyzacja implementacji)

BPMN – podsumowanie

- Słabości:
  - Skupienie na modelowaniu procesów biznesowych ze słabym modelowaniem przepływu danych; nic nie mówi o strukturze i dostępie do danych.
  - Nie najlepsze odwzorowanie organizacji firmy; skupienie się na typach procesów biznesowych.
  - Słaby opis dynamiki grup oraz hierarchii użytkowników.

BPMN – podsumowanie

- Inne źródła:
  - http://www.slideshare.net/samarin/how-to-use-bpmn-for-modelling-business-processes?related=1
  - http://www.slideshare.net/mzurmuehlen/bpmn-20-part-1-basic-constructs?related=2
  - http://www.slideshare.net/mzurmuehlen/getting-started-with-business-process-modeling?related=3

UML – diagramy

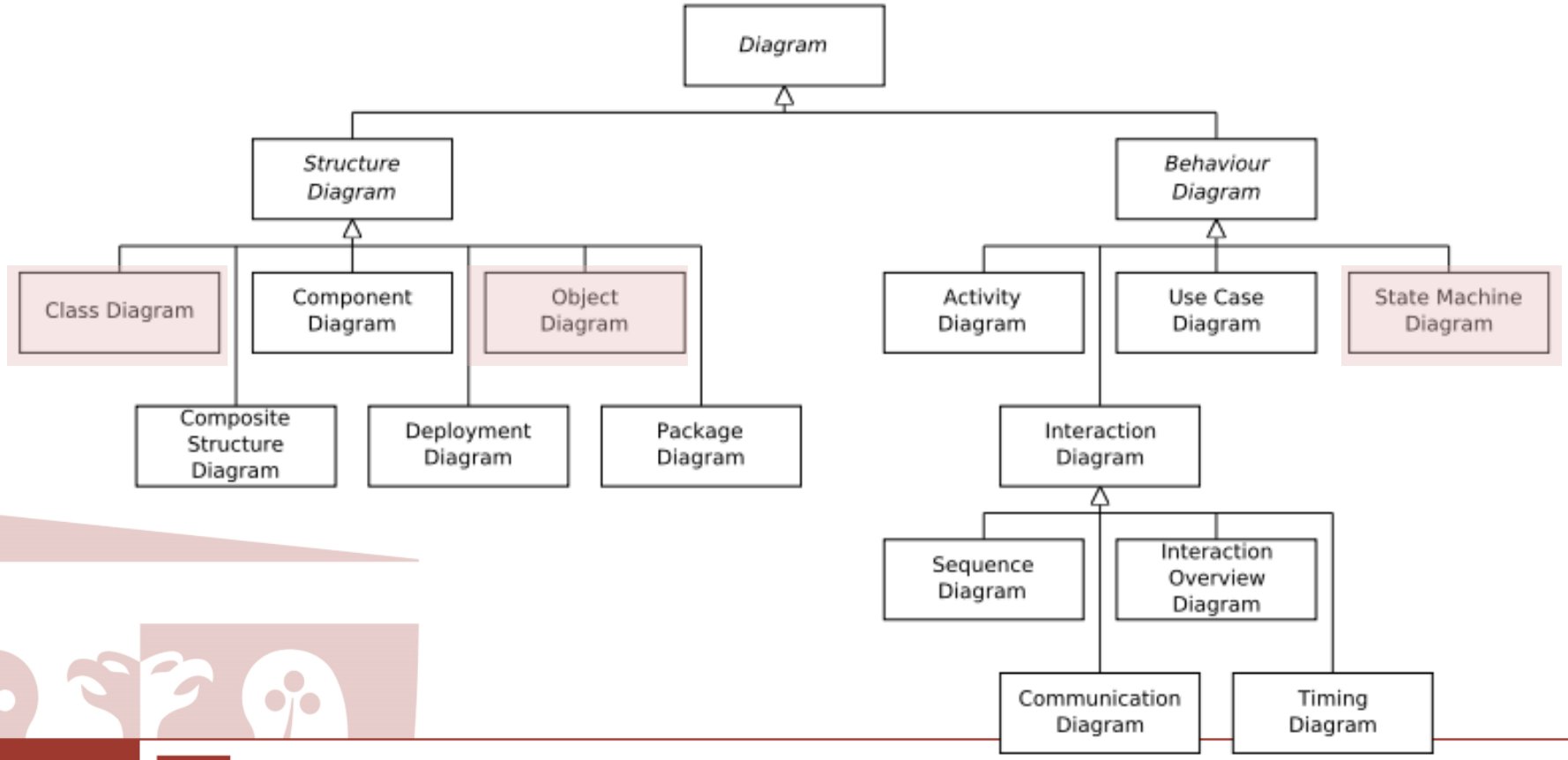


Diagram stanów

- Graficzna reprezentacja maszyny stanowej przypisanej do:
  - klasy
  - podsystemu
  - komponentu
  - metody
- Generator zbioru dopuszczalnych ciągów stanów w czasie życia obiektu

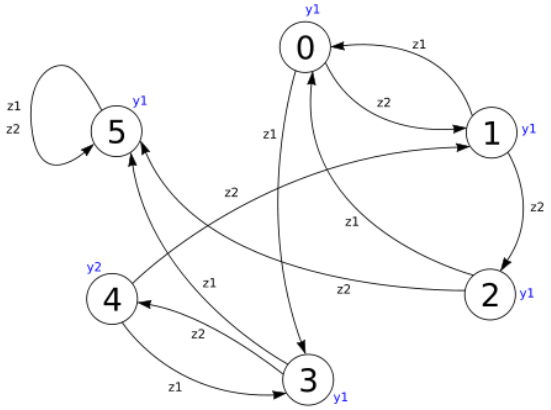
Diagram stanów

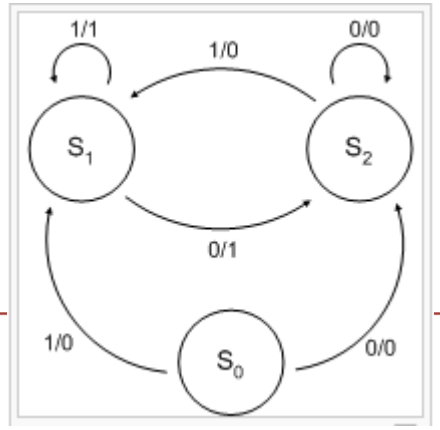
- Maszyna stanowa – abstrakcyjny, matematyczny, iteracyjny model zachowania systemu dynamicznego oparty na dyskretnych przejściach między jego kolejnymi stanami (diagram stanów)
  - deterministyczne
  - niedeterministyczne

Maszyna stanowa

- Maszyna stanowa =

  skończony automat + pamięć
- Skończony automat
  - Moore, Mealy
- Obliczenie automatu skończonego
  - ciąg przejść pomiędzy stanami





Maszyna stanowa

- Pamięć – stany pamięci
  - Zbiór funkcji wartościowania zmiennych (zbiór wartościowań)
- Konfiguracja maszyny stanowej:

  <stan, wartościowanie>
- Obliczenie maszyny stanowej
  - ciąg przejść pomiędzy konfiguracjami

Diagram stanów

- Wierzchołki (stany)
  - Stany proste
  - Stany złożone
    - zagnieżdżone sekwencyjnie
    - zagnieżdżone równolegle
  - Pseudostany
- Łuki
  - Tranzycje pomiędzy wierzchołkami (stanami)

Stan

- Nieformalnie – informacja określająca przyszłe zachowania obiektu pod wpływem napływających zdarzeń
- Dokładniej – sytuacja w czasie życia obiektu określona przez:
  - aktualne wartościowanie atrybutów
  - aktualne powiązania z innymi obiektami
  - aktualnie wykonywane aktywności
  - oczekiwanie na pewne zdarzenia

Stan prosty

- Brak stanów zagnieżdżonych
- Może posiadać:
  - akcje wejściowe **entry**/akcja-we
  - akcje wyjściowe **exit**/akcja-wy
  - tranzycje wewnętrzne zdarzenie/akcja
  - aktywności wewnętrzne **do**/aktywność
  - zdarzenia odroczone zdarzenie/**defer**

Stany proste – notacja

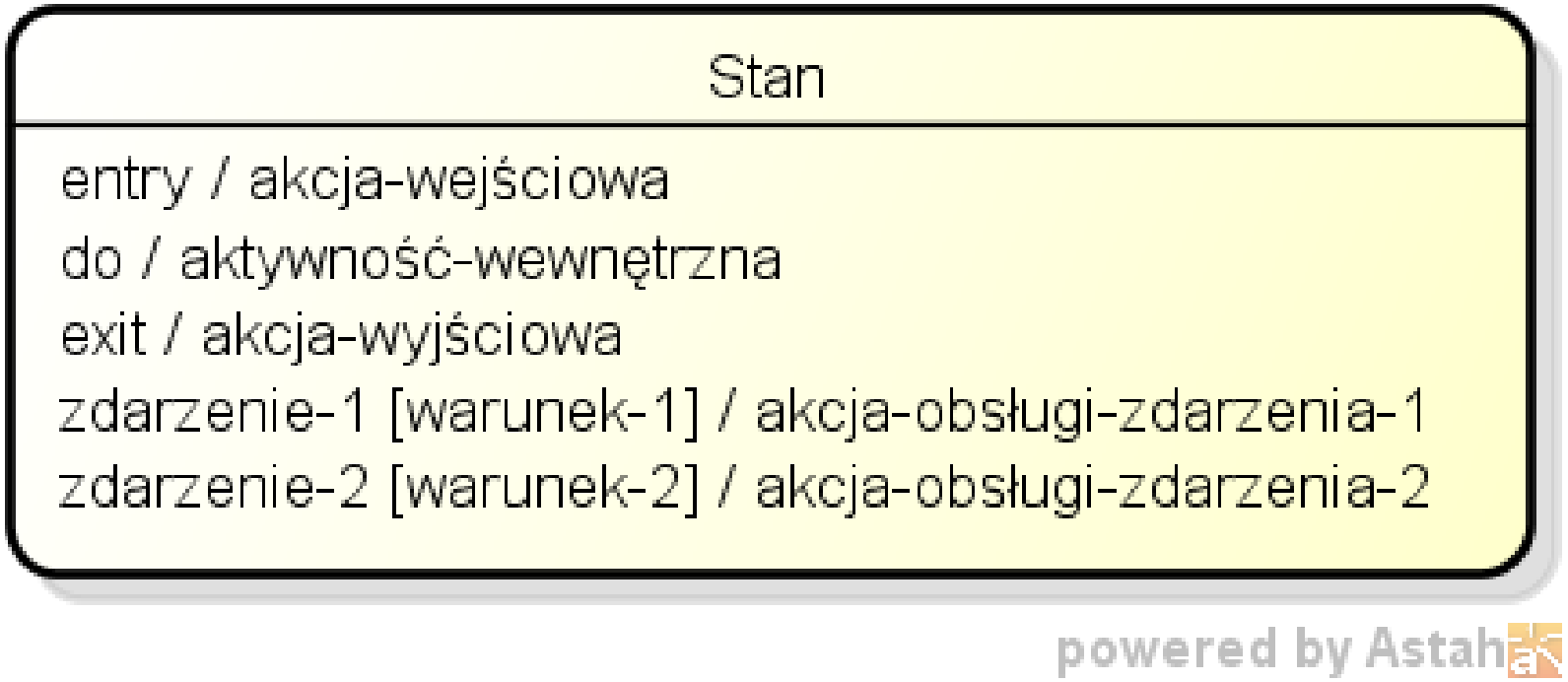


Diagram stanów
- tranzycje pomiędzy stanami

stary stan aktywny → nowy stan aktywny

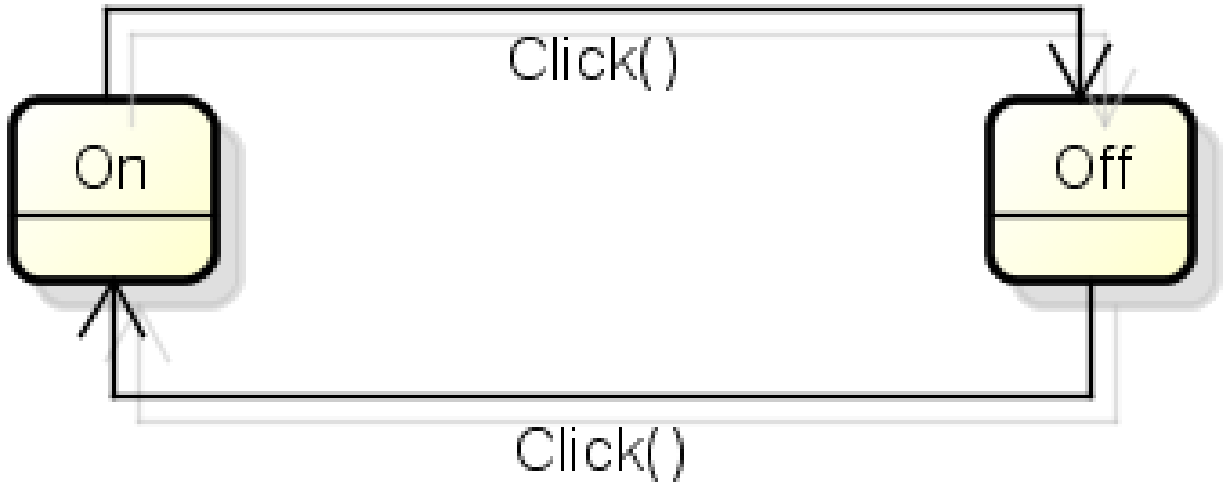


Diagram stanów
- tranzycje pomiędzy stanami

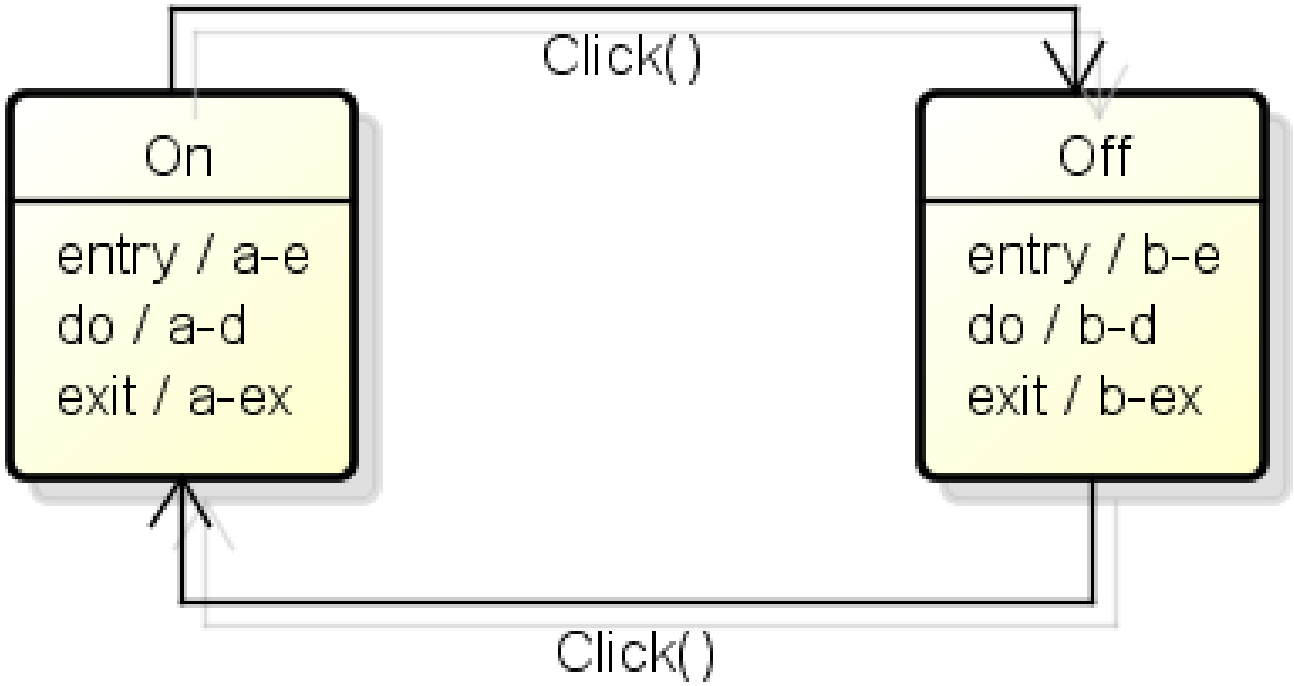


Diagram stanów
- tranzycje pomiędzy stanami

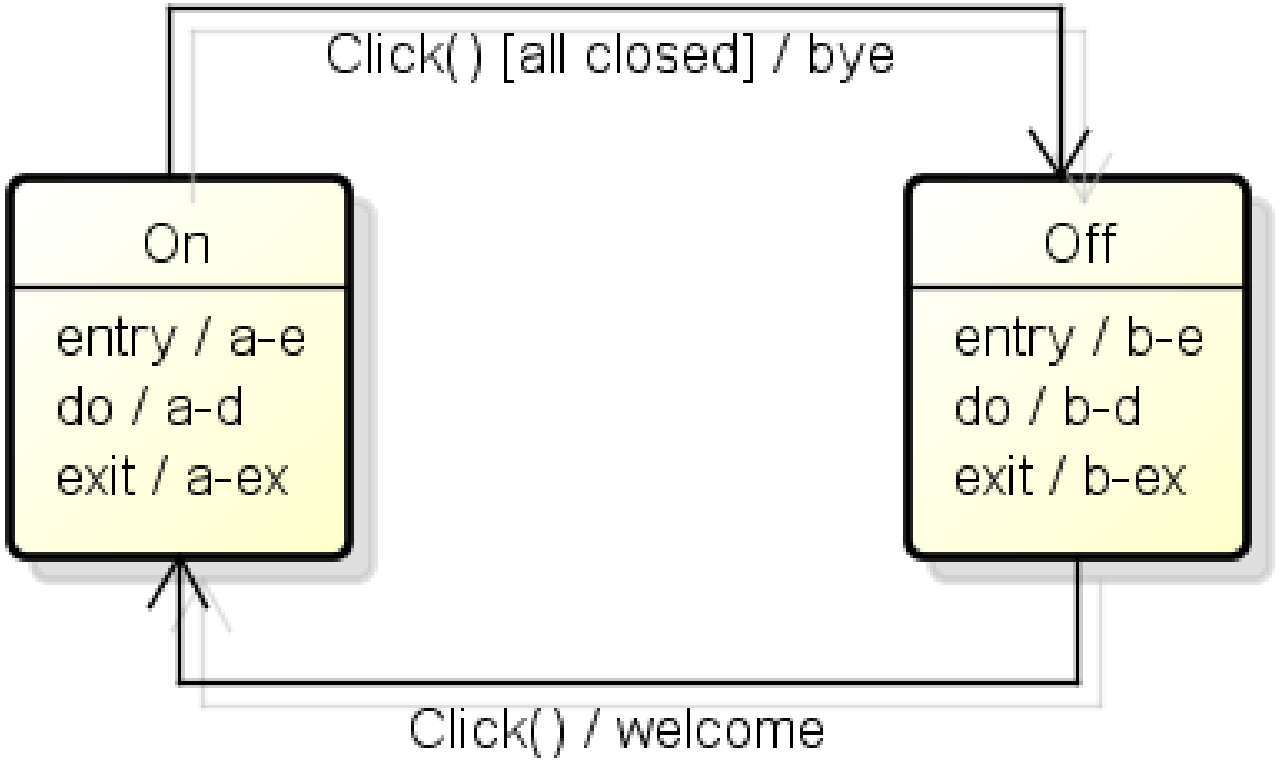
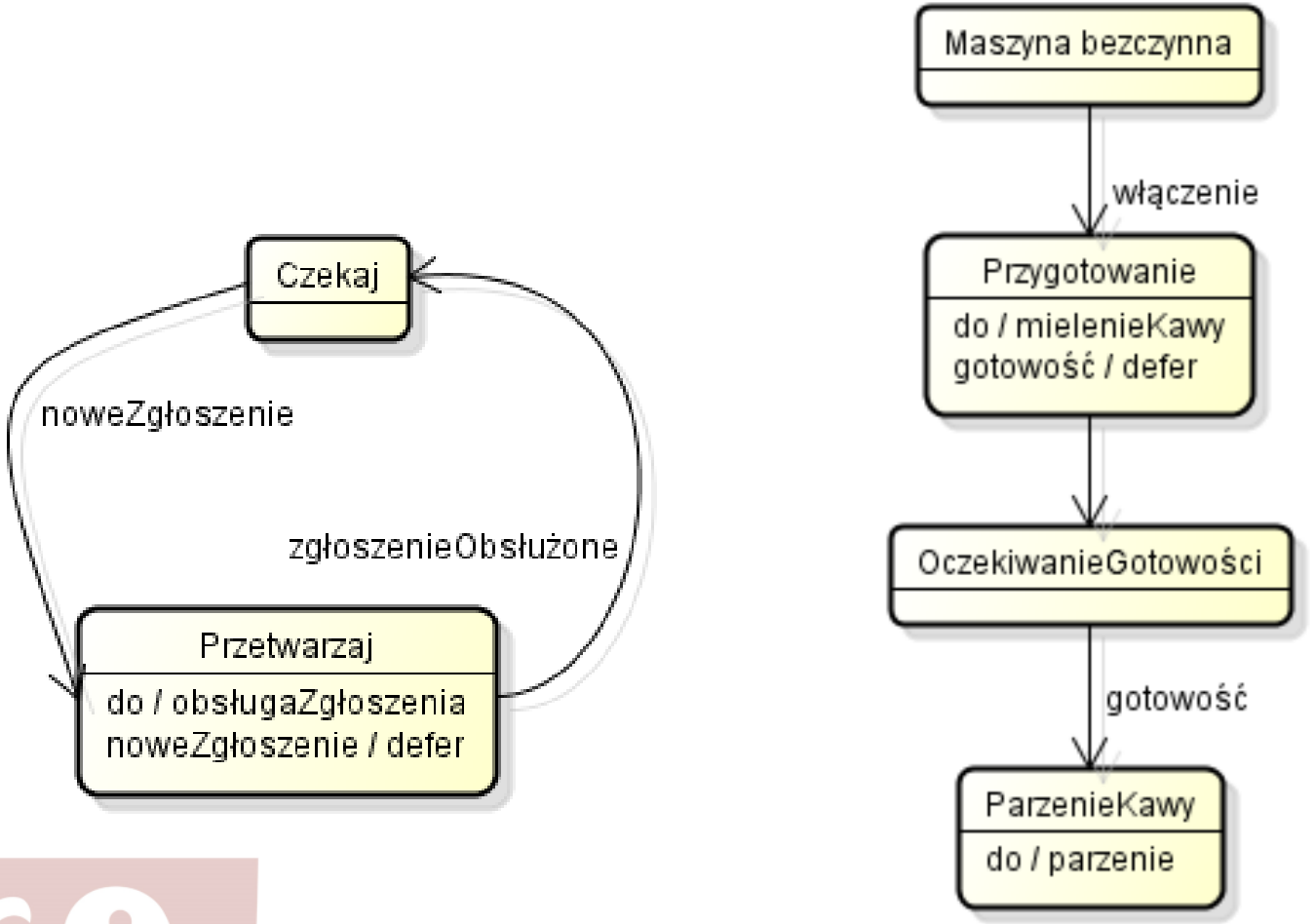


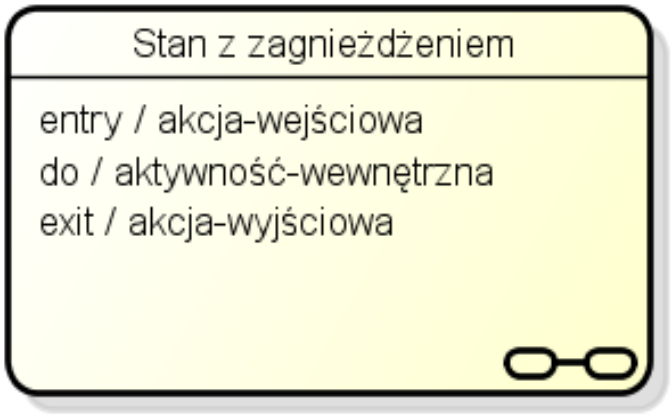
Diagram stanów – zdarzenie odroczone

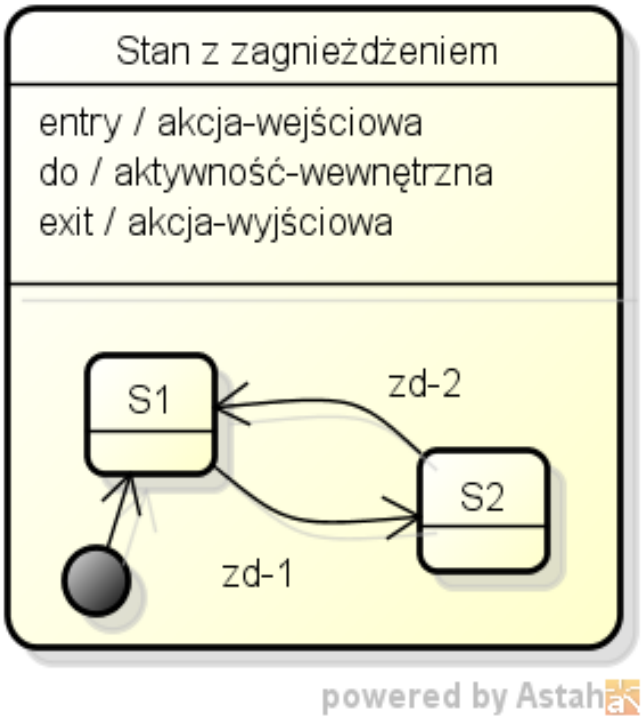


Stany złożone

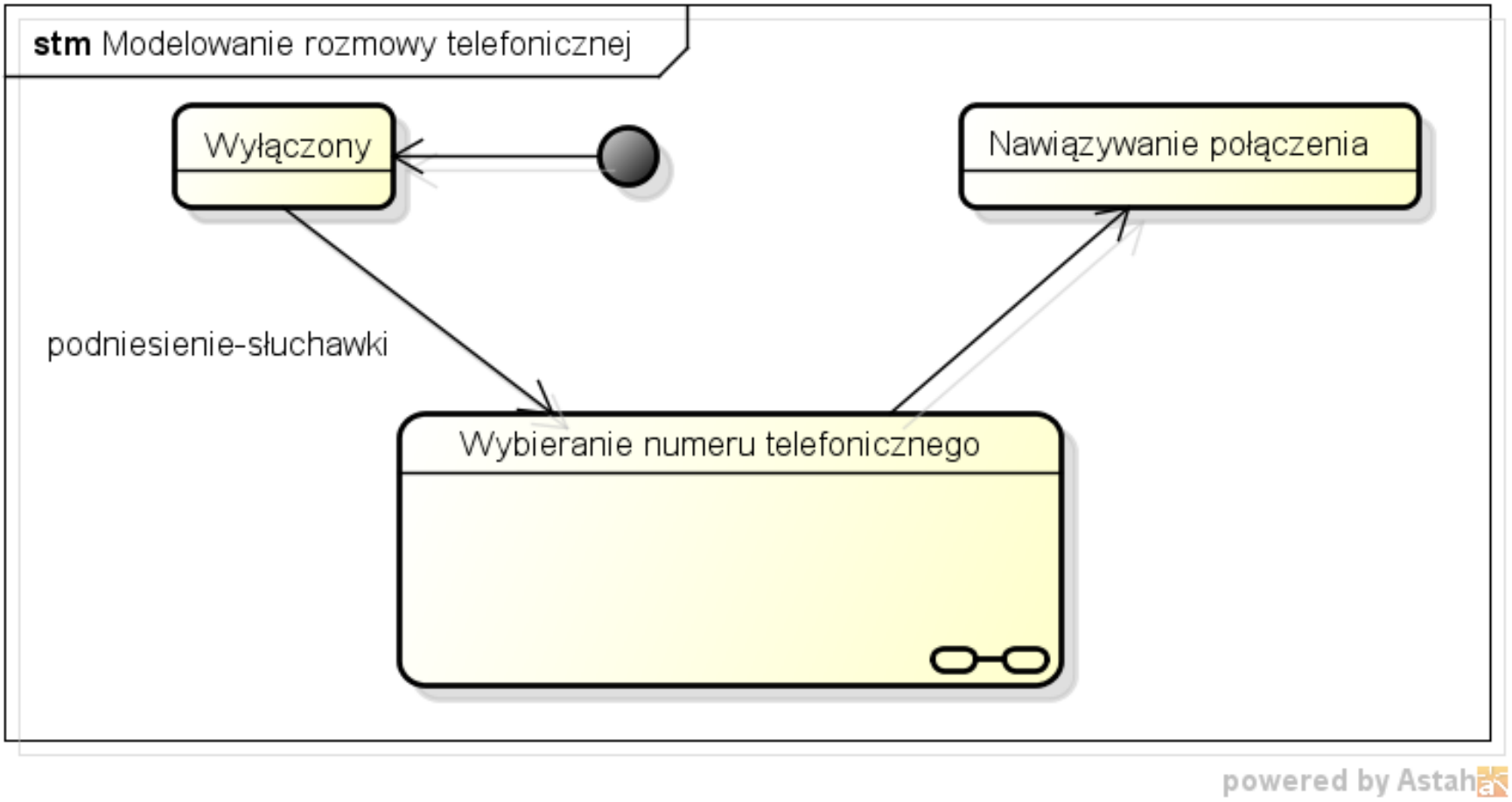
- Stan złożony – zawiera co najmniej jeden region zagnieżdżenia (diagram stanów):
  - jeden region – stan złożony sekwencyjnie
    - jeśli stan jest aktywny, to dokładnie jeden z jego podstanów jest aktywny
  - więcej regionów – stan złożony równolegle
    - jeśli stan jest aktywny, to dokładnie jeden z jego podstanów w każdym regionie jest aktywny

Stan złożony sekwencyjnie (stan nieortogonalny)

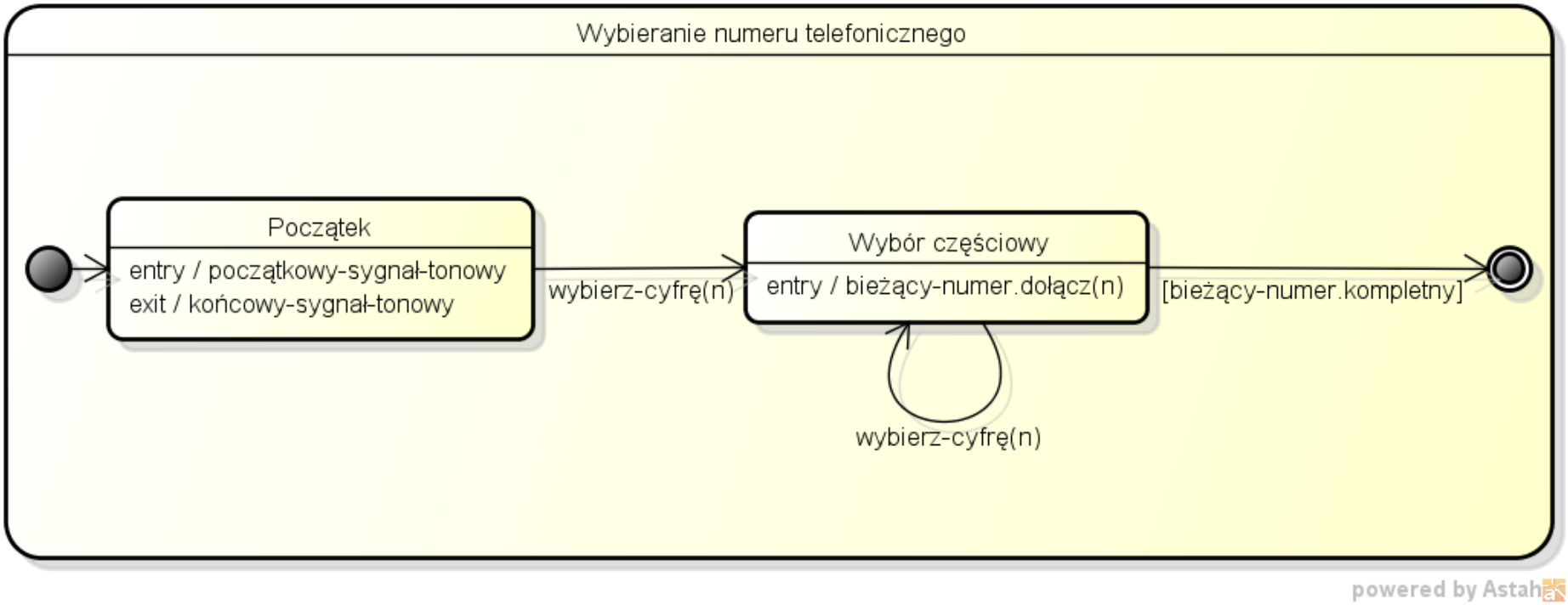




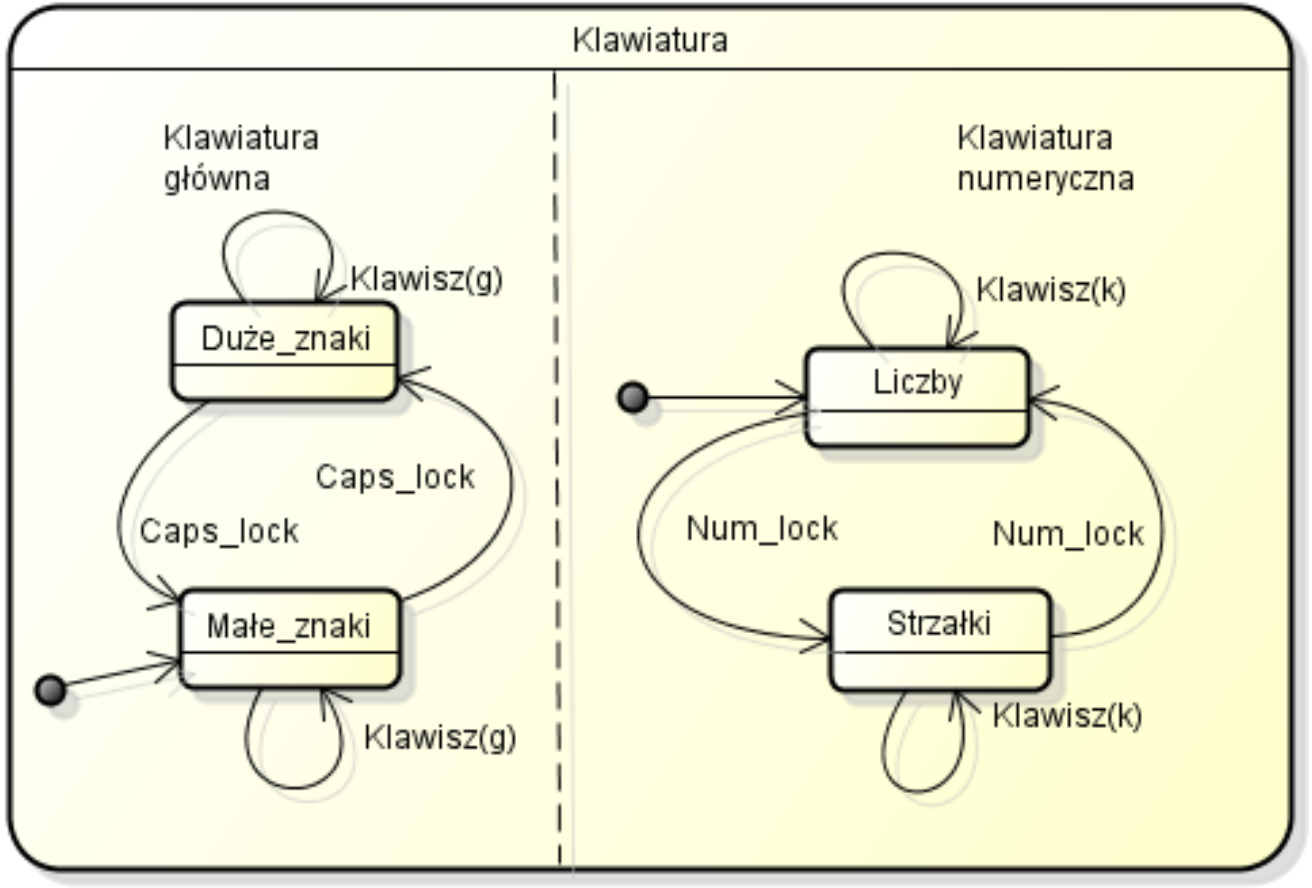
Stan złożony sekwencyjnie (stan nieortogonalny)



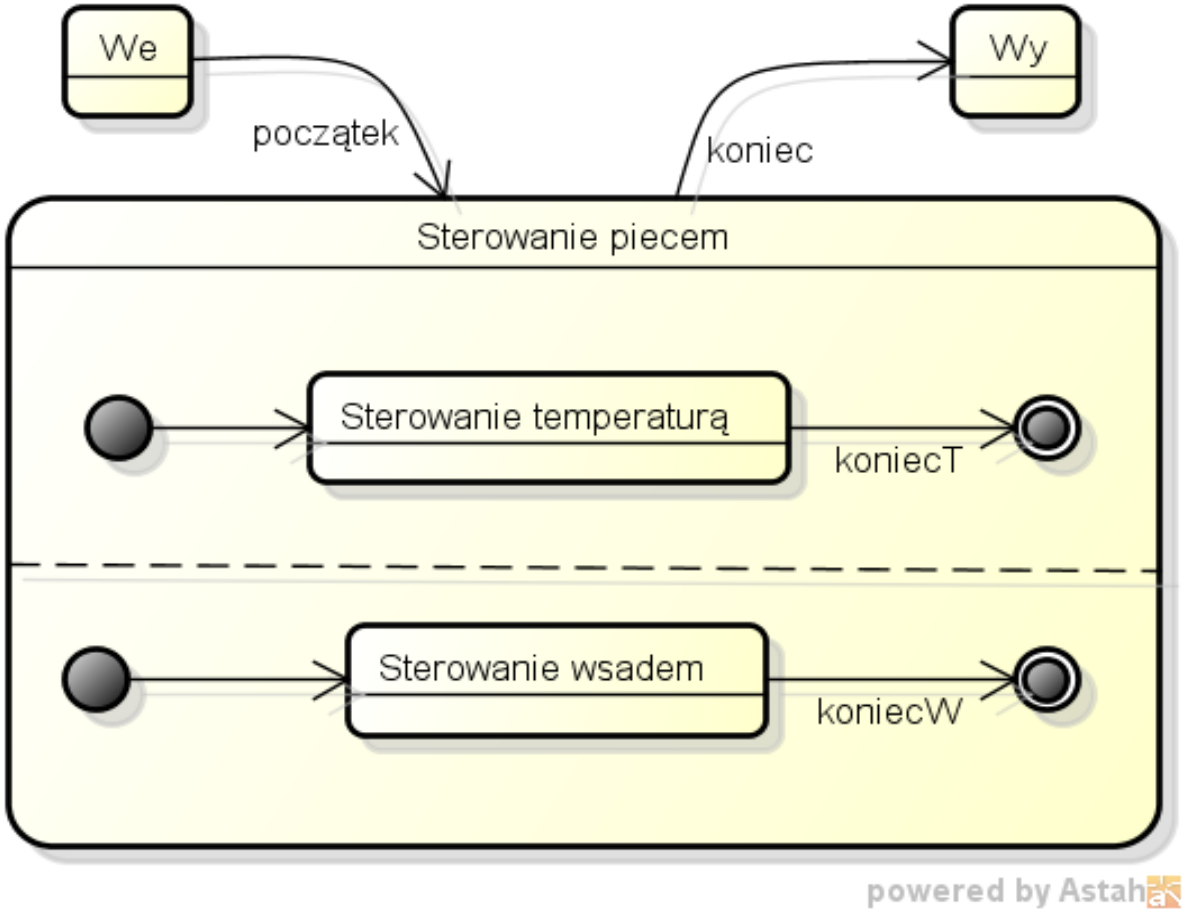
Stan złożony sekwencyjnie (stan nieortogonalny)



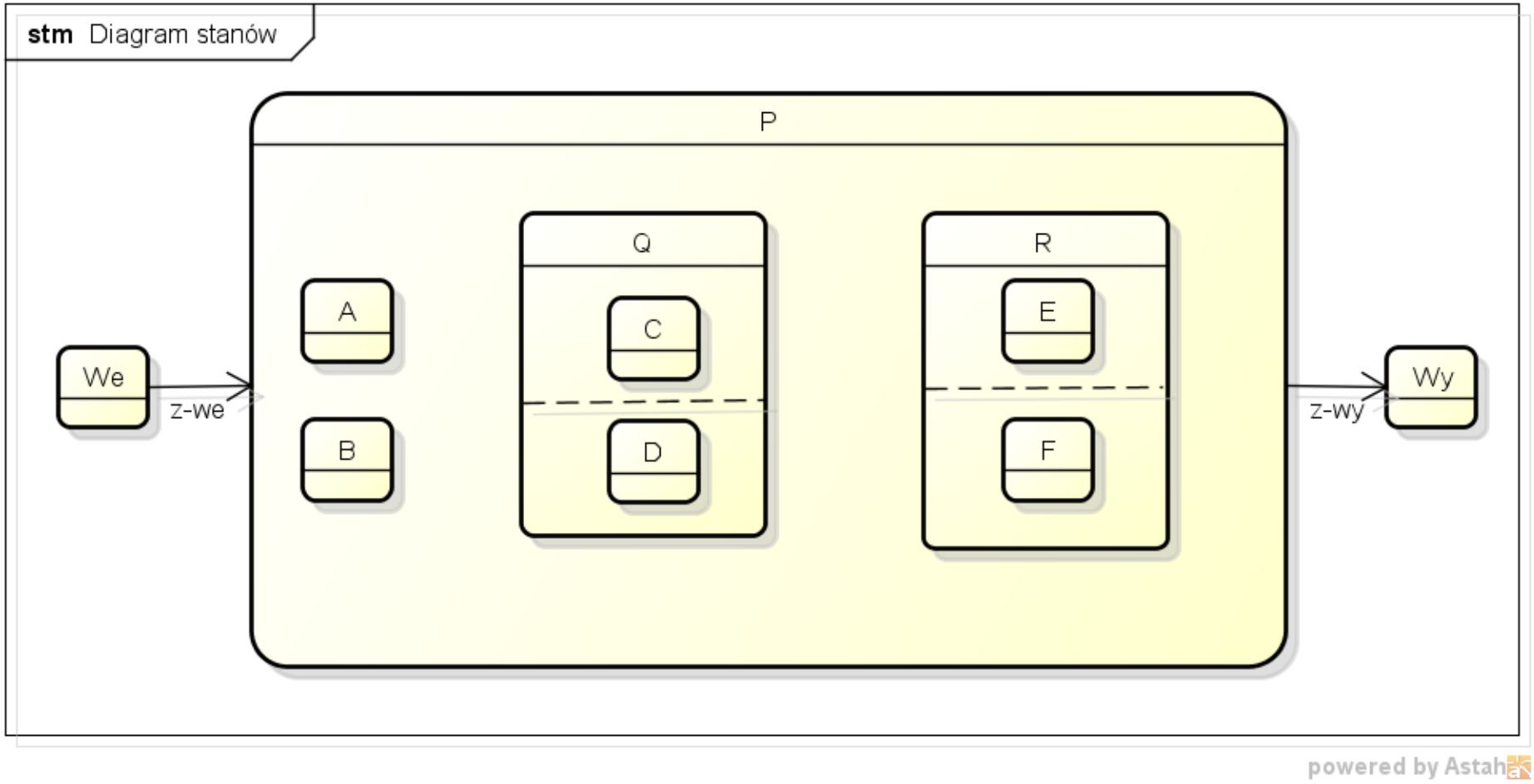
Stan złożony równolegle (stan ortogonalny)



Stan złożony równolegle (stan ortogonalny)



Konfiguracje stanów aktywnych



Pseudostany – notacja

(fork , join)
rozgałęzienia, złączenia

równoległego
(initiate)
początkowy

(shallow history)
łącznik historyczny, płytki

(junction)
złączenia

(choice)
wyboru

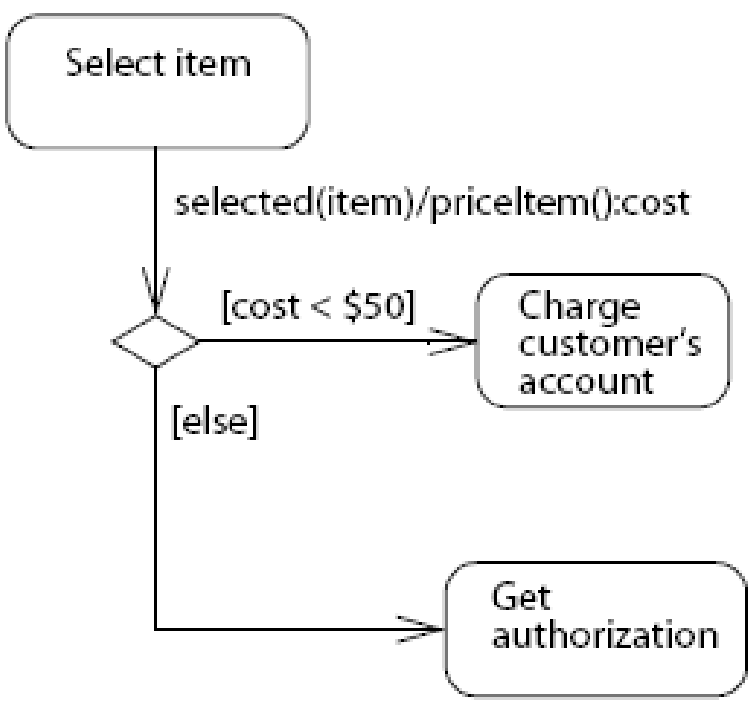
(final)
końcowy

Nie jest pseudostanem!

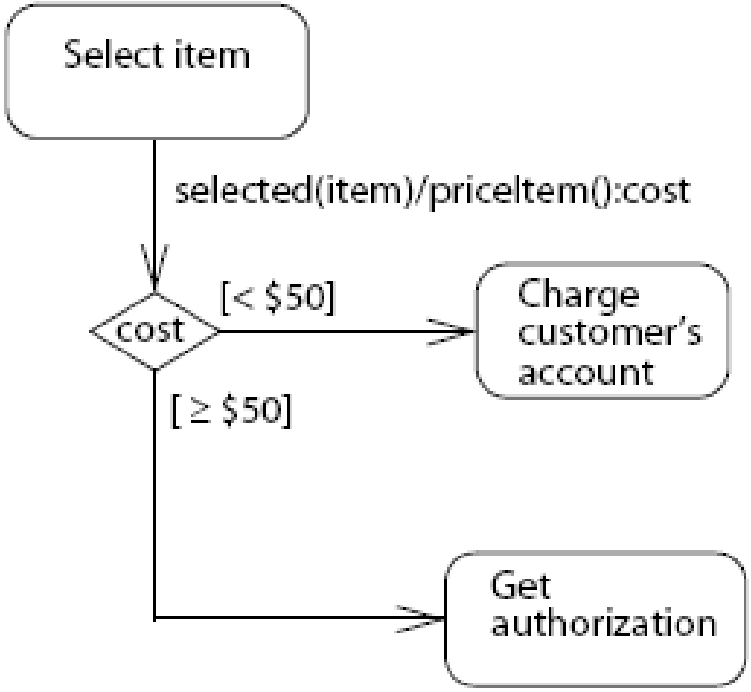
(deep history)
łącznik historyczny, głęboki

(terminate)
terminalny

Pseudostany – wybór



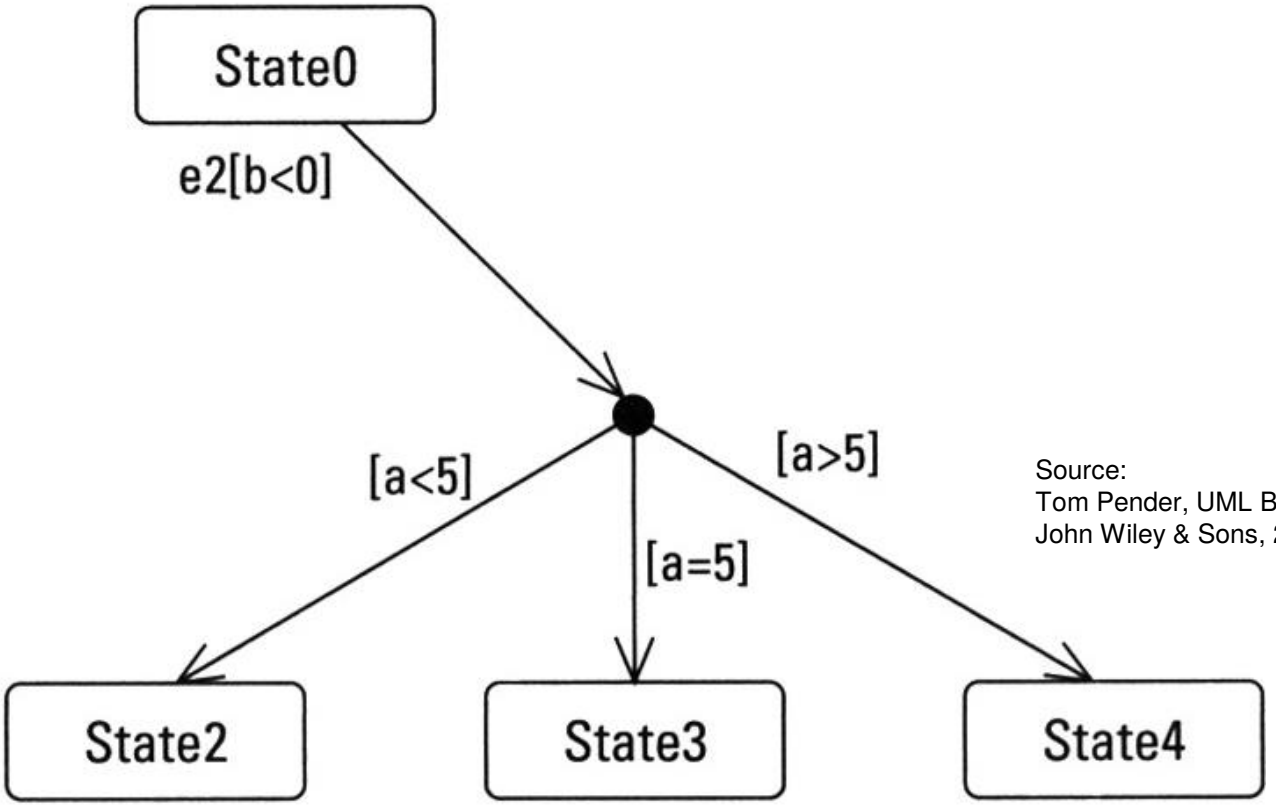
explicit guard conditions



shorthand guard conditions

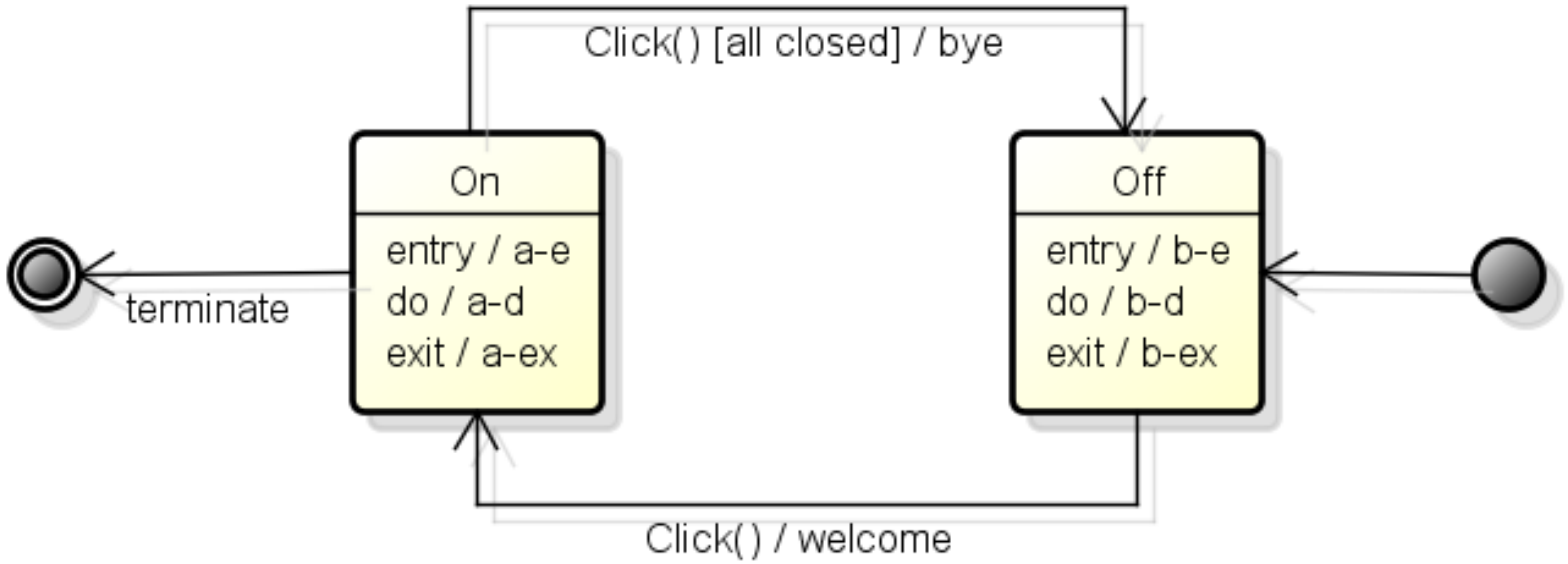
Source:
Rumbaugh J., Jacobson I., Booch G., *The Unified Modeling Language – Reference Manual*. Second edition, Addison-Wesley, 2005

Pseudostany – złączenie



Source:
Tom Pender, UML Bible,
John Wiley & Sons, 2003

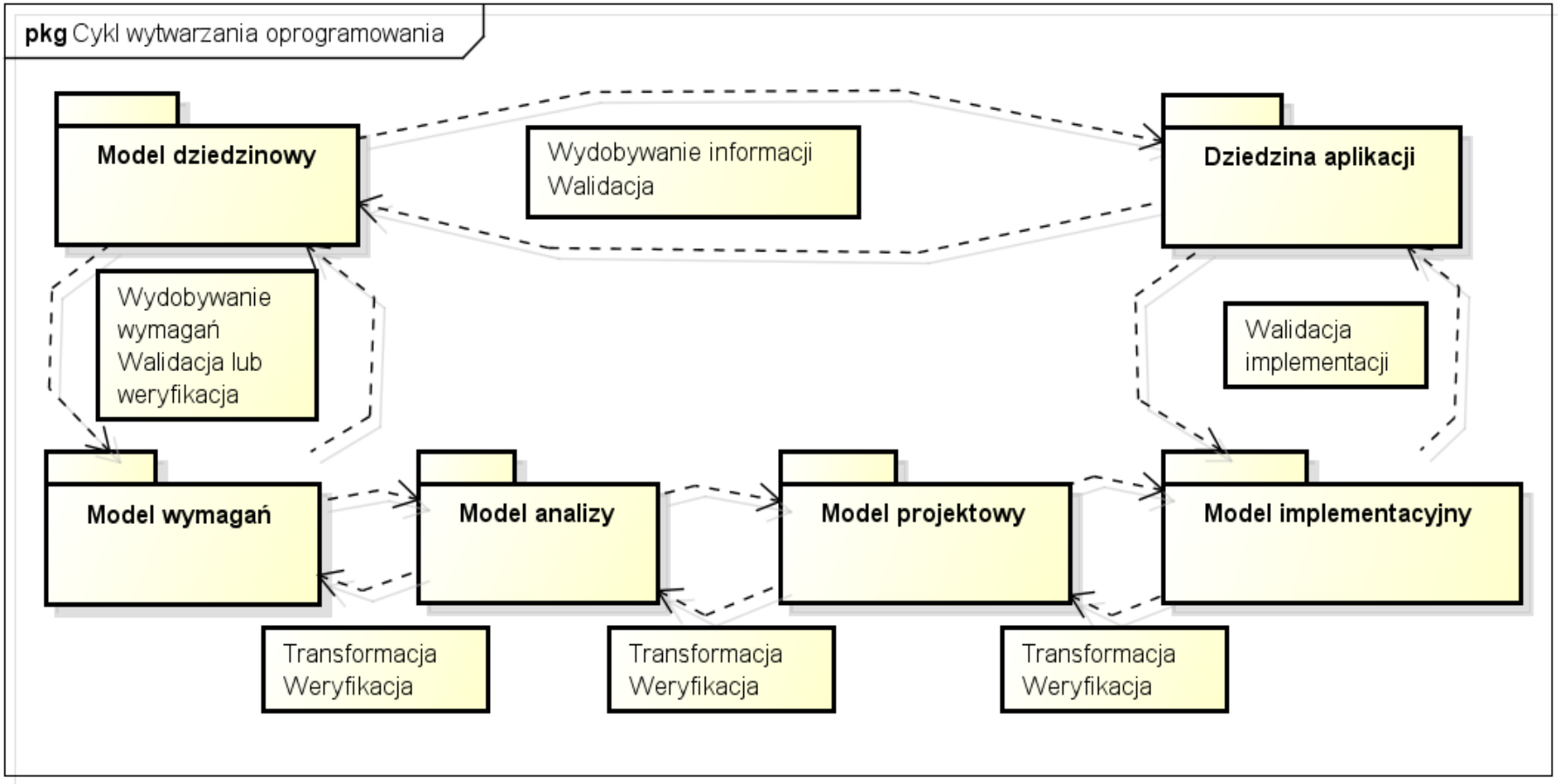
Stan końcowy



Diagramy stanów – zastosowania

- Podstawowe – opis zachowania klasy (komponentu, podsystemu); pokazanie jak obiekt reaguje na zdarzenia i jak zmienia swojej wewnętrzne stany
- Opis protokołów – elementów interfejsu klas, komponentów, podsystemów; pokazanie dopuszczalnych kolejności wywołań operacji (sygnałów) w ramach danego interfejsu

Cykl wytwarzania oprogramowania



Model wymagań

- Wymagania funkcjonalne
- Wymagania niefunkcjonalne:
  - dotyczące produktu (jakość produktu):
    - wydajność
    - niezawodność
    - dyspozycyjność
    - przenośność
    - …
  - dotyczące procesu wytwarzania

Wizja systemu – kontekst wymagań

- Sformułowanie rozwiązywanego problemu
- Zestawienie udziałowców i użytkowników wraz z podaniem ich potrzeb
- Opis systemu
  - ogólna architektura
  - możliwości
  - Właściwości
    - Funkcjonalne
    - Niefunkcjonalne

Wizja systemu – przykładowy problem

System *Demograf* ma służyć opracowywaniu analiz zagadnień demograficznych na podstawie danych gromadzonych przez GUS, zaś wyniki tych analiz mają być podstawą do przygotowania publikacji w piśmie *Demografia*.

Zainteresowany powstaniem systemu jest wydawca tego pisma.

Wizja systemu – opisy

Na wizję systemu składają się opisy:

- sformułowanie rozwiązywanego problemu,
- zestawienie udziałowców i użytkowników wraz z podaniem ich potrzeb,
- opis systemu, jego ogólnej architektury, możliwości i własności.

Wizja systemu – opis celu (1)

Celem dokumentu jest zebranie, analiza i definicja własności systemu *Demograf*.

W dokumencie opisano wizję systemu, który – na podstawie danych demograficznych zaimportowanych z innych systemów – umożliwia generację raportów dotyczących liczby osób pozostających w związkach (formalnych, nieformalnych) oraz ich potomstwa. Projekt jest powiązany z projektem zainicjowanym przez pismo *Demografia*. Powstaje przy współpracy Głównego Urzędu Statystycznego (GUS).

Wizja systemu – opis problemu (2)

Problem	a) Trudny dostęp do aktualnej informacji demograficznej o danym regionie. Niezadowalający czas oczekiwania na informacje demograficzne. b) Czasochłonna realizacja wniosków o udostępnienie informacji publicznej przez GUS. Istniejący system BDR nie pozwala opracować wszystkich potrzebnych raportów. c) Niska popularność i znajomość pisma Demografia.
Dotyczy	a) Każdy zainteresowany aktualną informacją (np. dziennikarze). b) Urzędy statystyczne. c) Właściciele pisma Demografia.
Wpływ problemu	a) Niezadowolenie osób szukających danych demograficznych spowodowane długim czasem wyszukiwania lub oczekiwania (w przypadku złożenia wniosku do urzędu statystycznego) na potrzebne informacje. b) Konieczność angażowania zasobów (czas, ludzie) do realizacji wniosków o udostępnienie informacji publicznej. c) Niska liczba abonentów, sprzedaż małych nakładów.
Pomyślne rozwiązanie	a) Dostęp do aktualnych informacji demograficznych w dowolnym miejscu i czasie. b) Redukcja liczby spływających wniosków poprzez bieżące udostępnienie części informacji publicznej. c) Reklama pisma Demografia.

Wizja systemu – opis udziałowców (3)

Nazwa	Opis/Rola	Kompetencje
Wiceprezes GUS	Reprezentuje GUS. Nadzoruje prace Departamentu Informacji.	Wyraża zgodę na udostępnienie danych statystycznych pod warunkiem zachowania ich poufności.
Administrator BDR	Reprezentuje GUS.	Odpowiada za nadawanie uprawnień do danych udostępnianych za pomocą systemu BDR.
Właściciel gazety Demografia	Reprezentuje stronę nabywcy.	Zleca wykonanie systemu. Odpowiada za porozumienie z GUS w sprawie dostępu do BDR. Formułuje wymagania na system.
Kierownik projektu	Reprezentuje stronę wytwórcy.	Odpowiada za terminowe dostarczenie systemu zgodnego ze specyfikacją
Zespół projektowy	Reprezentuje stronę wytwórcy.	Wytwarza produkt i dokumentację według wybranej metodyki.
Szef działu Informacje statystyczne w piśmie Demografia	Reprezentuje stronę nabywcy.	Ekspert dziedzinowy. Formułuje wymagania dotyczące raportowania.
Użytkownik końcowy	Osoba zainteresowana wykorzystaniem analiz demograficznych. Reprezentuje użytkownika systemu.	Korzysta z własności oferowanych przez system do zaspokojenia swoich potrzeb.
Administrator	Osoba odpowiedzialna za utrzymanie systemu Demograf po jego wdrożeniu.	Pielęgnuje system Demograf.

Wizja systemu – opis użytkowników (4)

Nazwa	Opis/Rola	Kompetencje
Anonimowy użytkownik	Niezarejestrowany użytkownik, potencjalnie zainteresowany pozyskaniem danych demograficznych.	Bez logowania ma dostęp do ogólnych informacji o systemie Demograf. Składa aplikację o konto użytkownika.
Użytkownik	Reprezentuje zarejestrowanego użytkownika systemu posiadającego konto.	Ma dostęp do danych demograficznych. Używa systemu do pozyskania informacji statystycznej.
Administrator	Użytkownik systemu odpowiedzialny za pielęgnację bazy danych i systemu Demograf.	Zakładanie kont dla użytkowników. Pielęgnacja systemu.
Specjalista ds. reklamy	Przedstawiciel pisma Demografia.	Publikuje w systemie reklamy pisma Demografia.

Wizja systemu – opis potrzeb (5)

Kluczowe potrzeby udziałowców i użytkowników

Potrzeba	Priorytet	Źródło
Bezpieczeństwo danych indywidualnych	Wysoki	Administrator BDR, Wiceprezes GUS
Raportowanie danych demograficznych	Wysoki	Szef działu Informacje Statystyczne w gazecie Demografia.
Reklama pisma Demografia	Wysoki	Właściciel pisma Demografia
Łatwość pielęgnacji systemu Niskie koszty utrzymania.	Średni	Właściciel pisma Demografia Wytwórca

Wizja systemu – opis architektury (6)

Architektura systemu jest trzypoziomowa:

- cienki klient,
- serwer aplikacji,
- serwer(y) bazy danych.

Użytkownik korzysta z systemu za pomocą dowolnej przeglądarki internetowej.

Aplikacja jest zbudowana z wykorzystaniem ADO.Net i posadowiona na serwerze aplikacji.

Dane demograficzne są przechowywane i pobierane przez serwer aplikacji z serwera bazy danych.

Wizja systemu – opis możliwości (7)

Korzyść	Cechy wspierające
Dostęp do aktualnej informacji statystycznej. Możliwość badań zmian tendencji statystycznej w czasie.	Raportowanie. Import danych.
Popularyzacja pisma Demografia	Banery reklamowe
Bezpieczne udostępnianie wybranych analiz demograficznych	Rejestracja do systemu. Rejestracja aktywności użytkowników. Bezpieczny protokół komunikacyjny.

Wizja systemu – opis właściwości(8)

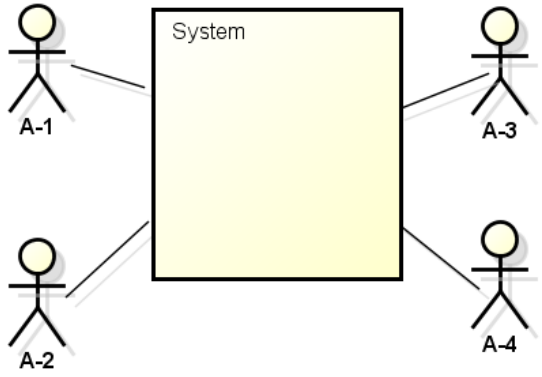
- FEAT 1 (<<functional>>) – Raportowanie: System udostępnia raporty dotyczące danych demograficznych i ich zmian w czasie.
- FEAT 2 (<<functional>>) – Import danych: System pobiera i przechowuje kopie wybranych danych z systemu BDR.
- FEAT 3 (<<non-functional>>) – System jest dostępny w typowych przeglądarkach internetowych (minimum Explorer, FireFox).
- FEAT 4 (<<functional>>) – System jest dostępny w polskiej i angielskiej wersji językowej.
- FEAT 5 (<<functional>>) – Rejestracja do systemu: Dostęp do systemu mają tylko uprawnieni użytkownicy.
- FEAT 6 (<<functional>>) – Rejestracja aktywności użytkowników: System – ze względów bezpieczeństwa – rejestruje wszystkie działania użytkowników.
- FEAT 7 (<<functional>>) – Banery reklamowe: System wyświetla banery reklamowe i umożliwia ich zmianę.
- FEAT 8 (<<non-functional>>) – System udostępnia żądane raporty w wymaganym czasie.

Wizja systemu – opis właściwości (9)

- FEAT 9 (<<non-functional>>) – System ma prosty, czytelny interfejs.
- FEAT 10 (<<non-functional>>) – System jest łatwy w pielęgnacji.
- FEAT 11 (<<functional>>) – Bezpieczny protokół komunikacyjny: komunikacja z systemem (z wyjątkiem rejestracji) używa bezpiecznego protokołu komunikacyjnego.
- FEAT 12 (<<functional>>) – Zarządzanie kontami: system umożliwia usuwanie nieaktywnych kont, zmianę hasła etc.
- FEAT 13 (<<functional>>) – Statystyki systemu: system umożliwia odczyt logów systemowych oraz wybranych statystyk wykorzystania systemu.

Model przypadków użycia

- Element składowy wymagań funkcjonalnych; używany w początkowych fazach wytwarzania systemów oprogramowania.
- Reprezentuje funkcjonalność systemu z punktu widzenia otoczenia systemu (aktora systemu).



Model przypadków użycia

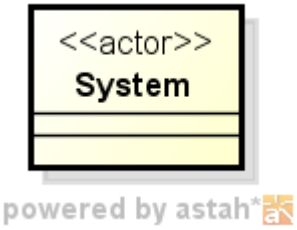
- Jest spojrzeniem na dynamikę (zachowanie) systemu poprzez tzw. przypadki użycia.
- Składa się z:
  - diagramu przypadków użycia
  - oraz ewentualnie innych diagramów opisujących przypadki użycia.

Diagram przypadków użycia

- System (subject)
- Wierzchołki:
  - Przypadki użycia
  - Aktorzy
- Łuki:
  - Asocjacje
  - Generalizacje
  - Zależności: <<include>>, <<extend>>

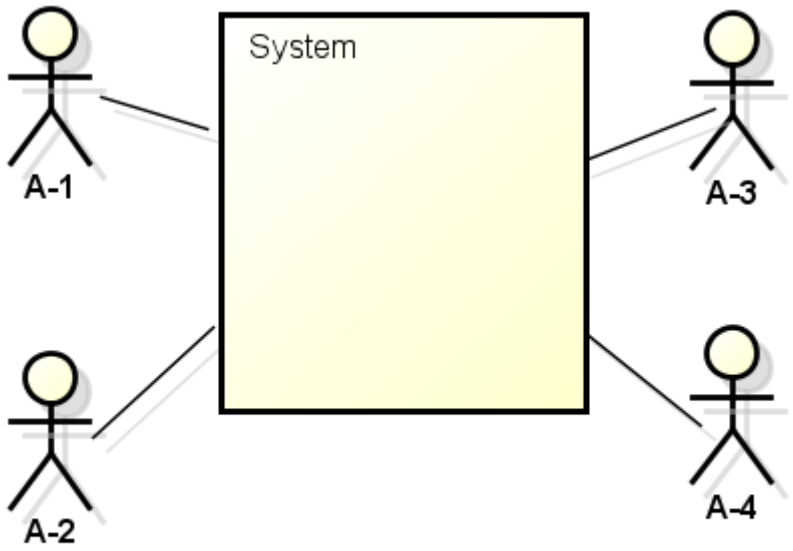
Aktor

- Składnia



- Semantyka
  - reprezentuje te byty z otoczenia systemu, które z systemem współdziałają (spójny zbiór ról)
  - klasyfikator (typ bytów/byt konkretny)

Aktorzy i system



- Aktorzy: osoby, inne systemy, urządzenia elektroniczne, mechaniczne, ...

Przypadek użycia

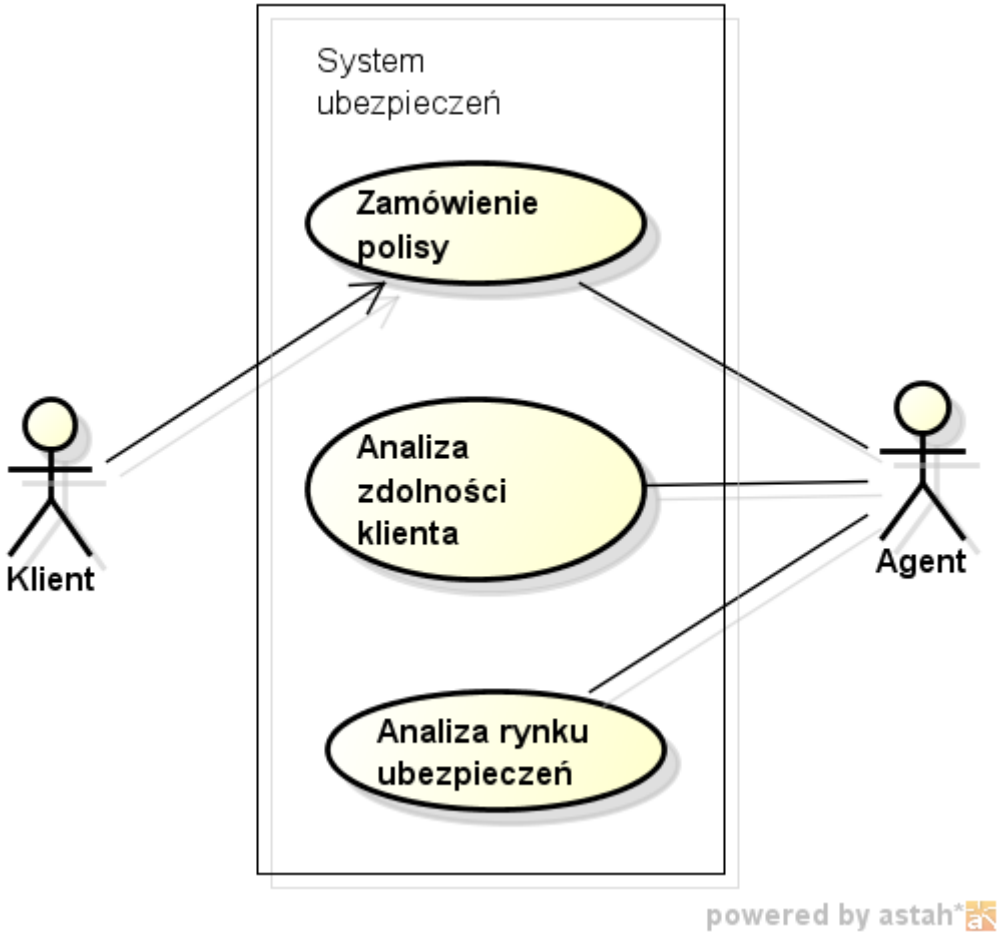
- Może reprezentować:
  - specyfikację zewnętrznych wymagań na system
  - specyfikację funkcjonalności systemu
  - specyfikację tego, jak aktorzy powinni współpracować z systemem, aby korzystać z jego usług

Przypadek użycia

- Składnia

- Semantyka
  - jednostka usługi (funkcjonalności); reprezentuje rodzaj usługi, który system udostępnia swemu otoczeniu
  - klasyfikator (typ usługi/konkretna usługa)

Przypadki użycia i aktorzy



Przypadek użycia

- Znaczenie przypadku użycia:
  - specyfikacja zbioru scenariuszy (ciąg wiadomości i akcji), których realizacja prowadzi do osiągnięcia pożądanego rezultatu
- Instancja przypadku użycia:
  - konkretny skończony scenariusz prowadzący do osiągnięcia pożądanego rezultatu

Przypadek użycia

- Opis przypadku użycia – zbiór scenariuszy
- Scenariusz – ciąg kroków
- Krok jest opisany przez:
    i. zdarzenie inicjujące
    ii. rola (aktor, system)
    iii. akcja (proces funkcjonalny)
    iv. obiekt danych

  widziane na ustalonym interfejsie

Przypadek użycia – przykład

Dodaj rezerwację

Warunek wstępny: wolny pokój spełniający wymagania

1. Pracownik klika przycisk Rezerwuj na głównym ekranie aplikacji.
2. System otwiera formularz dodawania nowej rezerwacji.
3. Pracownik wyszukuje pokoje spełniające odpowiednie wymagania na podstawie filtrów wyszukiwania.
4. System wyświetla listę pokoi spełniających kryteria.
5. ……

Przypadek użycia – opis

- Podstawowe sposoby:
  - opis tekstowy
  - diagram
    - aktywności
    - stanów
    - sekwencji

Przypadek użycia – opis tekstowy

1. Nazwa
2. Założenia dotyczące kontekstu jego użycia
3. Cel
   - Jaki jest oczekiwany cel realizacji? Co zamierza się osiągnąć?

     Przypadki użycia są zorientowane na cele, które muszą być jawnie określone

Przypadek użycia – opis tekstowy

4. Inicjator – aktor, który inicjuje wykonanie przypadku użycia
5. Inni aktorzy uczestniczący w realizacji przypadku użycia
6. Warunek wstępny inicjacji przypadku użycia
7. Warunek końcowy po realizacji przypadku użycia

Przypadek użycia – opis tekstowy

8. Przepływ wiadomości pomiędzy aktorem a systemem podczas realizacji przypadku użycia
   1. scenariusz główny
   2. scenariusze alternatywne (wariantywne)
   3. scenariusze wyjątkowe