Model-Driven Engineering

Radosław Klimek

2015-22

http://home.agh.edu.pl/rklimek

SysML

Systems **M**odelling **L**anguage

- Systems Modeling Language (**SysML**) jest uniwersalnym językiem modelowania wizualnego do zastosowań inżynierii systemów.
- **SysML** obsługuje specyfikację, analizę, projektowanie, weryfikację i walidację w szerokim zakresie systemów i systemów – systemów.
- Systemy te mogą obejmować sprzęt, oprogramowanie, informacje, personel, procedury i wyposażenie.

Historia

- **SysML** został stworzony przez SysML Partners, nieformalne stowarzyszenie ekspertów inżynierii systemów i dostawców narzędzi do modelowania oprogramowania, zwołanym w 2003 roku w celu utworzenia profilu (dialektu) UML dla Inżynierii Systemów nazwanym później SysML.

- SysML Partners zdefiniowała **SysML** jako otwartą specyfikację, która spełnia wymagania Object Management Group's UML dla Inżynierii Sytemów.

Opracowanie założeń nowego języka

- INCOSE – International Council on System Engineering
- OMG SE DESIG – Object Management Group System Engirneering Domain Special Interest Group
- ISO AP 233 – ISO Application Protocol 233

Chronologia wydarzeń

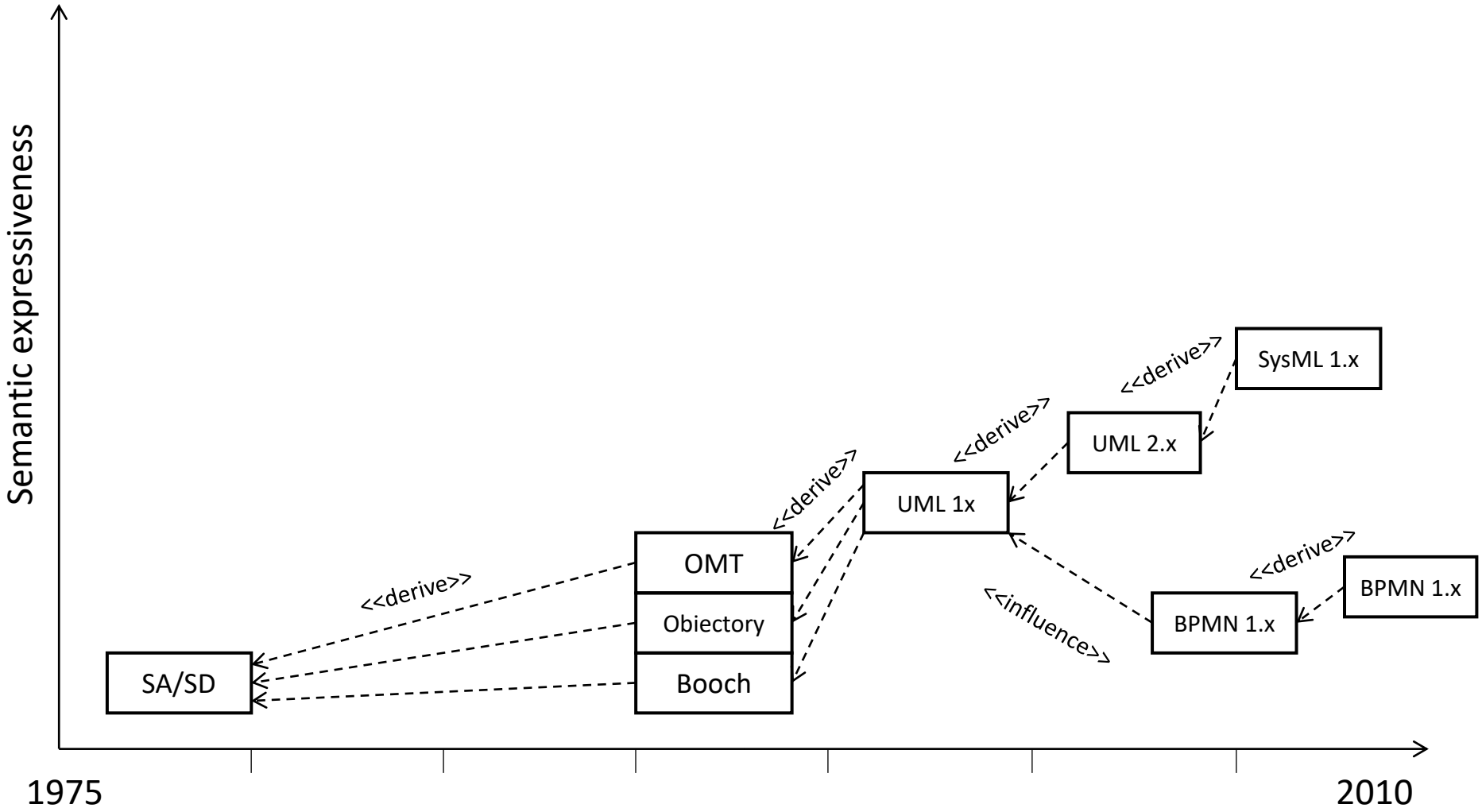
- 2003 – dokument RFP (request for proposal)

  robocza nazwa UML for System Engineering
- 2004 – wstępne założenia pod nazwą SysML
- 2006 – wersja 1.0

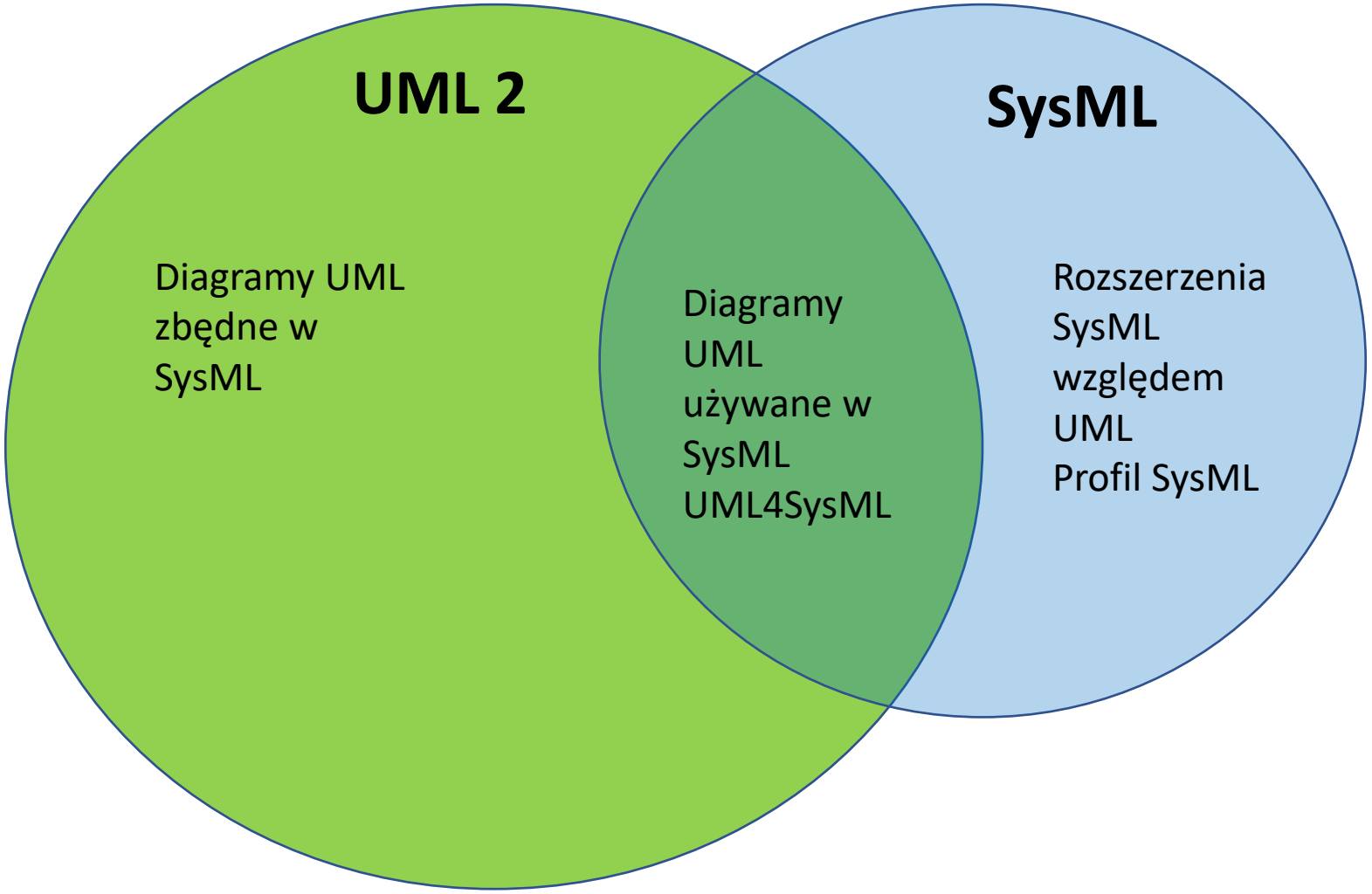
Prace nad udoskonaleniem języka zespół RTF (Revision Task Force)

- 2008 – wersja 1.1
- 2010 – wersja 1.2
- 2012 – wersja 1.3

Rozwój systemu



SysML vs. UML


UML 2
SysML
Diagramy UML zbędne w SysML
Diagramy UML używane w SysML
UML4SysML
Rozszerzenia SysML względem UML
Profil SysML

SysML vs. UML

Diagramy SysML :

- oryginalne dla SysML
- zmodyfikowane diagramy UML 2. SysML korzysta z podzbioru UML2.1, co zwane jest **UML4SysML**.
- diagramy przeniesione bezpośrednio z UML 2

SysML vs. UML

- **SysML** jest oparty na języku **UML**, więc nie całkowicie nowym językiem, traktowany jest jako zbiór dodatkowych uzupełnień do istniejącego rdzenia koncepcji i diagramów modelowania **UML**.
- **SysML** korzysta z podzbioru **UML2.1**, co zwane **jest UML4SysML**.
- Znaczna część koncepcji **UML** została odrzucona, ponieważ uznano je za nieodpowiednie dla modelowania w inżynierii systemów.

SysML vs. UML

- **SysML** odrzucił diagramy **UML**, takie jak diagram obiektu, komponentu, wdrożeniowe, komunikacji, przedziałów czasowych i interakcji
- zachowano diagram maszyny stanowej, sekwencji, diagram przypadku użycia
- inne diagramy zostały rozszerzone, tak jak diagram czynności
- dodano dwa nowe diagramy, czyli diagram wymagań i parametryczny

SysML vs. UML

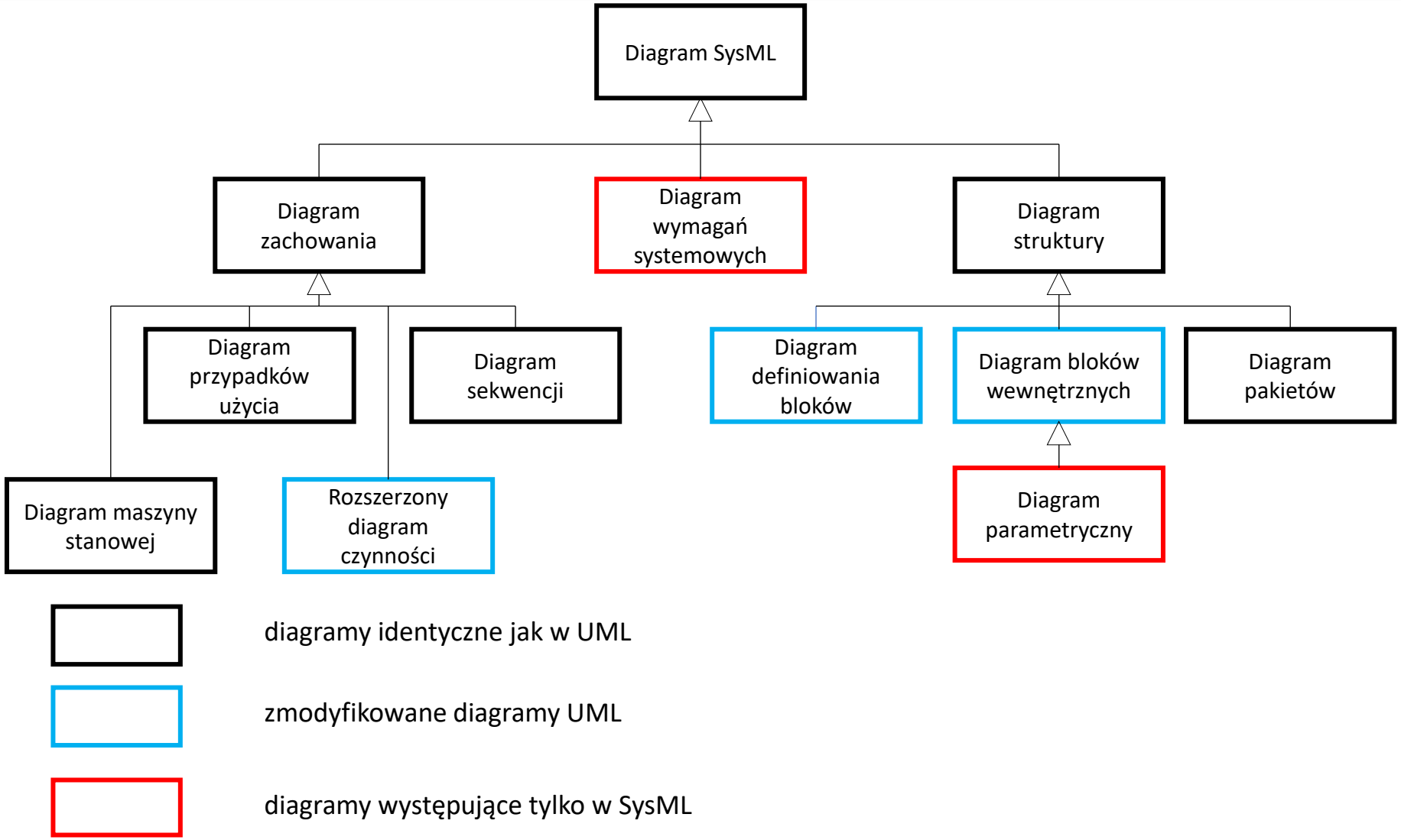
- semantyka bardziej elastyczna i ekspresyjna
- jest mniejszy niż UML - usunięcie konstrukcji dedykowanych dla oprogramowania
  – mniej diagramów
  – mniej konstrukcji
- wspieranie różnych typów rodzajów alokacji
  – UML – alokacje tabelaryczne
  – elastyczne tabele alokacji – alokacje wymagań, funkcjonalna, strukturalna
- konstrukcje zarządzania modelem wspierają modele, widoki, punkty widzenia
  – rozszerzają możliwości UML

Diagramy SysML

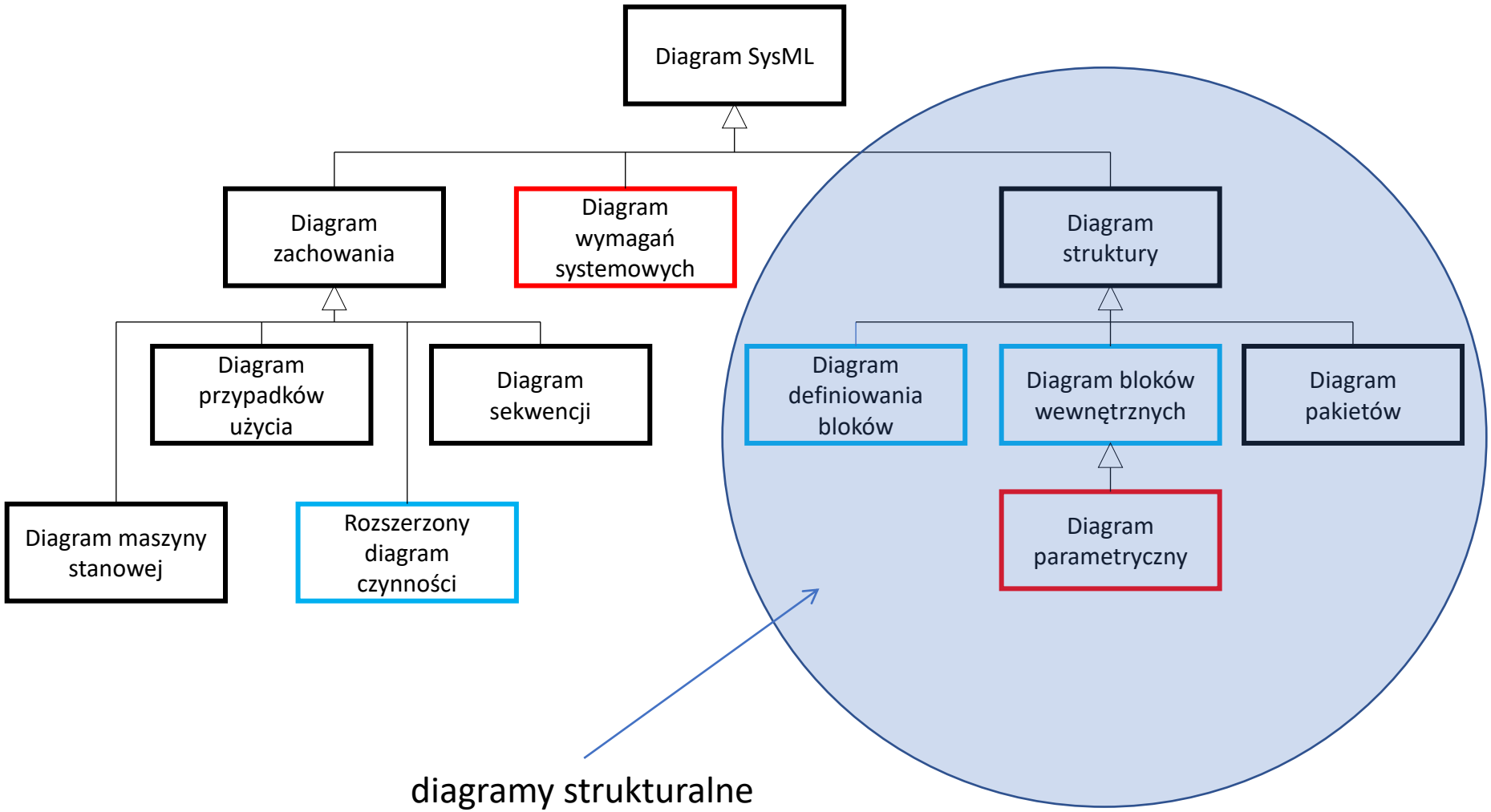
Diagram SysML składa się z 9 diagramów

- diagram wymagań systemowych
- diagram przypadków użycia
- rozszerzony diagram czynności
- diagram sekwencji
- diagram maszyny stanowej
- diagram definiowania bloków
- diagram bloków wewnętrznych
- diagram parametryczny
- diagram pakietów

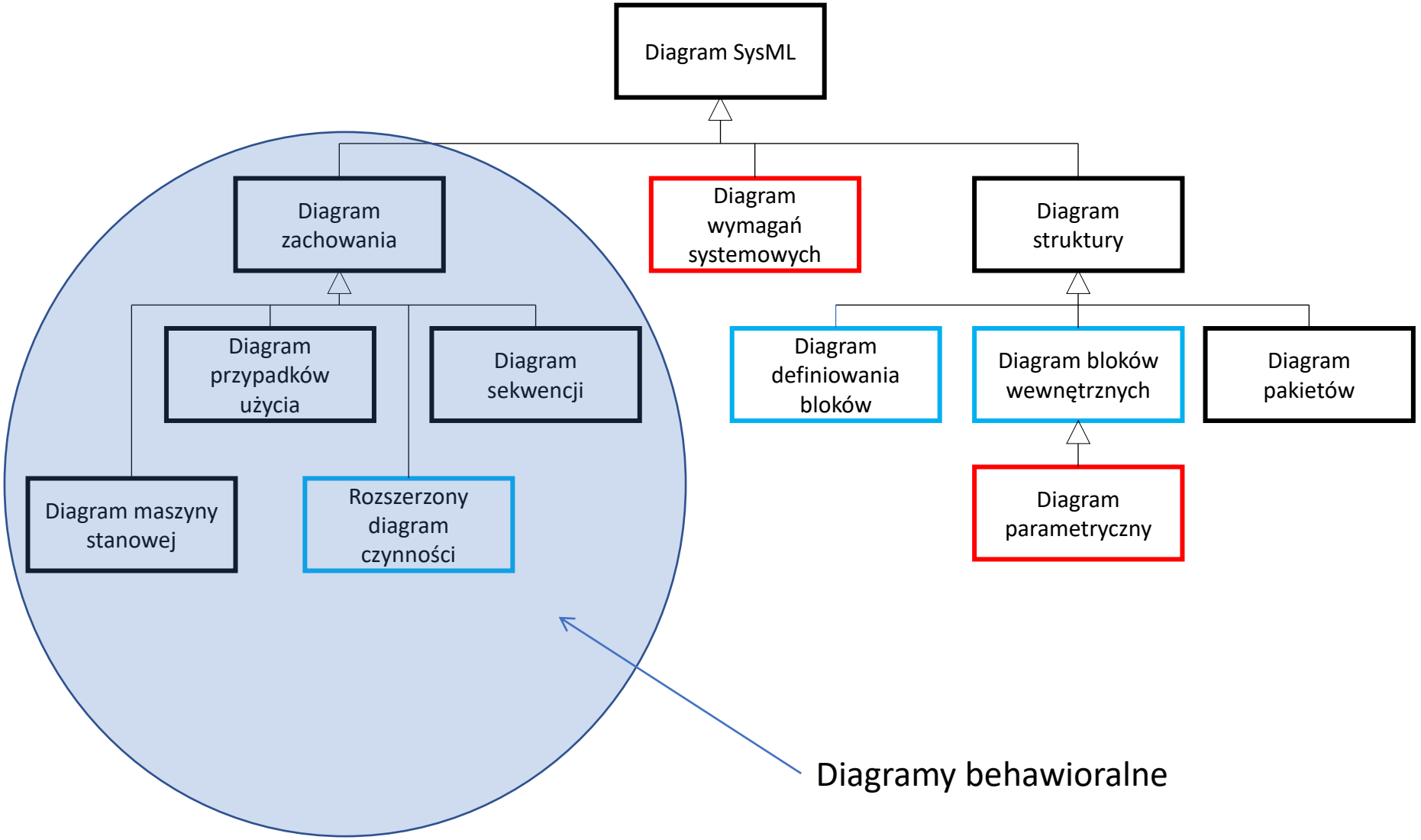
Hierarchia diagramów SysML



Hierarchia diagramów SysML



Hierarchia diagramów SysML



Diagramy języka SysML

- **diagram wymagań systemowych** – graficzne przedstawienie wymagań systemowych i ich relacji z innymi kategoriami modelowania (UML nie występuje)

  zastosowanie: inżynieria wymagań

- **diagram przypadków użycia** – graficzne przedstawienie przypadków użycia, aktorów oraz związków między nimi, występujących w danej dziedzinie (UML diagram przypadków użycia)

  zastosowanie: specyfikowanie precyzowanie wymagań funkcjonalnych

Diagramy języka SysML

- **rozszerzony diagram czynności** – graficzne przedstawienie sekwencyjnych i/lub współbieżnych przepływów sterowania oraz danych pomiędzy uporządkowanymi ciągami czynności, akcji oraz obiektów (UML diagram czynności)

  zastosowanie: analiza funkcjonalna

- **diagram sekwencji** – graficzne przedstawienie interakcji pomiędzy aktorami, blokami, częściami bloków obiektami w postaci sekwencji komunikatów wymienianych pomiędzy poszczególnymi kategoriami modelowania (UML diagram sekwencji)

  zastosowanie: analiza i projektowanie

Diagramy języka SysML

- **diagram maszyny stanowej** – graficzne przedstawienie dyskretnych, skokowych zachowań skończonych systemów stan – przejście (UML diagram maszyny stanowej)

  zastosowanie: projektowanie systemów, symulacje i generowanie szkieletowego kodu źródłowego

- **diagram definiowania bloków** – graficzne przedstawienie struktury systemu w postaci bloków, ich cech i związków (UML diagram klas)

  zastosowanie: analiza i projektowanie

Diagramy języka SysML

- **diagram bloków wewnętrznych** – graficzne przedstawienie wewnętrznej struktury bloku, wyrażanej poprzez wzajemnie powiązane części bloków (UML diagram struktur połączonych)

  zastosowanie: analiza i projektowanie

- **diagram parametryczny** – graficzne przedstawienie ograniczeń parametrycznych pomiędzy kategoriami modelowania struktury systemu (UML nie występuje)

  zastosowanie: analiza wydajnościowa i ilościowa

Diagramy języka SysML

- **diagram pakietów** – graficzne przedstawienie logicznej struktury systemu w postaci zestawu pakietów, połączonych zależnościami zagnieżdżania (UML diagram pakietów)

  zastosowanie: zarządzanie modelami

Diagramy języka SysML

Diagramy UML nie mające zastosowania w SysML

- diagram obiektów
- diagram komunikacji
- diagram harmonografowania
- diagram sterowania interakcją
- diagram komponentów
- diagram rozlokowania

Diagramy języka SysML

- Każdy diagram SysML reprezentuje element modelu.
- Każdy diagram SysML musi mieć swoją ramkę.
- Kontekst diagramu określa nagłówek:
  - rodzaj diagramu (act, bdd, ibd, sd ….)
  - rodzaj elementu modelu (package, block, activity ….)
  - nazwa elementu modelu
  - nazwa diagramu lub widoku
- Oddzielny blok opisu diagramu pozwala wskazać, czy diagram jest kompletny, czy też jego elementy zostały pominięte

Diagramy SysML – frame

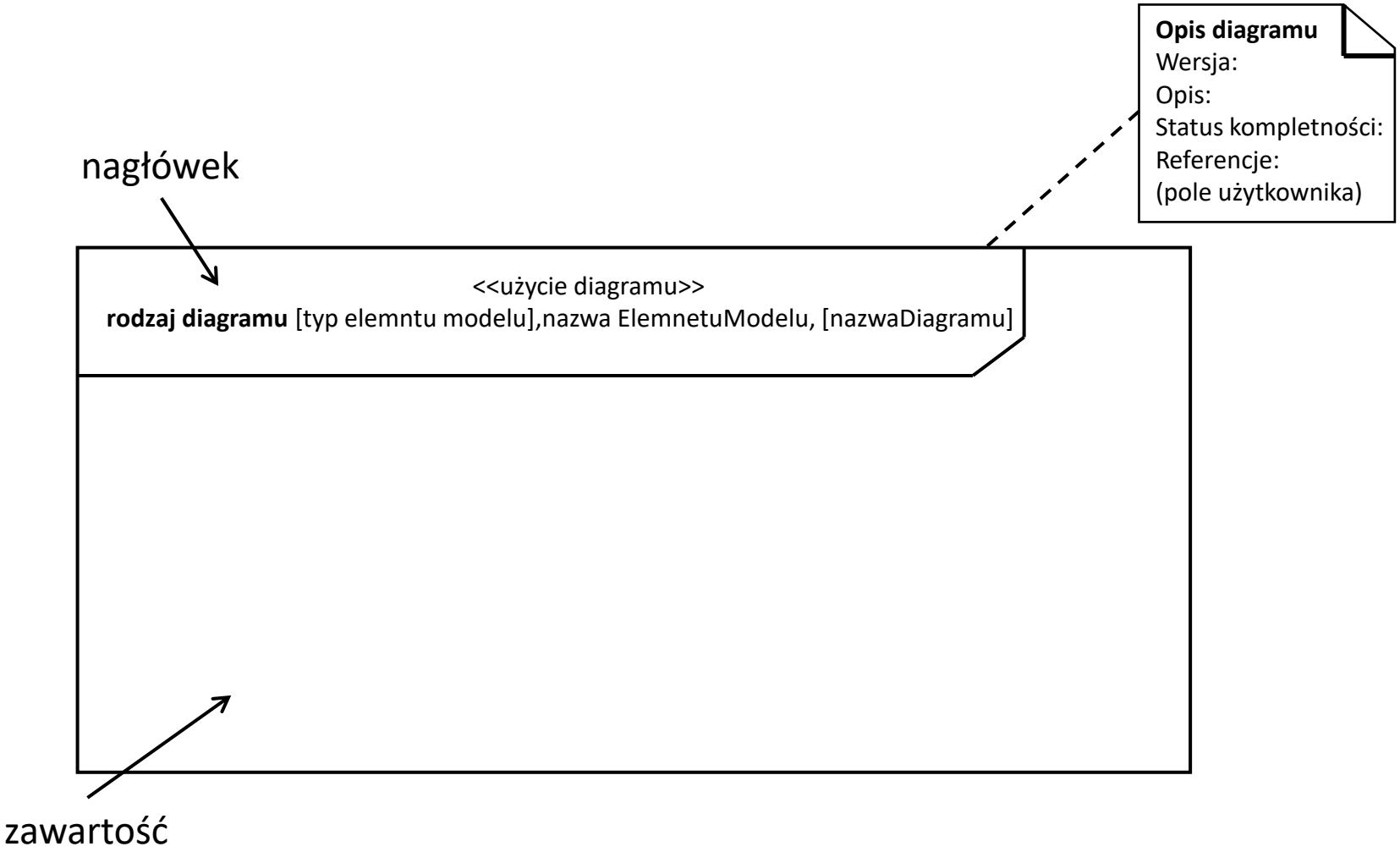


Diagram wymagań systemowych (req)

SysML wprowadza **diagram wymagań**, którego wcześniej nie było w UML. Ten nowy diagram zapewnia środki, by wskazać wymagania i związać je z innymi modelami specyfikacji, projektowania i weryfikacji. Wymagania mogą być prezentowane w formacie graficznym, tabularycznym lub struktury drzewiastej.

Wymagania

W SysML **wymaganie** przedstawia cechę, własność, zachowanie systemu:
• to inaczej rzeczy, które opisują co system (traktowany jako czarna skrzynka) ma robić, ale nie jak ma to robić;
• to warunki, które system musi spełniać lub cechy , które musi wykazywać (zgodne z wymaganiami zamawiających system).

• Zadanie definiowania i listowania wymagań jest wykonane w pierwszych krokach procesu projektowania systemu.
• Wymagania pozwalają projektantowi jasno ustalić czego się oczekuje od przyszłego systemu. Ponadto, wymagania tworzą punkt centralny procesu weryfikacji i walidacji.

Klasyfikacja wymagań

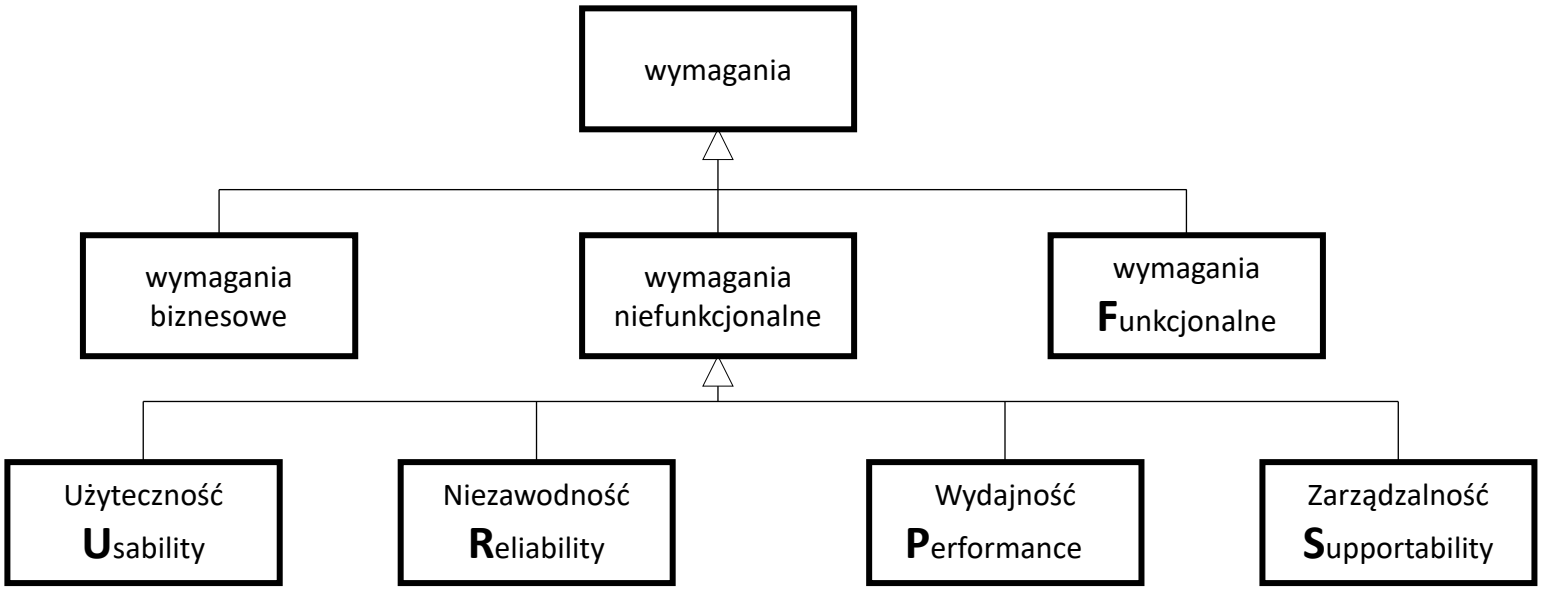
Wymagania funkcjonalne – zachowanie systemu (jakie akcje ma wykonywać system), są utożsamiane z wymaganiami użytkownika.

Wymagania niefunkcjonalne – ograniczenia, które mają wpływ na wykonywane zadania systemu. Reprezentują zalecenia, które ograniczają w pewien sposób inne wymagania. Przykłady wymagań poza funkcjonalnych obejmują wymagania jakości, wymagania wdrożeniowe, wymagania zastosowania specyficznych rozwiązań.

Wymagania biznesowe – ograniczenia dotyczące projektowania systemu, nie mające wpływu na jego zachowanie, wynikające ze zobowiązań technicznych, ekonomicznych oraz umowy.

Wymagania funkcjonalne i niefunkcjonalne ujęte są w modelu **FURPS**

Model FURPS

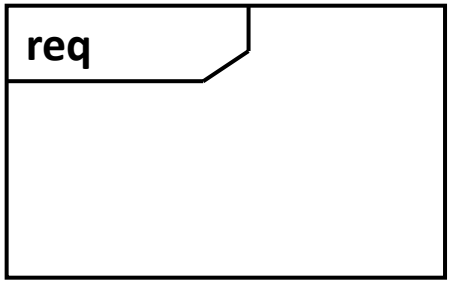

wymagania
wymagania biznesowe
wymagania niefunkcjonalne
wymagania Funkcjonalne
Użyteczność Usability
Niezawodność Reliability
Wydajność Performance
Zarządzalność Supportability

Wymagania niefunkcjonalne

- **Użyteczność**: wymagany czas szkolenia, czas wykonania poszczególnych zadań, ergonomia interfejsu, pomoc, dokumentacja użytkownika …
- **Dostępność**: średni czas międzyawaryjny (MTBF), średni czas naprawy (MTTR), dokładność, maksymalna liczba błędów …
- **Wydajność**: czas odpowiedzi, przepustowość, konsumpcja zasobów, pojemność …
- **Zarządzalność**: łatwość modyfikowania, skalowalność, weryfikowalność, kompatybilność, możliwości konfiguracyjne, możliwości serwisowe, przenośność …

Diagram wymagań

Definiuje i organizuje wymagania systemowe



Przedstawia wymagania i ich relacje z innymi wymaganiami, elementami projektu, przypadkami testowymi dla wspomagania śledzenia wymagań

Diagram wymagań

Podstawowe atrybuty wymagania:

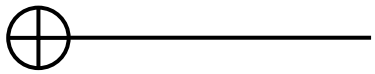
- Unikalny identyfikator wymagania (id): ze względu na hierarchiczny charakter systemów i samych wymagań, stosuje się także hierarchiczną numerację wymagań (1, 1.1, 1.2, 2, 2.1, 2.2 …)
- treść wymagania/opis (text): opis tego, co jest wymagane.

<<requirement>> przechowywanie parametrów automatycznego zamawiania w kartach magazynowych
Text: „System powinien umożliwiać podawanie minimalnego poziomu magazynowego, maksymalnego poziomu magazynowego, poziomu odnowienia, ilości zamawianej oraz współczynnik tolerowanego opóźnienia” Id: 1.1

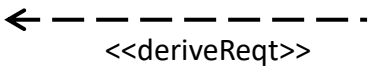
Zawiązki między wymaganiami

Poszczególne wymagania łączą się z innymi logicznie za pomocą różnych typów związków (ang. *relationships*):

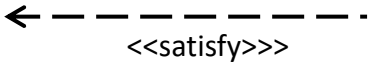
- **zagnieżdżenia**



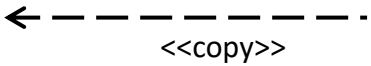
- **wyprowadzania**



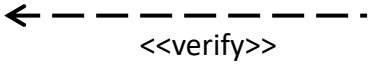
- **realizacji**



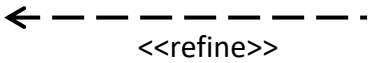
- **powielania**



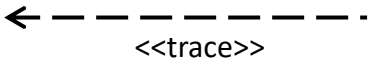
- **weryfikowania**



- **precyzowania**



- **śledzenia**



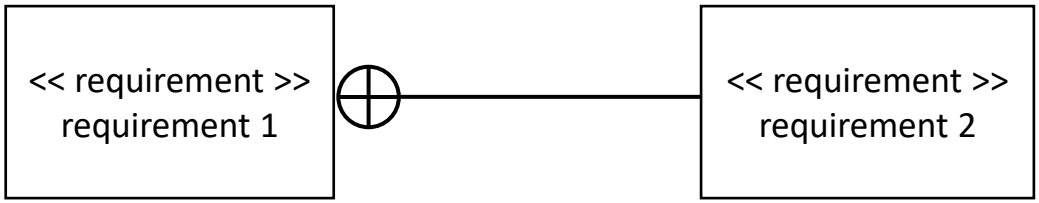
Zagnieżdżenie

Wymagania w diagramach wymagań systemowych języka SysML łączy się przez związki **zagnieżdżenia** (ang. *containment*) umożliwiające tworzenie wielopoziomowej hierarchii wymagań oraz przez **zależności** (ang. *dependencies*). Wskazują one na charakter logicznej zależności między poszczególnymi wymaganiami.

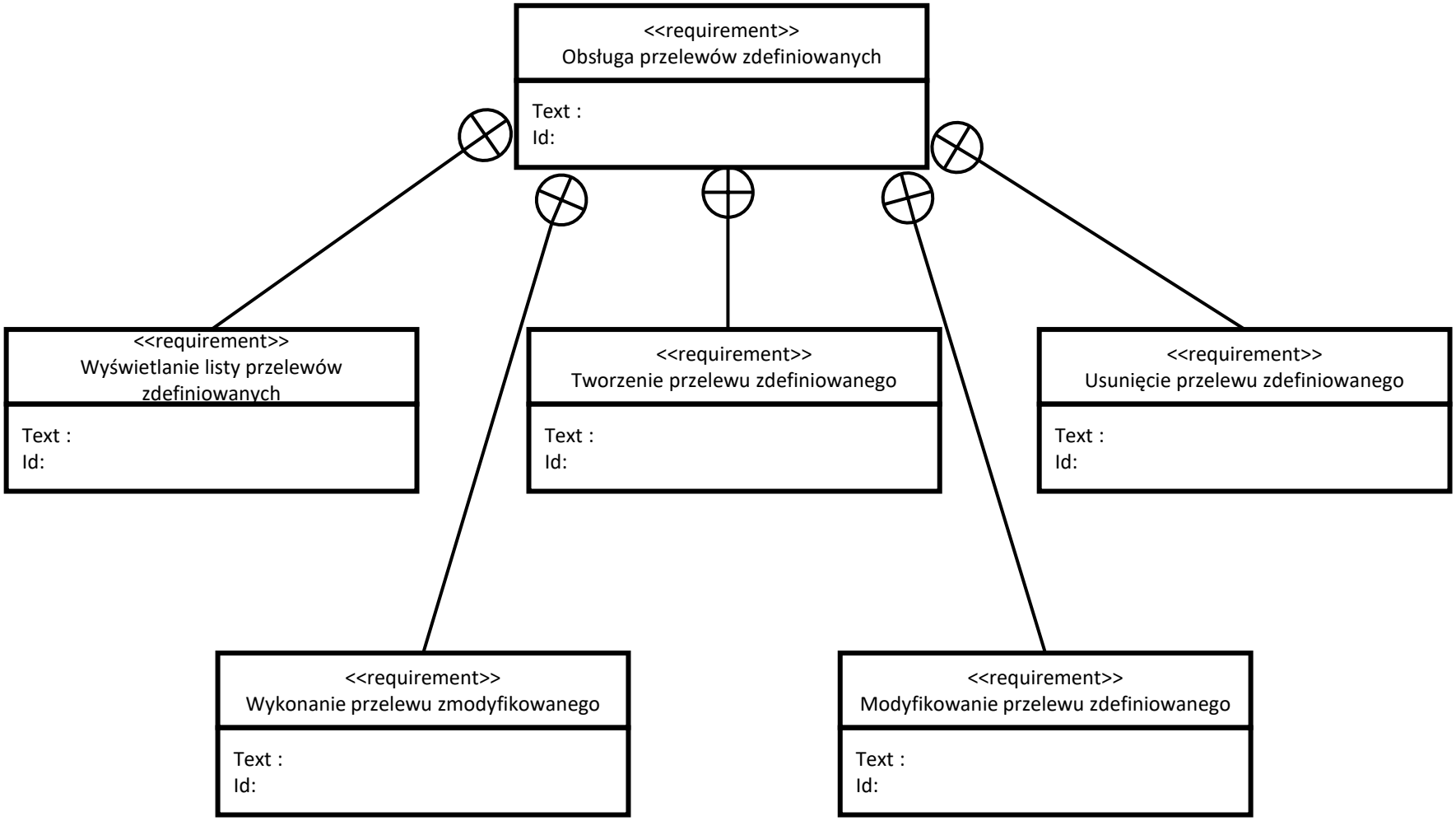
Zagnieżdżenie

Zagnieżdżenie do służy budowania hierarchii wymagań

- wymaganie może mieć tylko jeden element nadrzędny
- wielokrotne użycie wymagania możliwe jest dzięki zależności powielania (<<copy>>)
- wymaganie nadrzędne uznaje się za spełnione wtedy i tyko wtedy jeżeli wszystkie wymagania podrzędne są spełnione
- jest to najczęściej wykorzystywana zależność



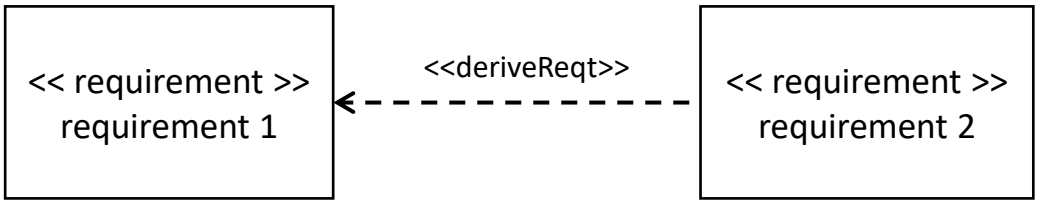
Zagnieżdżenie – przykłady



Wyprowadzenie <<deriveReqt>>

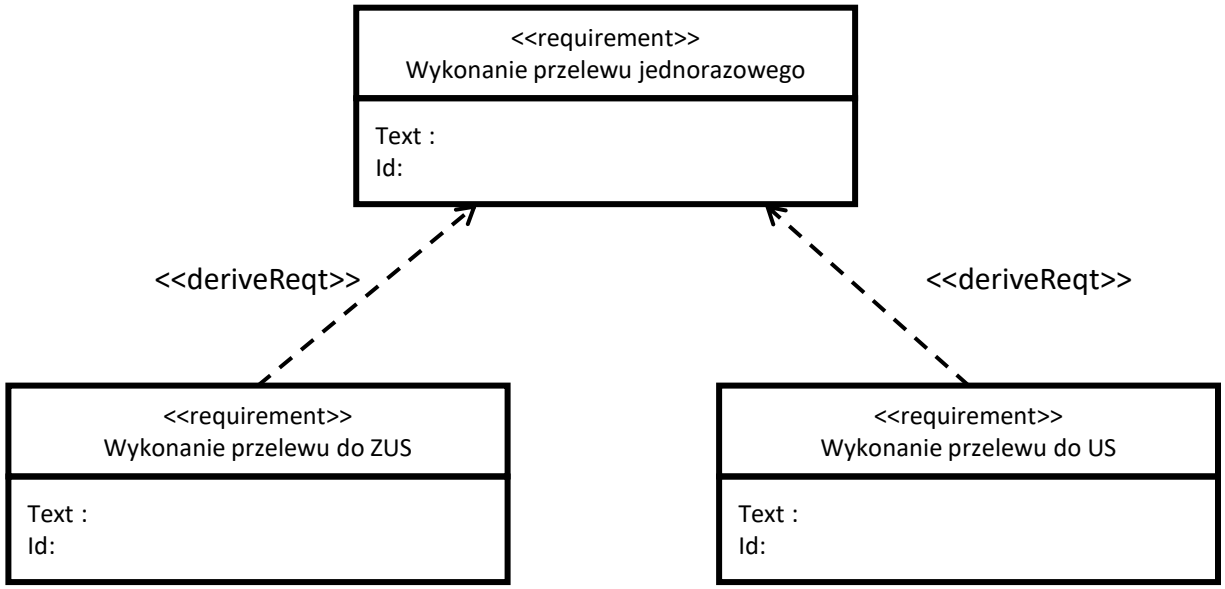
Wyprowadzanie umożliwia wyprowadzić wymaganie pochodnego z wymagania nadrzędnego :

- cechy systemu reprezentowane przez wymaganie pochodne są pochodnymi cech wymagania nadrzędnego
- wymagań docelowych może być kilka
- relacja wyprowadzania może istnieć tylko między wymaganiami.
- strzałka wskazuje wymaganie, z którego wyprowadzono wymaganie pochodne.



Wyprowadzenie <<deriveReqt>>

- Wyprowadzanie jest bardziej uniwersalne od zagnieżdżania.
- Może specyfikować związki między wymaganiami w różnych gałęzia chhierarchii, także na tym samym poziomie hierarchii.



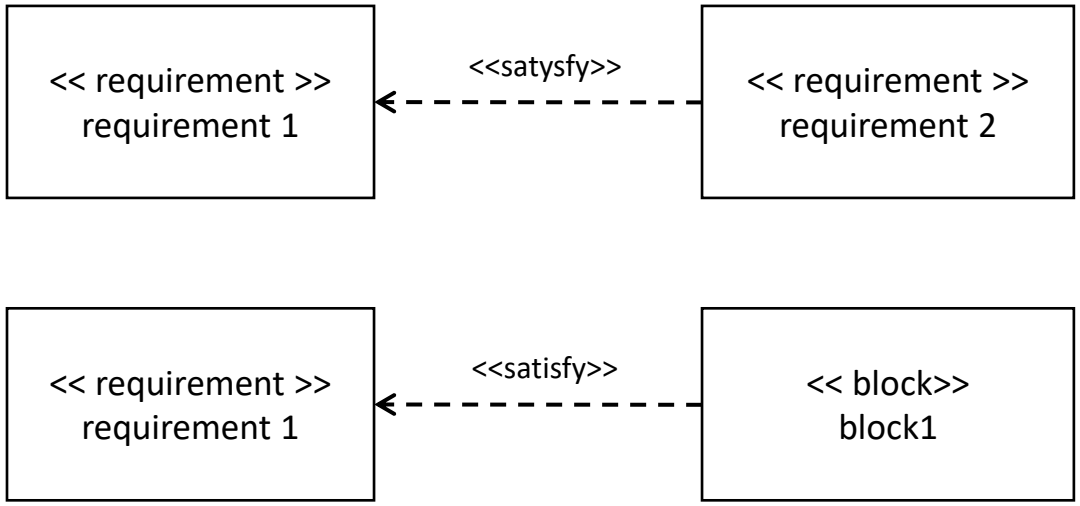
Realizacja <<satisfy>>

Realizacja oznacza spełnienie określonego wymagania.

Wymaganie może być spełnione przez:

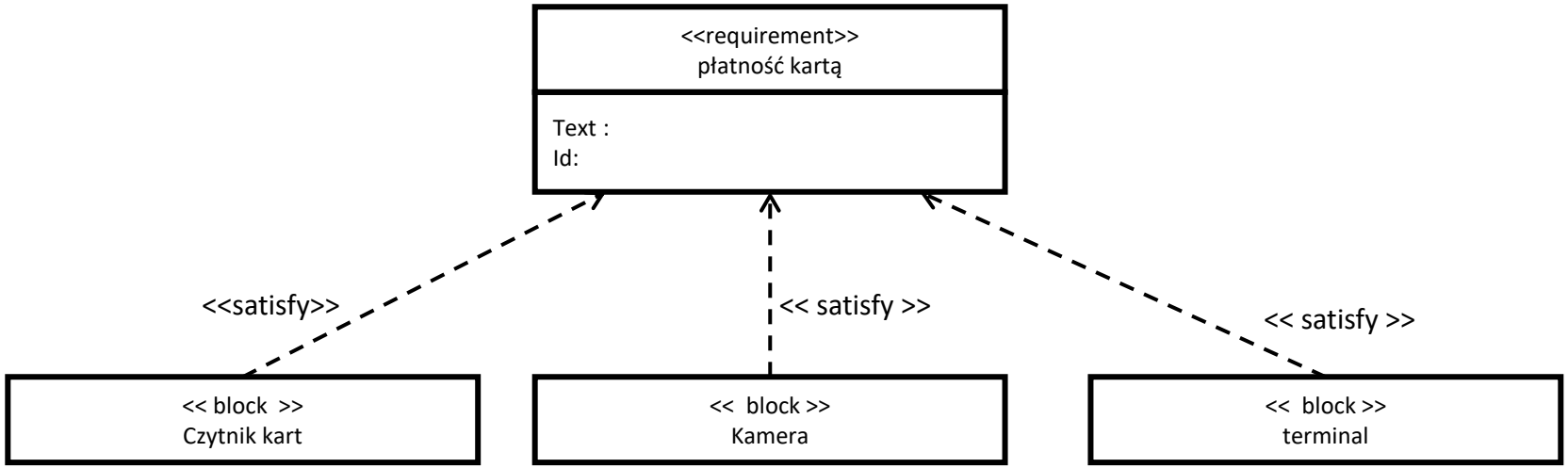
- inne wymagania
- inne elementy modelu (programowe lub sprzętu) pokazywane na diagramach jako bloki

Strzałka wskazuje spełniane wymaganie



Realizacja <<satisfy>> – przykład

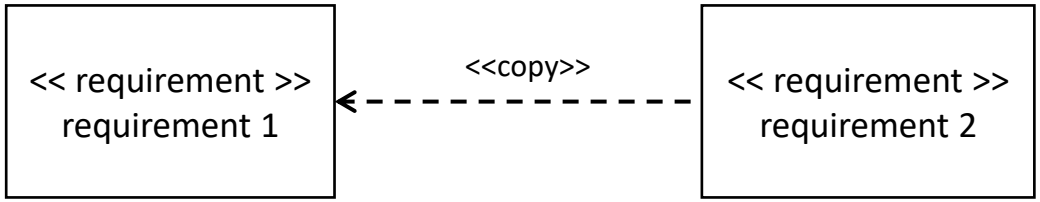
- Zadaniem projektanta systemu jest identyfikacja kluczowych elementów niezbędnych do spełnienia wszystkich wymagań oraz wskazanie, które wymagania są przez nie spełniane.
- Realizacja pozwala określić skutki zmian w obrębie wymagania wobec elementów od niego zależnych.



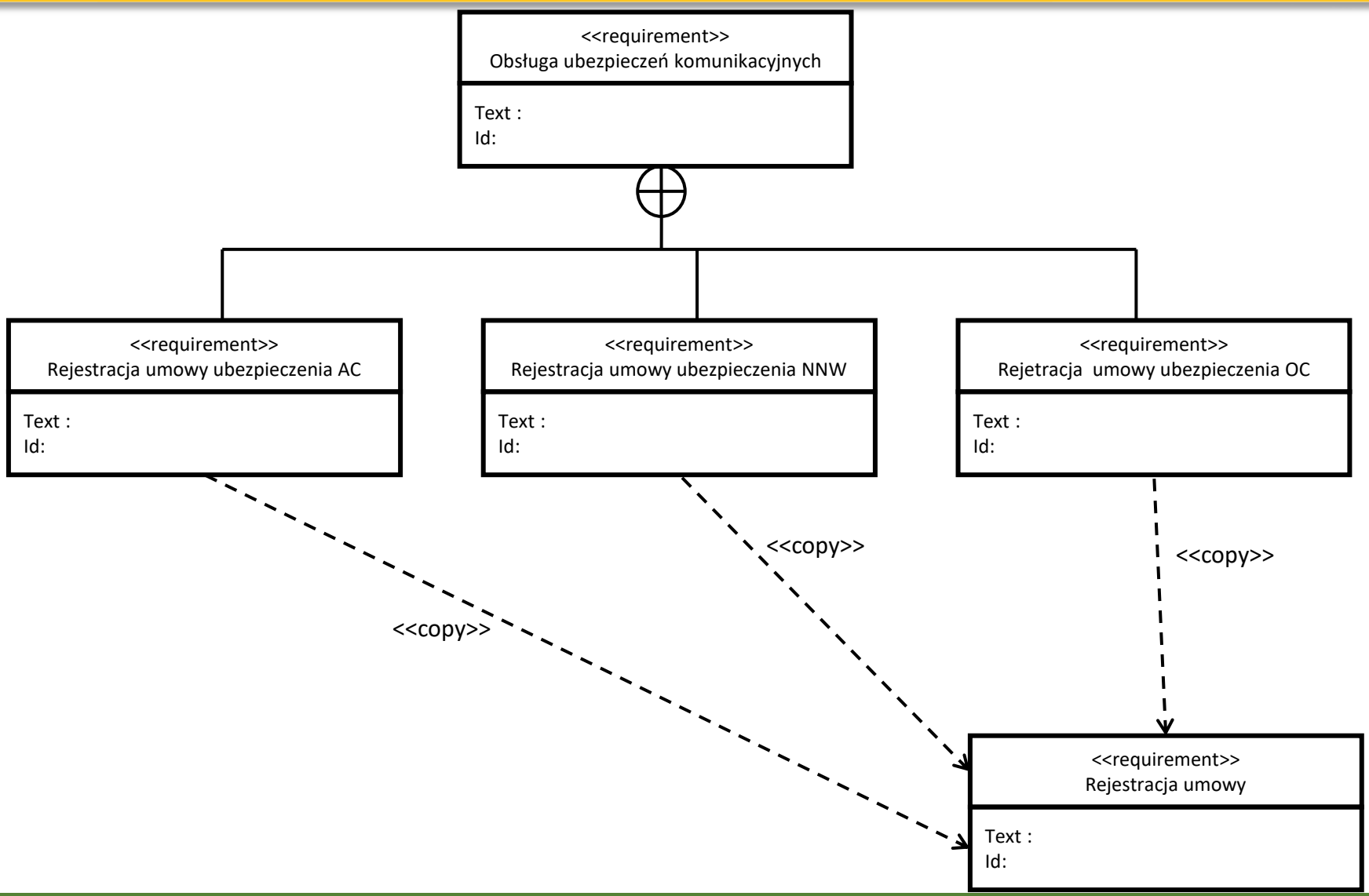
Powielanie <<copy>>

Powielanie pozwala uniknąć wielokrotnego definiowania tego samego wymagania:

- przyjmuje się, że wymaganie docelowe można tylko odczytywać (przejmuje ono treść wymagania powielanego),
- zmiany w wymaganiu powielanym automatycznie przechodzą do wymagań docelowych,
- numery porządkowe i nazwy wymagań mogą się różnić,
- strzałka wskazuje powielane wymaganie.



Powielanie <<copy>> – przykład



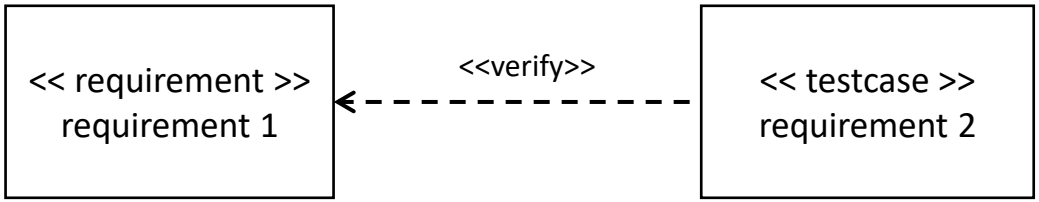
Weryfikowanie <<verify>>

Weryfikowanie określa w jaki sposób sprawdzić czy wymaganie zostało prawidłowo zrealizowane.

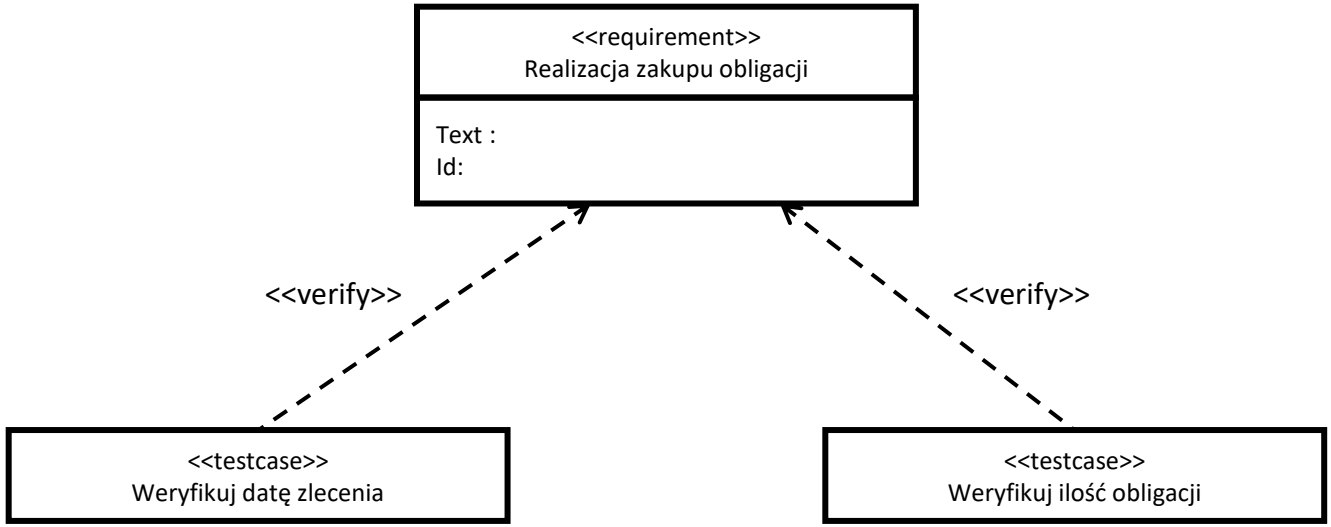
W SysML za weryfikowanie odpowiadają testy formalne reprezentowane przez testowe przypadki użycia.

- większość wymagań ma więcej niż jeden przypadek testowy
- każdy przypadek użycia może być udokumentowany dodatkowymi diagramami (np. sekwencji lub aktywności)
- przypadki testowe mogą mieć dodatkowe stereotypy:

  <<analyze>>, <<inspect>>, <<demonstrate>>, <<test>>



Weryfikowanie <<verify>> – przykład

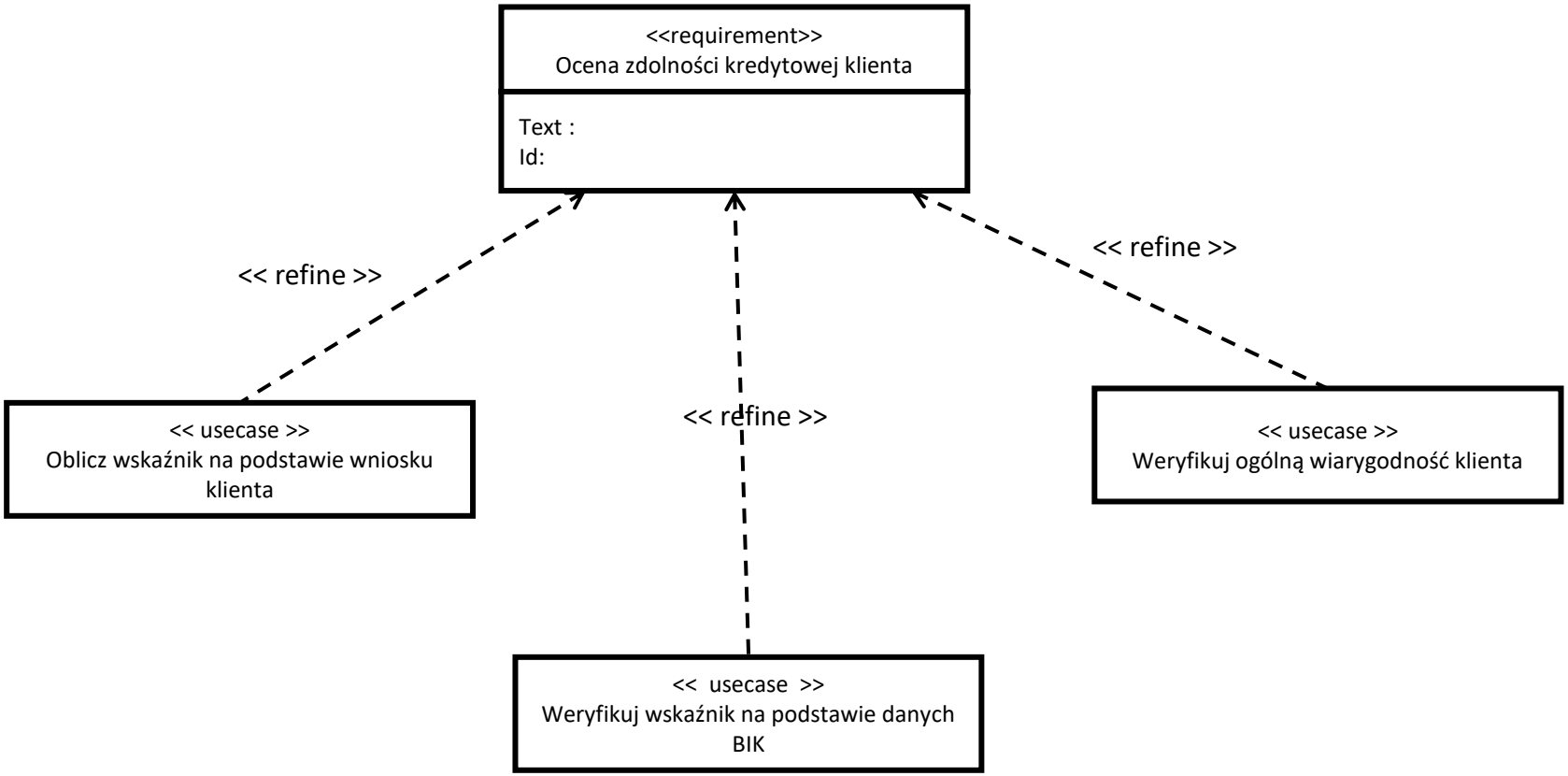


Precyzowanie <<refine>>

Precyzowanie pozwala na uściślenie sensu wymagania poprzez dodanie do diagramu detali:

- kategorii modelowania SysML
- projektowania interfejsu
- specyfikacji tekstowych
- standardów

Precyzowanie <<refine>> – przykład



Śledzenie <<trace>>

Śledzenie umożliwia przedstawienie nieformalnego związku między wymaganiem a dowolnym elementem modelu, ustalenie kolejności realizacji wymagań.

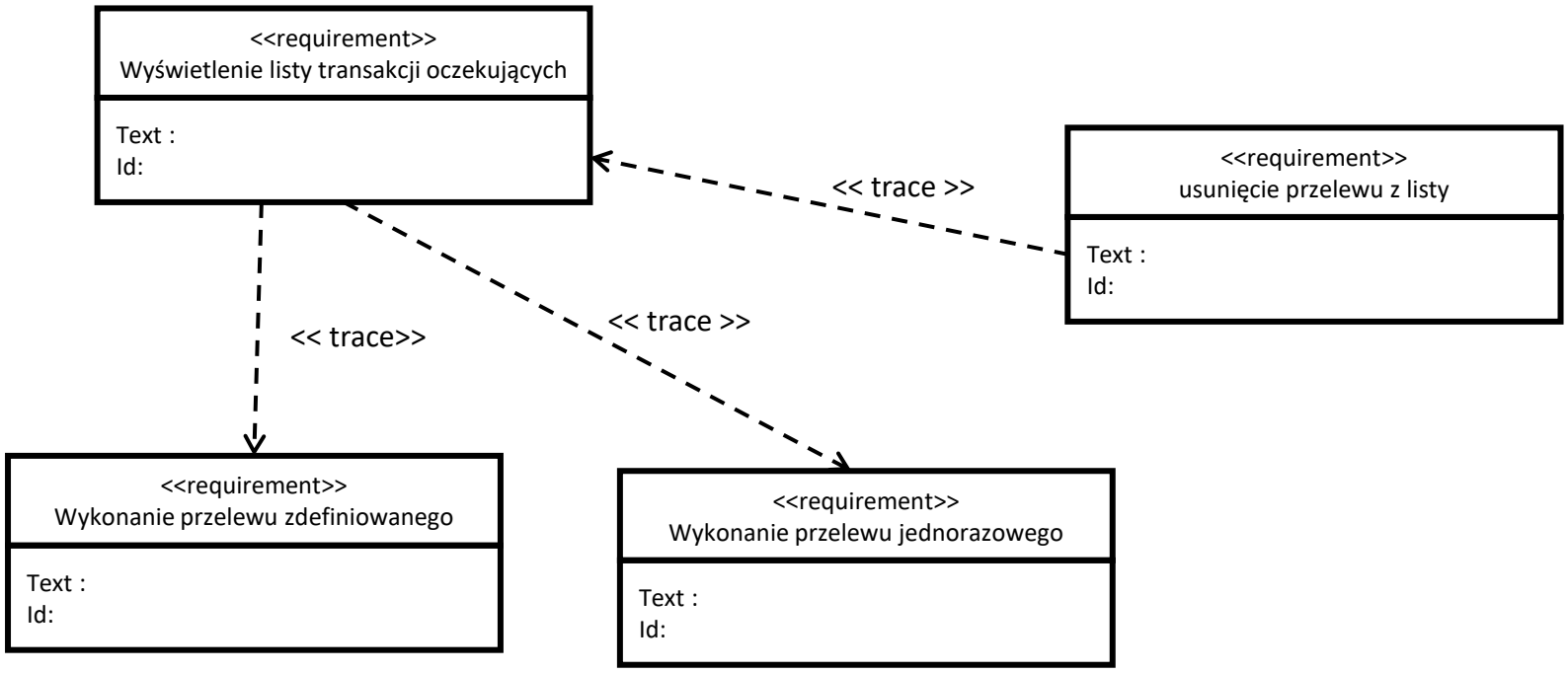


Diagram wymagań – specyfikowanie

Do specyfikowania wymagań i związków pomiędzy wymaganiami w SysML można stosować:

- tabelaryczną specyfikację wymagań,
- tabelaryczną specyfikację związków,
- rozszerzone wymagania systemowe,
- stereotypowanie rozszerzonych wymagań systemowych.

Postaci tabelaryczne wymagań

Tabelaryczna specyfikacja wymagań

numer porządkowy	nazwa	treść
1.1	nazwa1	System powinien …

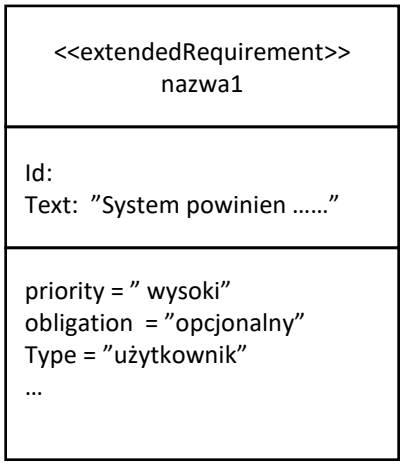
Tabelaryczna specyfikacja związków

Wymaganie źródłowe		Rodzaj związku	Wymaganie docelowe	
Id	nazwa		Id	nazwa
1.1	nazwa1	zagnieżdżanie	1.3	nazwa2

Rozszerzone wymagania systemowe

Dodatkowe właściwości wymagania:

- **priority** priorytet – kolejność implementacji
- **obligation** istotność – ewentualna opjonalność
- **stability** stabilność – prawdopodobieństwo zmiany
- **type** typ – zródło wymagania
- **risk** ryzyko – ewentualne zagrożenia



Stereotypowanie wymagań

Typy wymagań rozszerzonych często uściśla się poprzez nadania im dodatkowego stereotypu:

- wymagania funkcjonalne << **functionalRequirement**>>
- wymagania interfejsowe << **interfaceRequirement**>>
- wymagania wydajnościowe << **performance Requirement**>>
- wymagania fizyczne << **physicalRequirement**>>
- ograniczenia projektowe << **designConstraint**>>

Diagram definiowania bloków (bdd)

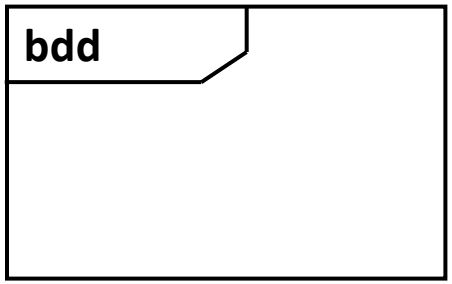
bdd w SysML – to rozwinięcie **diagramu klas** w UML.

Diagramy definiowania bloków służą do precyzyjnej charakterystyki systemu. SysML wykorzystuje koncepcję bloków w celu określenia hierarchii i wzajemnych połączeń w ramach projektu systemowego.

Diagram definiowania bloków bdd jest graficznym przedstawieniem struktury systemu w postaci bloków, ich cech i związków.

Podstawowe kategorie modelowania **bdd** obejmują:

- Blok (**block**)
- Związek (**relationship**)
- Typ wartości (**value type**)
- Aktora (**actor**)
- Port (**port**)
- Pakiet (**package**)



Blok

Blok w SysML – to rozszerzenie **klasy** znanej z UML.

Blok jest to podstawowa jednostka, element strukturalny, który opisuje strukturę systemu lub elementu. Blokiem może być komponent systemu, część sprzętu, oprogramowania, komponent danych, osoba, usługa.

Blok – charakterystyka

Bloki posiadają

- unikatową tożsamość
- zespół cech
- zestaw opcjonalnych sekcji (compartments)

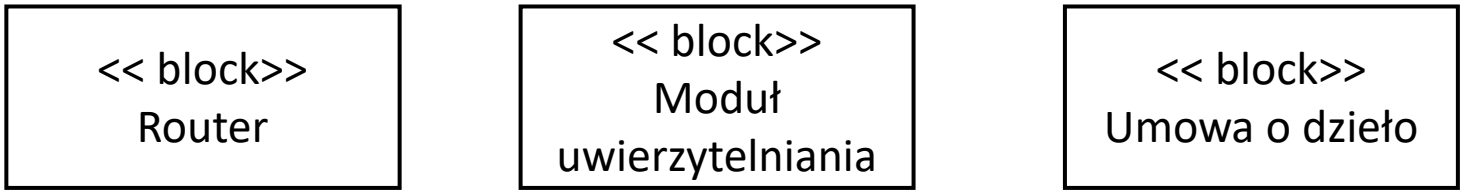
Bloki łączą się za pomocą związków takich jak w UML

- asocjacji
- generalizacji
- zależności
- realizacji
- zagnieżdżenia

Blok – przykłady

Blok może mieć charakter:

- sprzętowy – reprezentują istniejące urządzenie
- programowy – reprezentują zasoby danych, moduły oprogramowania lub usługi
- organizacyjny – reprezentują jednostki organizacyjne, procedury, dokumenty,



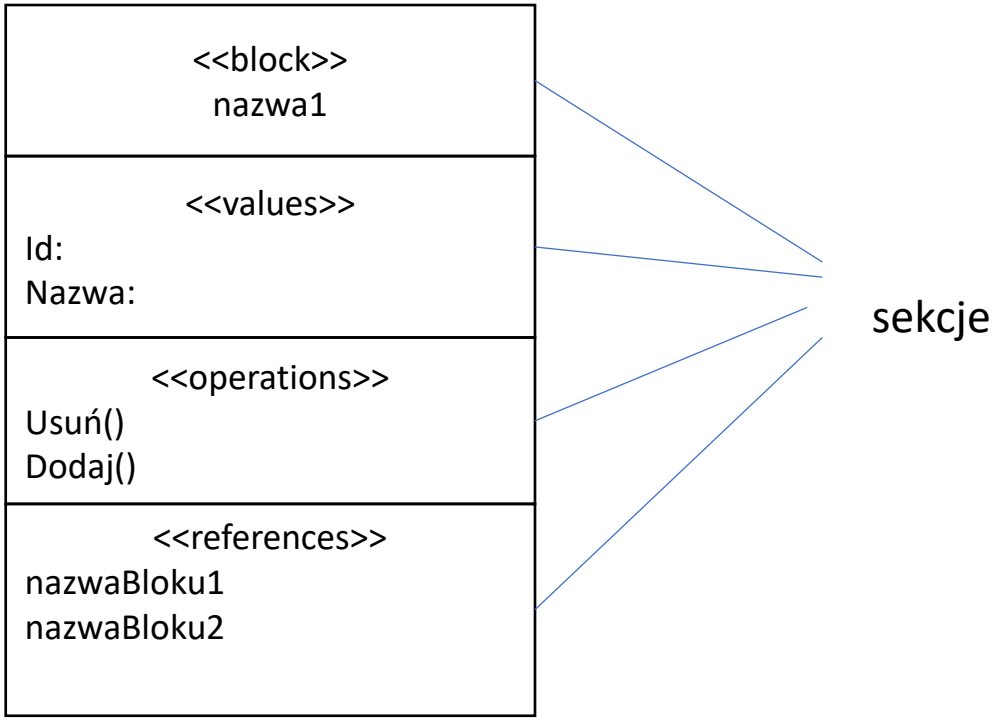
Cechy bloku

Bloki definiują zestaw wspólnych cech pierwotnych dla instancji danego bloku.

- wartości (values) – atrybuty bloku
- operacje (operations) – operacje jakie można wykonywać na bloku
- ograniczenia (constraints)– ograniczenia dotyczące atrybutów, maksymalny czas złożenia zamówienia
- części (parts) – wyszczególnienie elementów składowych bloku (dekompozycja)
- odniesienia (references) – lista bloków powiązanych z danym blokiem
- cechy uniwersalne (properties) – pozostałe cechy
- nie stosuje się znaczników widoczności (+, -, #) jak w UML, przyjmuje się dla wszystkich cech poziom widoczności publiczny

Sekcje bloku

Każdy blok na diagramie musi posiadać co najmniej jedną sekcję – sekcję nazwy bloku.



Związki

Związki stosowane w SysML:

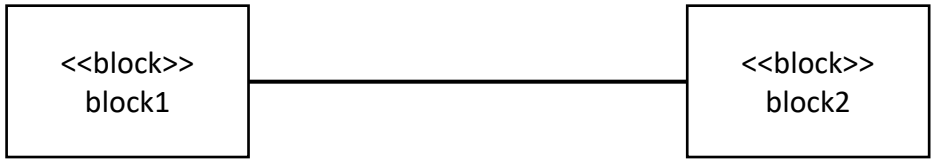
- **asocjacja**
- **generalizacja**
- **zależność**
- **realizacja**
- **zagnieżdżenie**

Różnice w stosunku do UML:

- zrezygnowano z asocjacji n-arnych (blok może być umieszczony w strukturze innych bloków),
- wyłączono cechy asocjacji takie jak kwalifikacja i zaawansowane aspekty nawigacji,
- bardziej intensywne zastosowanie związków zagnieżdżania (w UML głownie w diagramach pakietów).

Asocjacja

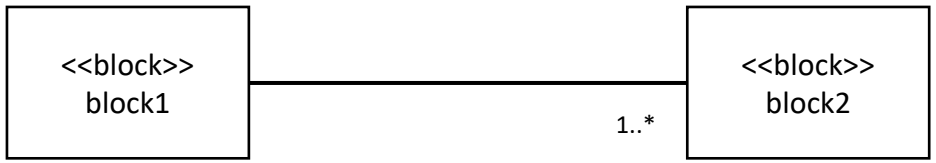
asocjacja



Opisuje związek pomiędzy dwoma blokami.

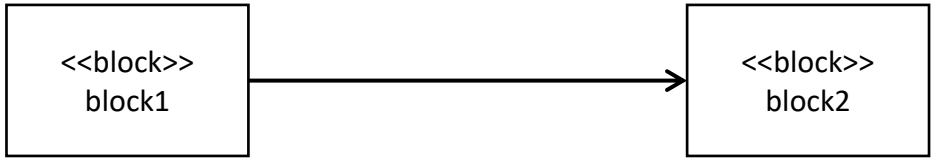
Cechy asocjacji:

- **nazwa**
- **rola** jest odgrywana przez dany blok w stosunku do innego i na odwrót
- **liczebność** określenie liczby instancji danego bloku z pojedynczą instancją powiązanego z nim bloku

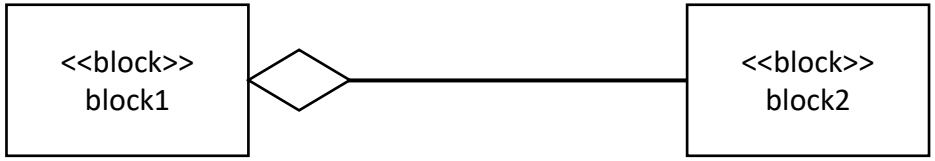


Asocjacja

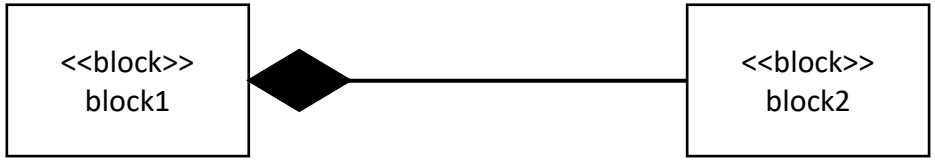
- **nawigacja –** kierunek przesyłania informacji w ramach asocjacji tzw. asocjacja skierowana (jednokierunkowa)



- **agregacja** związek całość – część pomiędzy blokami

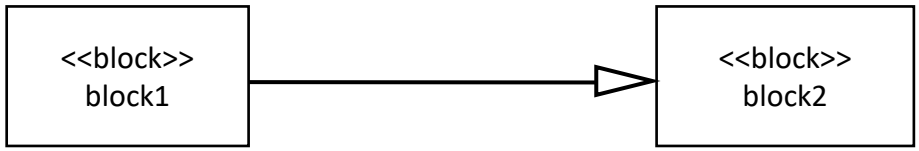


- **kompozycja –** jest formą agregacji, długość życia części jest zgodny z długością życia całości



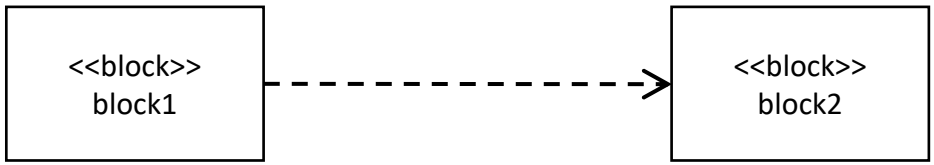
Asocjacja

generalizacja



Specyfikuje związek o charakterze taksonomicznym pomiędzy ogólną a specjalizowaną kategorią modelowania. Związek pomiędzy elementem bardziej ogólnym a elementem bardziej wyspecyfikowanym.

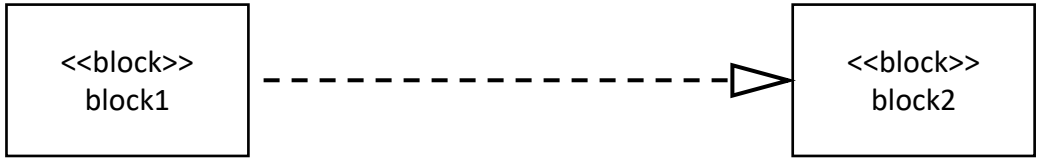
zależność



Wskazuje związek pomiędzy dwoma kategoriami modelowania, w którym zmiana jednego z nich (niezależnego) wpływa na drugi (zależny) W szczególności wskazuje na sytuację, kiedy dany blok wymaga funkcjonalności oferowanej przez powiązany interfejs.

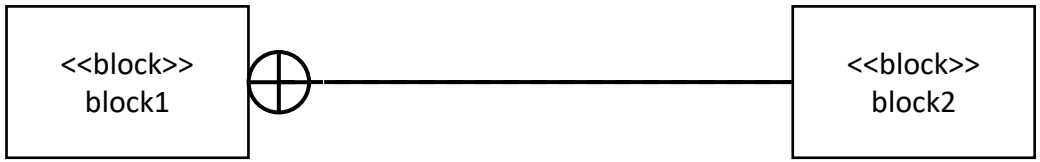
Asocjacja

realizacja



Związek znaczeniowy między elementami, w którym jeden określa kontrakt, a drugi zapewnia wywiązanie się z niego. Na diagramach **bdd** jest to szczególny rodzaj zależności, w której dany blok definiuje i oferuje funkcjonalność wskazaną przez powiązany interfejs.

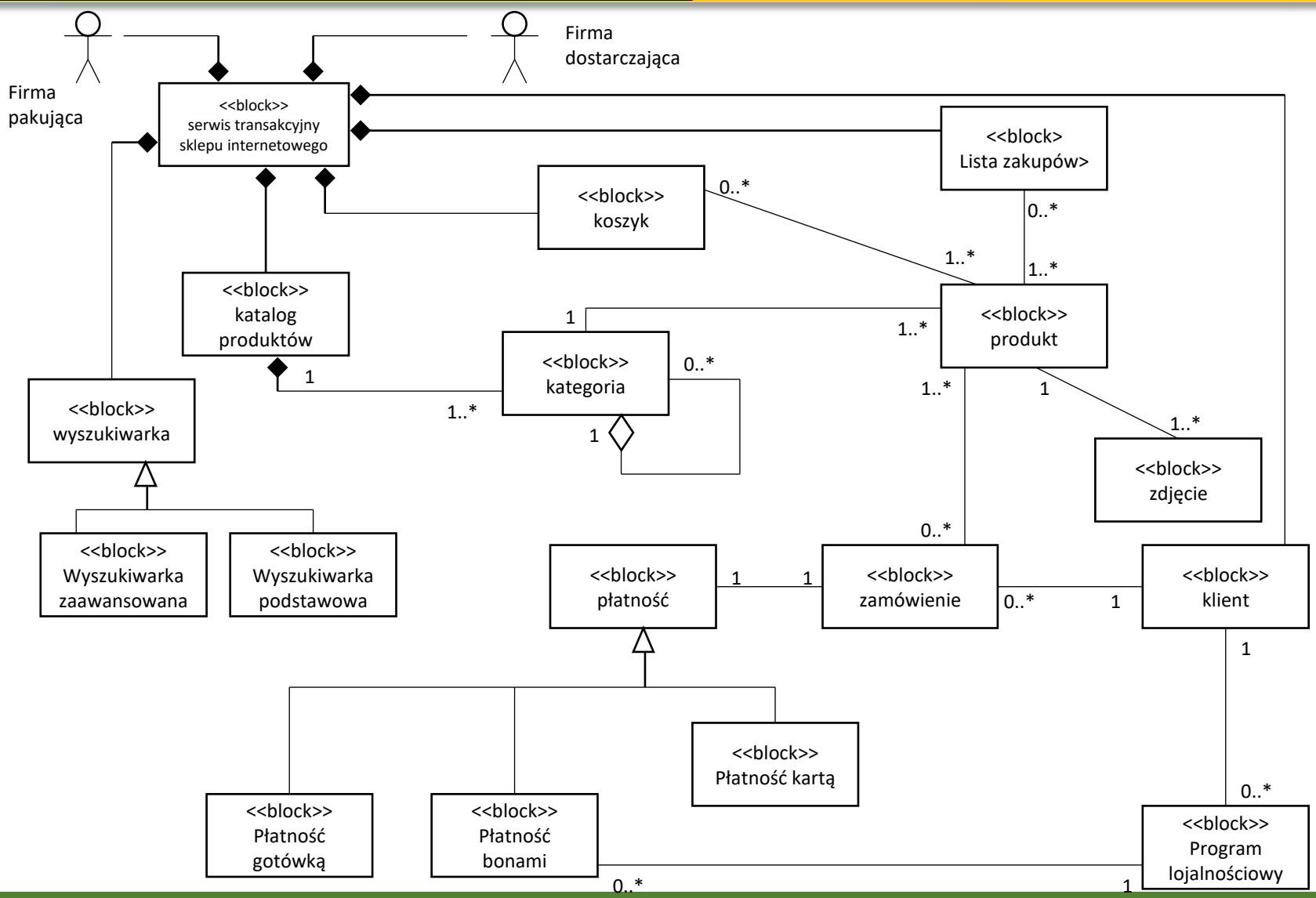
zagnieżdżanie



Związek łączący blok nadrzędny z podrzędnym, dzięki czemu możliwe jest zbudowanie wielopoziomowej hierarchicznej struktury bloków

Przykład

Sub serwis transakcyjny sklepu internetowego

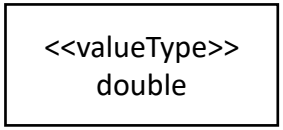


Typy wartości

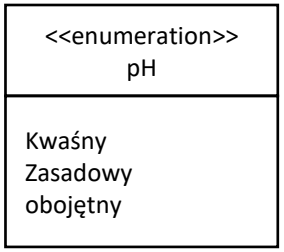
W SysML typy wartości stosuje się do opisu cech i parametrów operacji.

Rodzaje typów wartości:

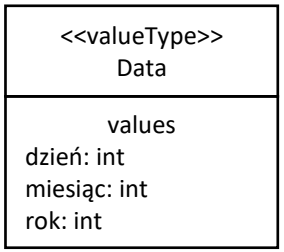
- **proste** – niepodzielne jednostki danych (np. int, float, real, float, boolean, char…)



- **wyliczeniowe** – czyli typy wartości ograniczone do policzalnej listy wariantów, określane stereotypem <<enumeration>>

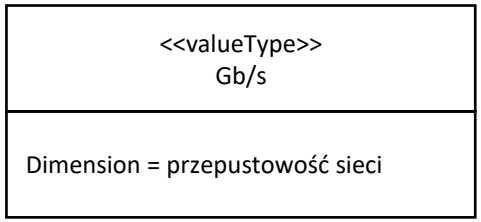


- **struktury** – łańcuch typów prostych

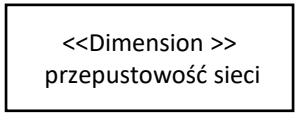


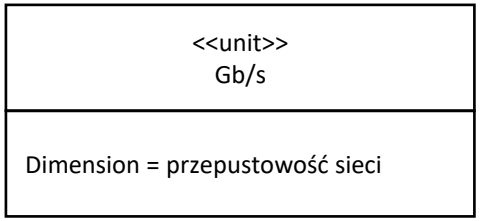
Typy wartości

- Typy wartości można charakteryzować i klasyfikować w kategoriach miar (dimensions) i jednostek (units)

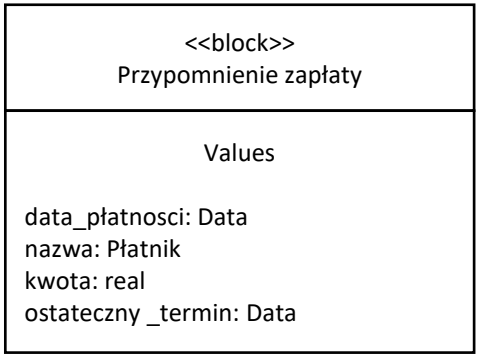


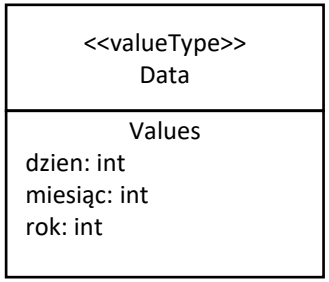
Albo w formie stereotypowanej

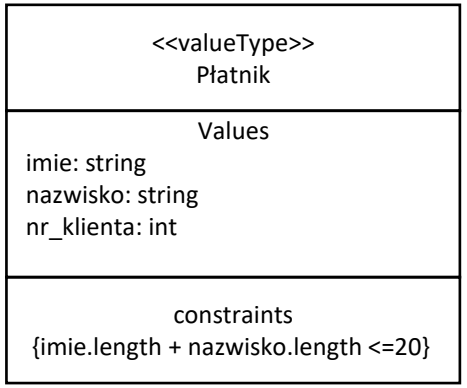




- Można również zdefiniować własne typy wartości







Zaawansowana specyfikacja bloków

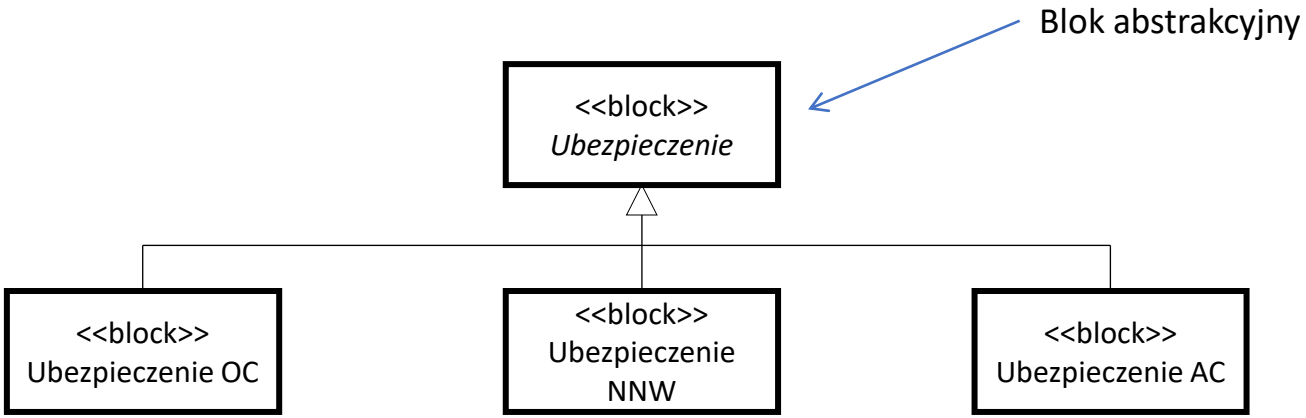
Sekcja struktury – dodatkowa sekcja bloku, która pozwala na zamieszczenie kompletnego albo też fragmentu **diagramu bloków wewnętrznych** w bloku modelowanym.

Diagram bloków wewnętrznych omówiony zostanie dalej.

Bloki abstrakcyjne

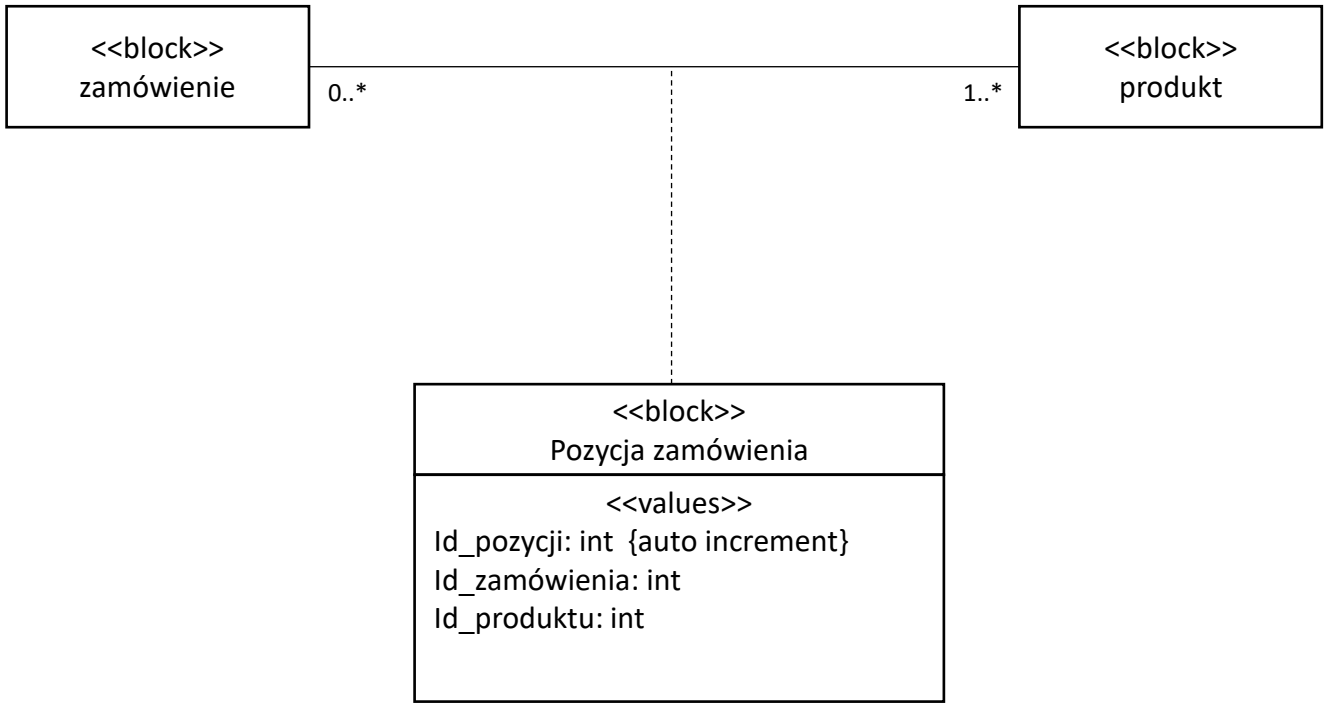
Bloki mogą mieć charakter abstrakcyjny albo konkretny.

Blok abstrakcyjny – stanowi wyłącznie podstawę dziedziczenia, nie zawierają kompletnej deklaracji wszystkich cech, jest ona rozszerzana przez bloki podrzędne. Nie posiada bezpośredniej instancji.



Bloki asocjacyjne

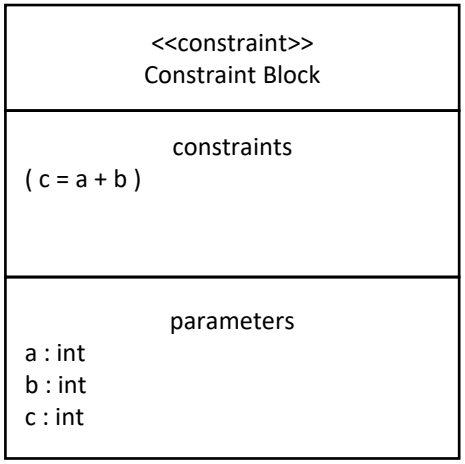
Podobnie jak w UML asocjacje mogą być szczegółowo zdefiniowane za pomocą bloków asocjacyjnych



Bloki ograniczeń

Ograniczenie, które jest jedną z cech bloku, można przedstawić jako osobny blok

- w celu ponownego wykorzystania
- bardziej szczegółowa specyfikacja
- są definiowane w **bdd** a wykorzystywane przede wszystkim w diagramach parametrycznych **par**



Alokacje

Alokacja stanowi sposób wzajemnego przyporządkowania różnych kategorii modelowania wywodzących się z różnych diagramów.

Umożliwia luźne powiązanie elementów modelu obejmujących różne diagramy.

Stosowana jest we wczesnej fazie projektowania systemu.

Alokacje

- **alokacja zachowania** – dotyczy alokowania elementów diagramów dynamiki systemów (czynności, akcji, stanów, przepływu obiektów…) do elementów strukturalnych (bloków, cech, części…), typowy przykład dla SysML to alokacja czynności do bloków.

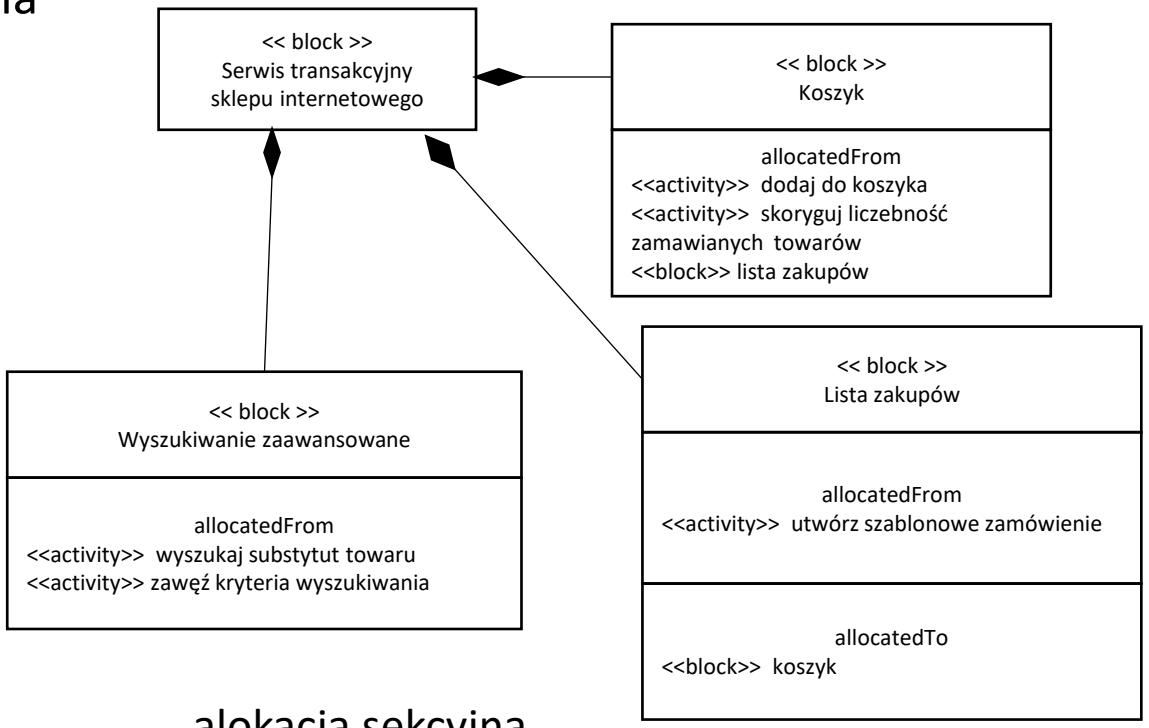
- **alokacja struktury** – alokowanie kategorii modelowania jednego modelu strukturalnego do kategorii modelowania innego modelu strukturalnego, zależności pomiędzy elementami diagramów definiowania bloków a diagramów bloków wewnętrznych, typowy przykład dla SysML to alokowanie oprogramowania na sprzęcie (UML diagram komponentów i rozlokowania języka UML).

- **alokacja przepływów** – wskazują w jakim zakresie przepływy zasobów na diagramach bloków wewnętrznych wynikają z przepływów obiektów na diagramie.

Notacje alokacji

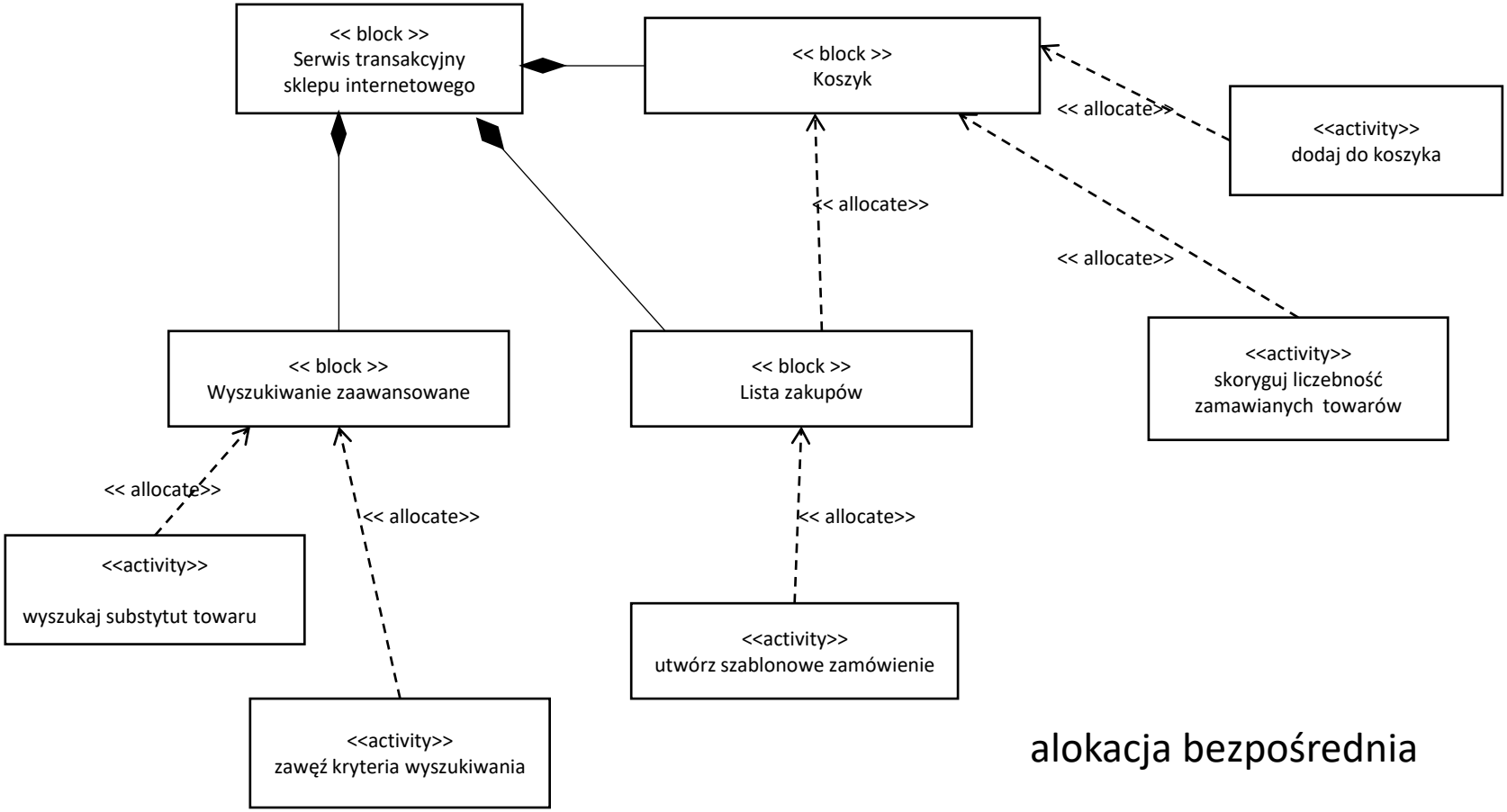
Notacje alokacji:

- bezpośrednia
- sekcyjna
- notatkowa
- tabelaryczna



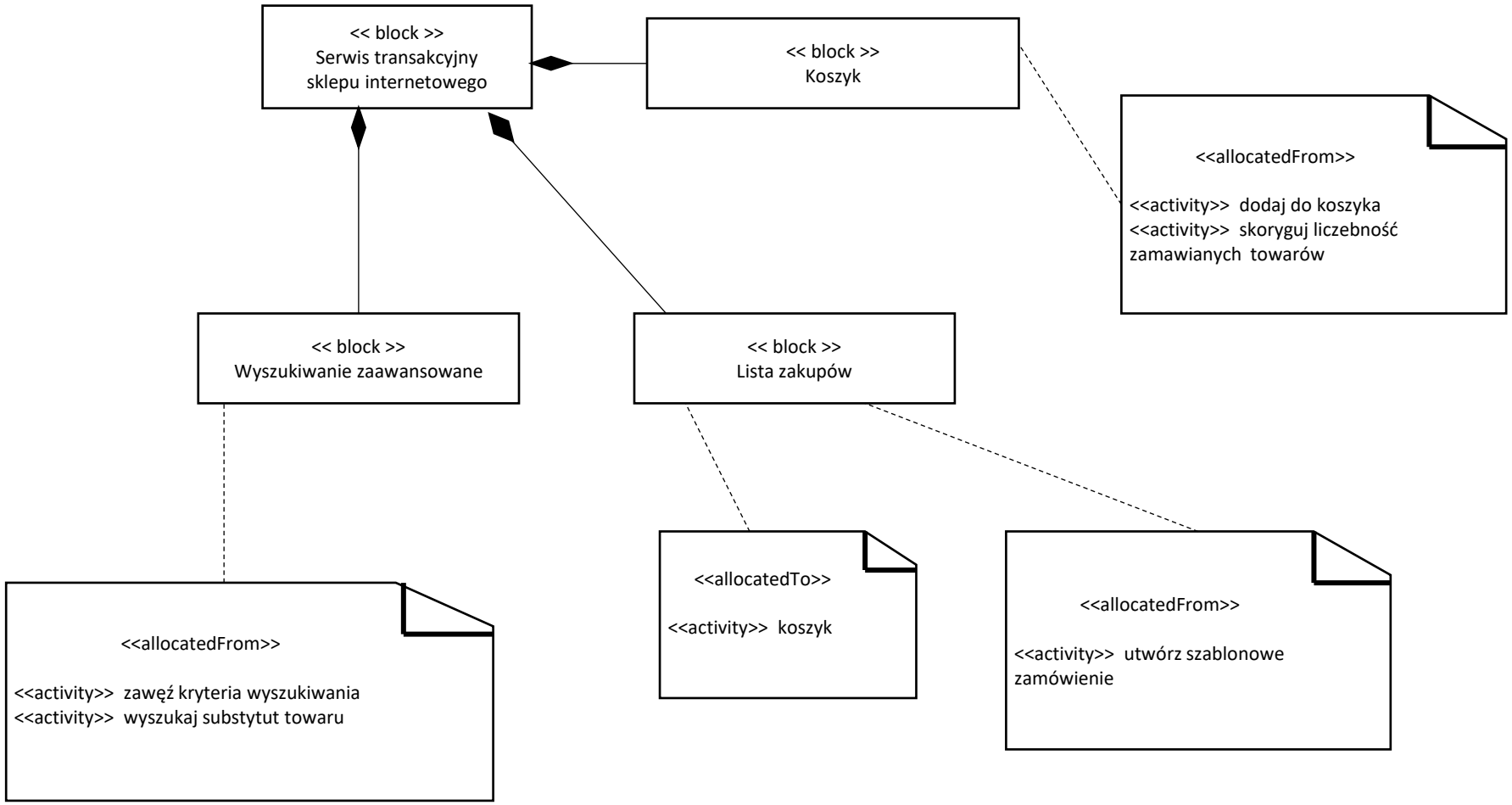
alokacja sekcyjna

Notacje alokacji



alokacja bezpośrednia

Notacje alokacji



alokacja notatkowa

Notacje alokacji

Rodzaj kategorii modelowania	Nazwa kategorii modelowania	Typ	Związek	Rodzaj kategorii modelowania	Nazwa kategorii modelowania	Typ
blok	Lista zakupowa	źródło	alokacja	Blok	koszyk	cel
blok	koszyk	cel	alokacja	czynność	Dodaj do koszyka	źródło

.....

alokacja tabelaryczna

Diagram bloków wewnętrznych (ibd)

Diagram bloków wewnętrznych (ang. *Internal Block Diagram*) pokazuje zastosowanie bloków w określonym kontekście poprzez wskazanie, jakie zasoby (fizyczne, informacyjne), oraz usługi systemowe są użytkowane przez poszczególne części bloków

- Diagramy bloków wewnętrznych bazują na diagramach struktur połączonych języka UML
- Pokazuje dynamiczne aspekty bloku w systemie
- Przedstawia interpołączenia i interfejsy między częściami bloków
- Pozwala na bardziej precyzyjne modelowanie wewnętrznej struktury bloku

Kategorie modelowania

Główne kategorie modelowania na diagramach bloków wewnętrznych:

- część **part**
- port **port**
- związek **relationship**

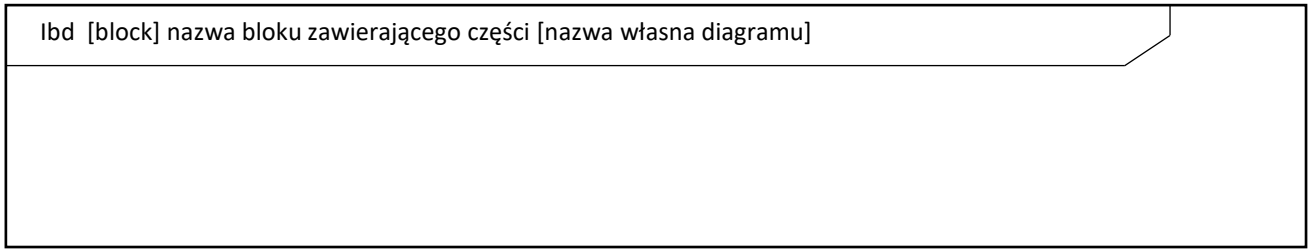
Część

Część opisuje lokalne zastosowanie bloku, definiującego część w określonym kontekście, do którego ta część należy (np. czytnik kart może być użytkowany na wiele sposobów: inne w module rejestracji czasu pracy a inne w module nadzoru)

Części obok wartości, operacji, ograniczeń, odniesień i cech uniwersalnych należą do pierwotnych cech bloku.

Część

Blok macierzysty na diagramie jest ramką diagramu bloków wewnętrznych



Ramy wykorzystuje się w odniesieniu do dowolnego diagramu języka SysML, dla kompletnych diagramów bloków wewnętrznych ma charakter obligatoryjny.

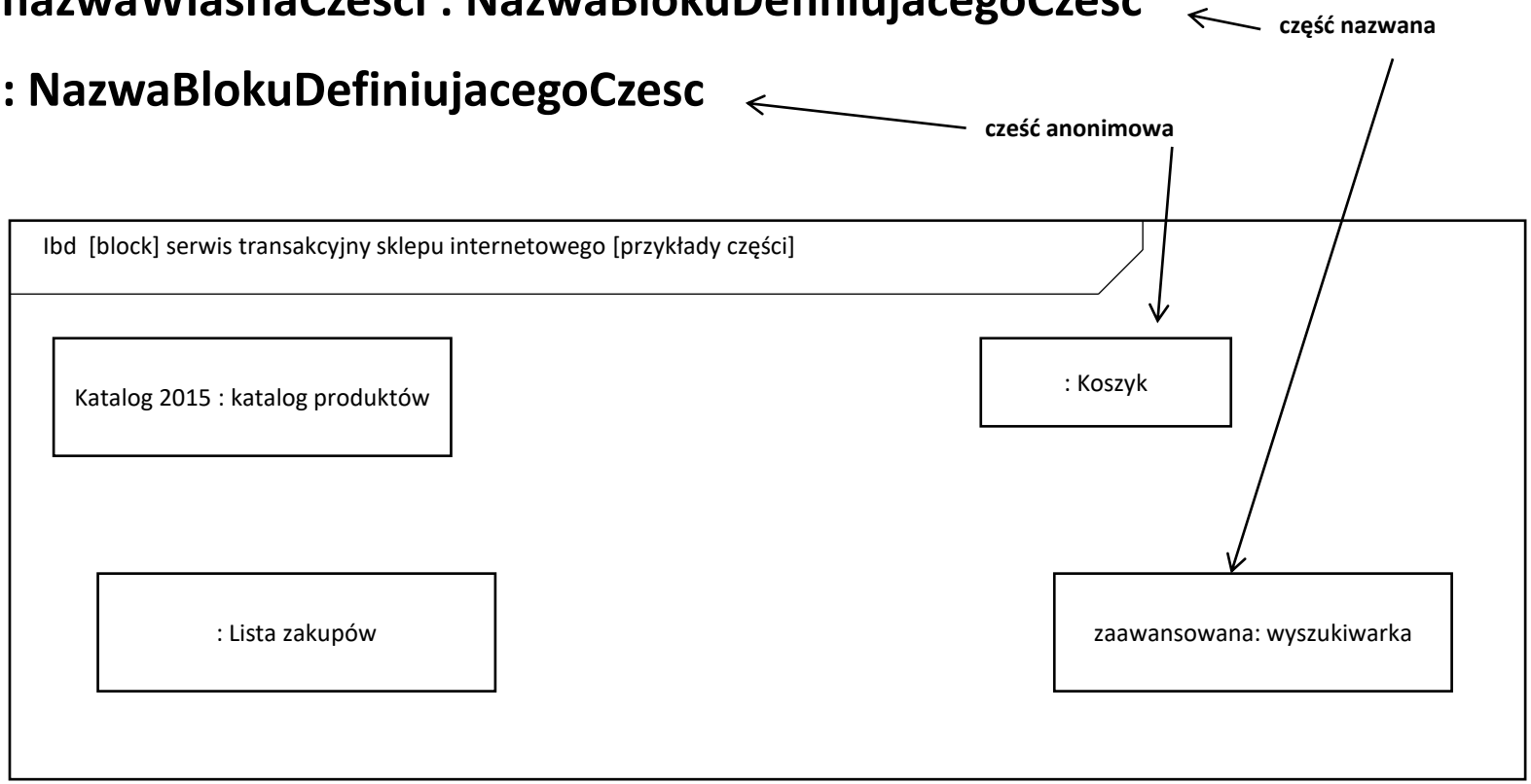
Część

Nazwa diagramu powinna zawierać nazwę bloku, którego części są opisywane na tym diagramie.

Nazwa części zawiera nazwę właściwą oraz nazwę bloku definiującego tę część:

nazwaWlasnaCzesci : NazwaBlokuDefiniujacegoCzesc

: NazwaBlokuDefiniujacegoCzesc



Port

Port jest to punkt interakcji z otoczeniem na krawędzi bloku lub jego części.

- Wskazuje na aspekty dynamiczne bloku lub jego części poprzez umożliwienie przepływu zasobów lub przywołanie usług systemowych.
- Bloki zazwyczaj projektuje się modularnie.
- Jednoznaczna definicja interfejsów pozwala na wielokrotne użycie bloków modularnych.
- Część lub blok mogą posiadać wiele portów, każdy dla innego rodzaju interakcji
- Porty mogą być powiązane łącznikami (binding connectors) tylko z innymi portami tego samego typu.

Klasyfikacja portów

- **port standardowy** – (standard port) wiąże się z usługami systemowymi, świadczonymi lub wymaganymi przez blok, które określane są przez interfejsy (udostępniające lub pozyskujące)

- **port transmisyjny** – (flow port) określa co wpływ a co wypływa z bloku.
  - **pojedynczy port transmisyjny** – (atomic) przepływ wyselekcjonowanego niepodzielnego zasobu (np. prądu)
  - **zagregowany port transmisyjny** – (non-atomic) przepływ ma charakter złożony (mogą przepływać różne zasoby i w przeciwnych kierunkach)

  Może być stosowana kombinacja portów transmisyjnych i standardowych ale nie można ich łączyć razem.

Pojedynczy port transmisyjny

Stereotypy graficzne dla pojedynczych portów transmisyjnych

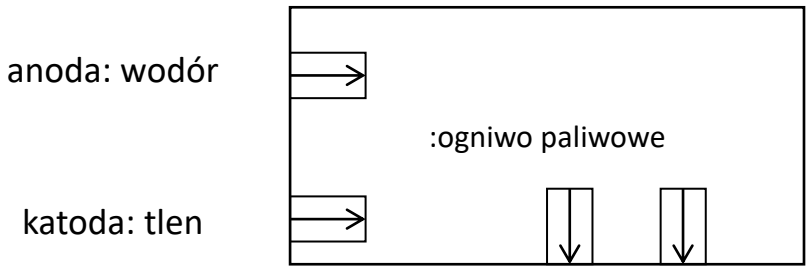
transmisja jednokierunkowa

transmisja dwukierunkowa

składnia portu:

[<identyfikator kierunku>] <nazwa właściwa> : <zasób>

in, out, inout



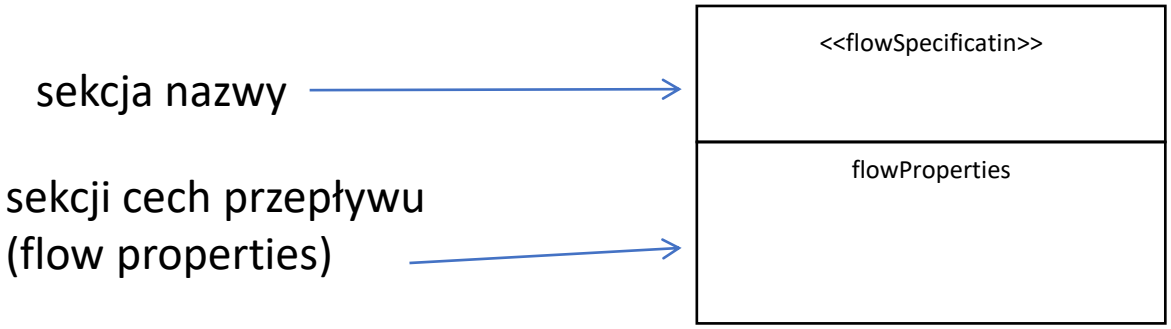
zasobnik : woda elektrolit: prąd elektryczny

Zagregowane porty transmisyjne

Zagregowany port transmisyjny jest alternatywą dla pojedynczego portu transmisyjnego. Różne zasoby składowe mogą przepływać w przeciwnych kierunkach.

Specyfikacja przepływu ma charakter interfejsu ze stereotypem <<flowSpecificatin>>

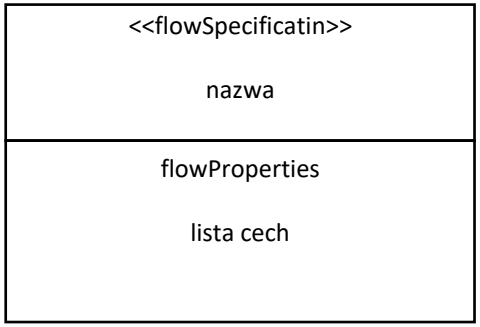
Składa się z dwóch sekcji:



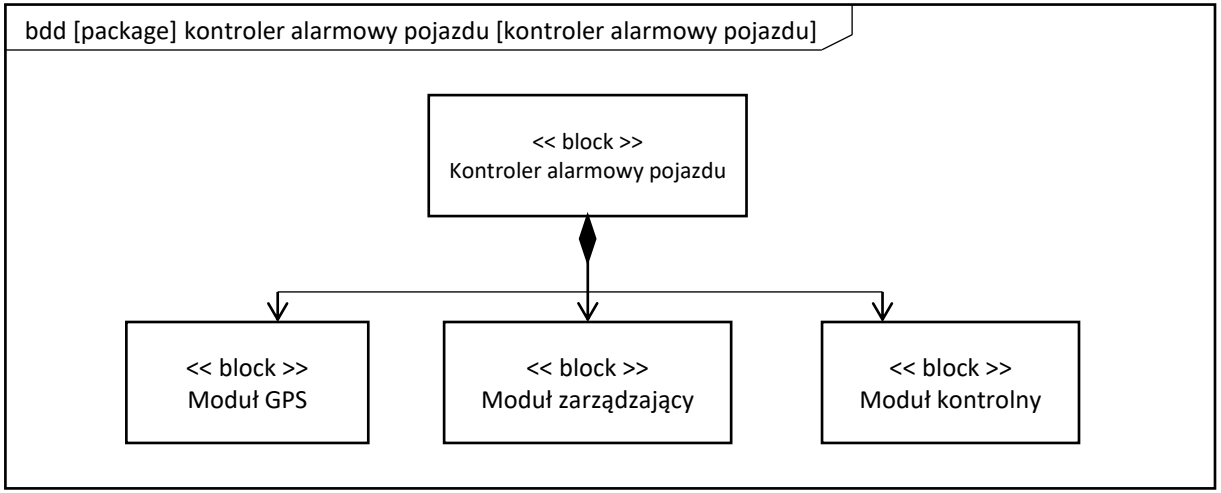
Zagregowane porty transmisyjne

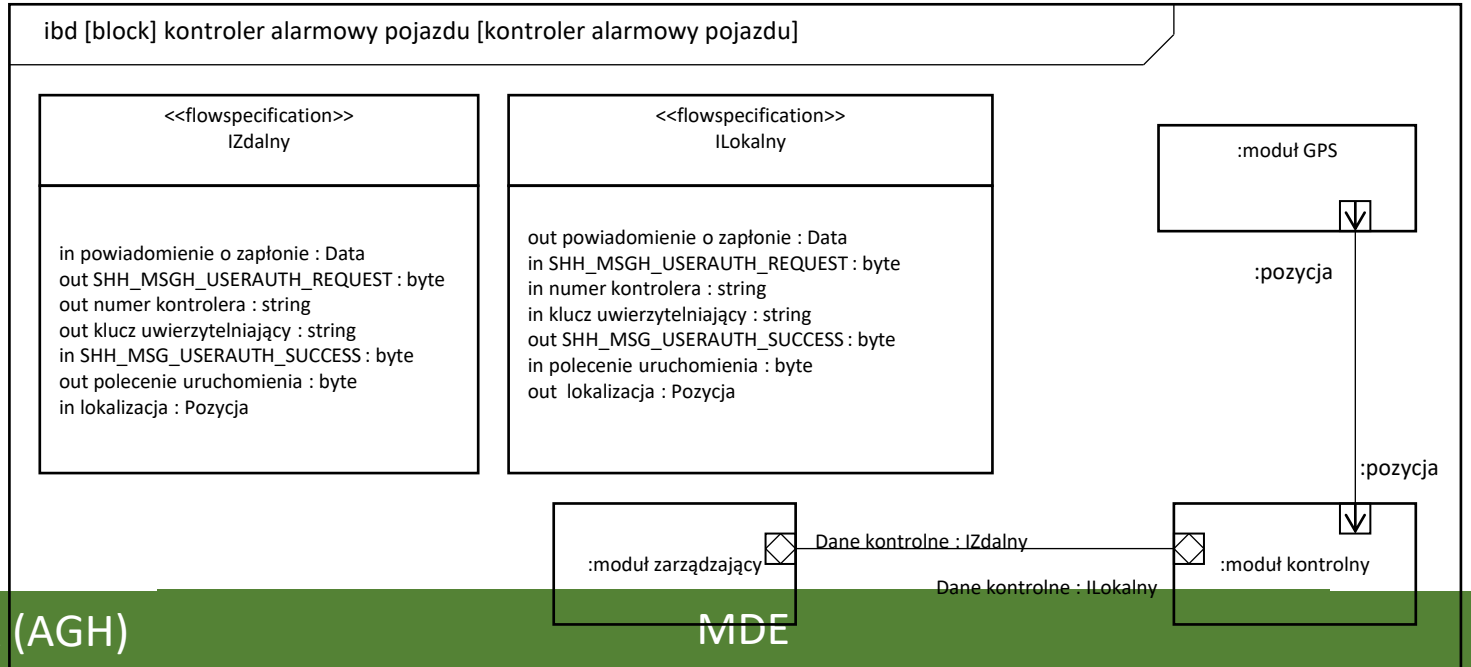
flow properties to lista cech, gdzie każda pozycja odpowiada odrębnemu zasobowi/komponentowi, składnia cechy jest identyczna jak dla pojedynczego portu transmisyjnego:

[<identyfikator kierunku>] <nazwa właściwa> : <zasób>



Zagregowane porty transmisyjne





Sprzęganie portów zagregowanych

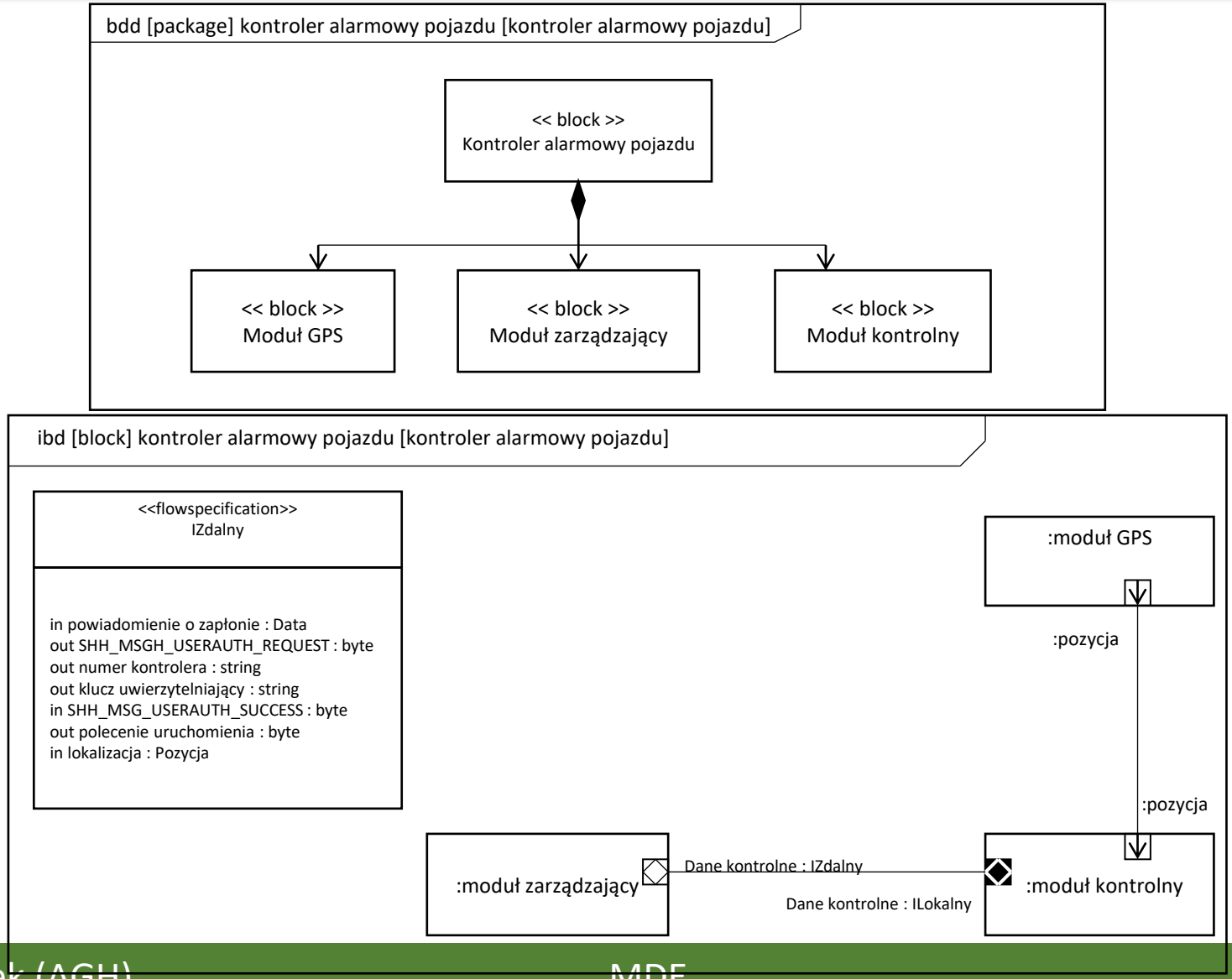
sprzężony port zagregowany (conjugated port) używamy w przypadku kiedy specyfikacja identyfikatorów kierunku w jednym porcie jest odwrotnością specyfikacji w innym porcie.

- specyfikacja występuje po jednej stronie łącznika
- przez porty wymieniane są analogiczne zasoby ale w przeciwnych kierunkach
- zaznacza się to przy pomocy symboli:

port specyfikowany

port sprzężony

Zagregowane porty transmisyjne – sprzęganie



Porty standardowe

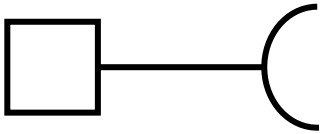
Porty standardowe wywodzą się bezpośrednio z **UML**, umożliwiają udostępnianie i pozyskiwanie usług systemowych (są stosowane w kontekście architektury opartej na usługach.

Port standardowy może użytkować dwa rodzaje interfejsów

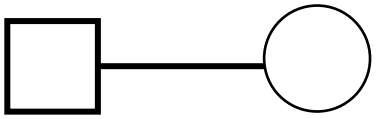
- interfejs **pozyskujący** (required)
- interfejs **udostępniający** (provided)

Jeżeli istnieje część wyposażona w port z interfejsem pozyskującym to musi istnieć cześć, która zaoferuje określone usługi.

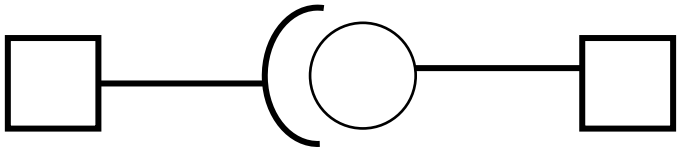
Interfejsy



Interfejs pozyskujący określa usługi (operacje) wymagane przez blok albo część w celu wykonywania funkcjonalności bloku lub części.

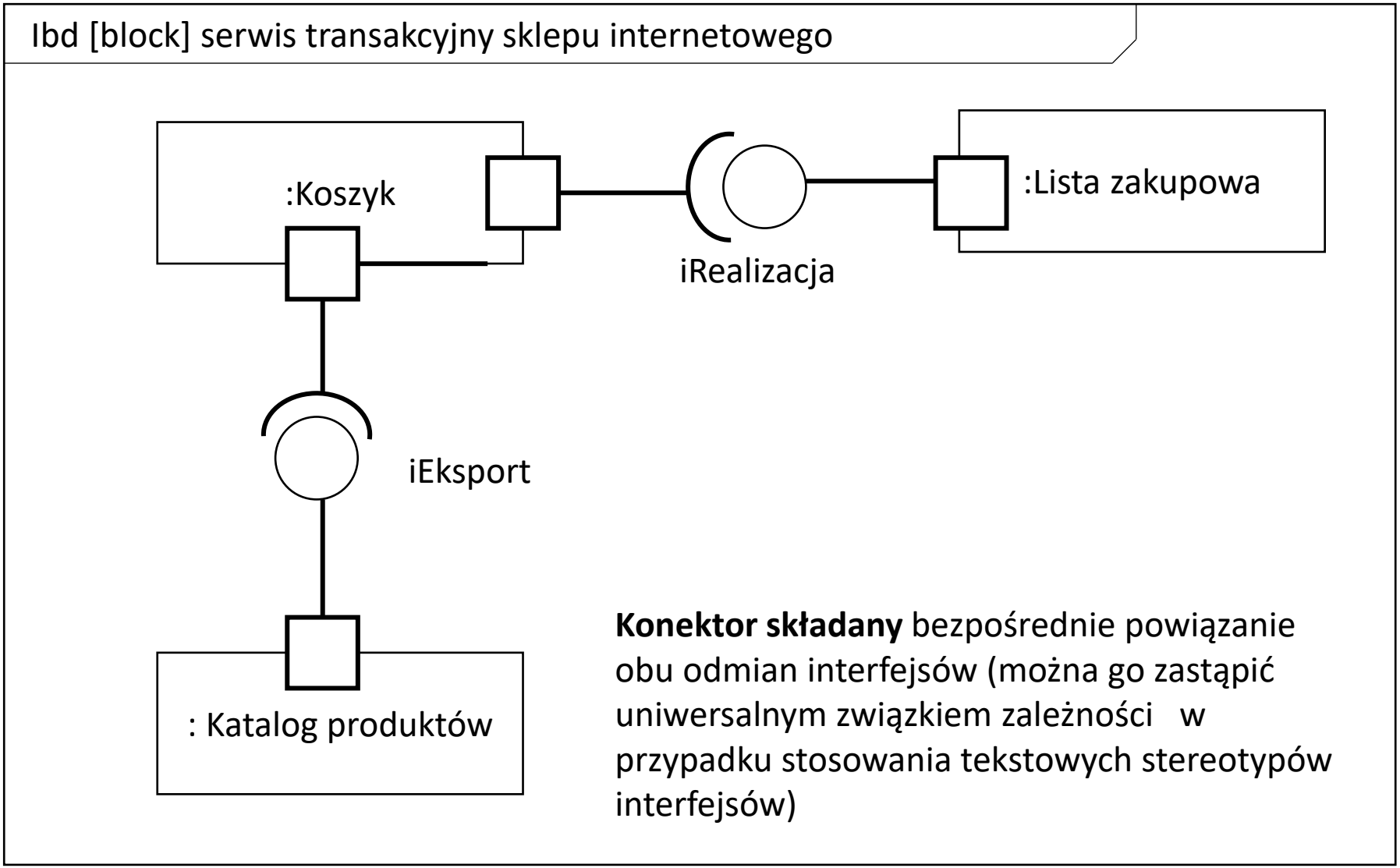


Interfejs pozyskujący określa usługi (operacje), które blok lub część świadczy na rzecz innego bloku lub części w celu wykonania funkcjonalności bloku lub części



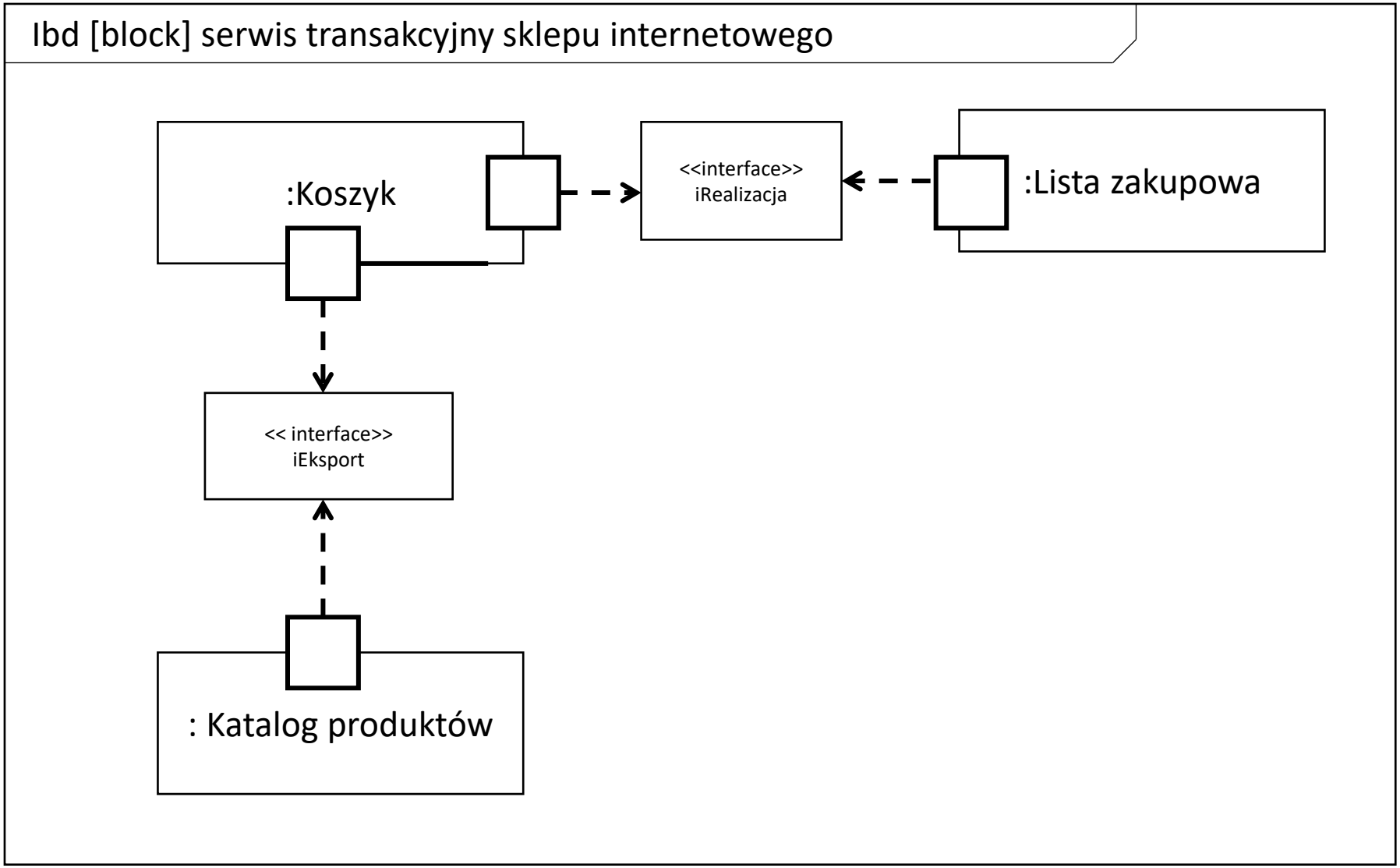
Konektor składany bezpośrednie powiązanie obu odmian interfejsów (można go zastąpić uniwersalnym związkiem zależności w przypadku stosowania tekstowych stereotypów interfejsów)

Interfejsy i porty standardowe



Konektor składany bezpośrednie powiązanie obu odmian interfejsów (można go zastąpić uniwersalnym związkiem zależności w przypadku stosowania tekstowych stereotypów interfejsów)

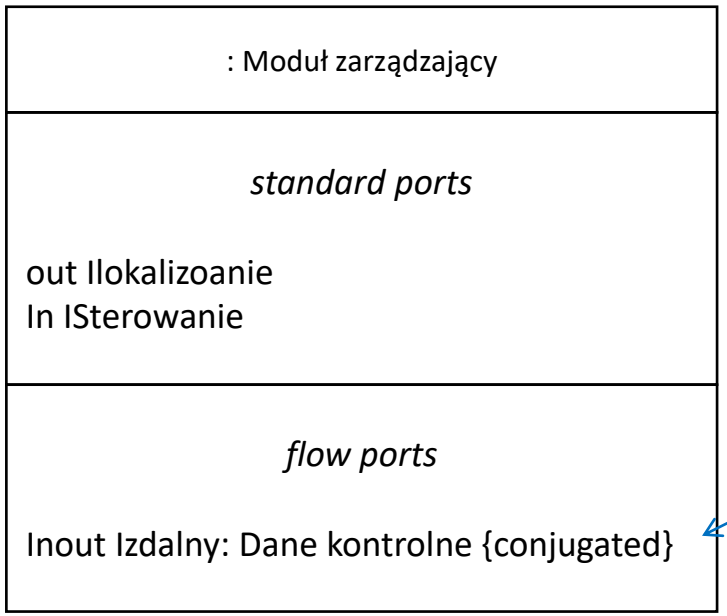
Porty standardowe



Definiowanie portów w sekcjach

Porty można także definiować w sekcjach bloków/części

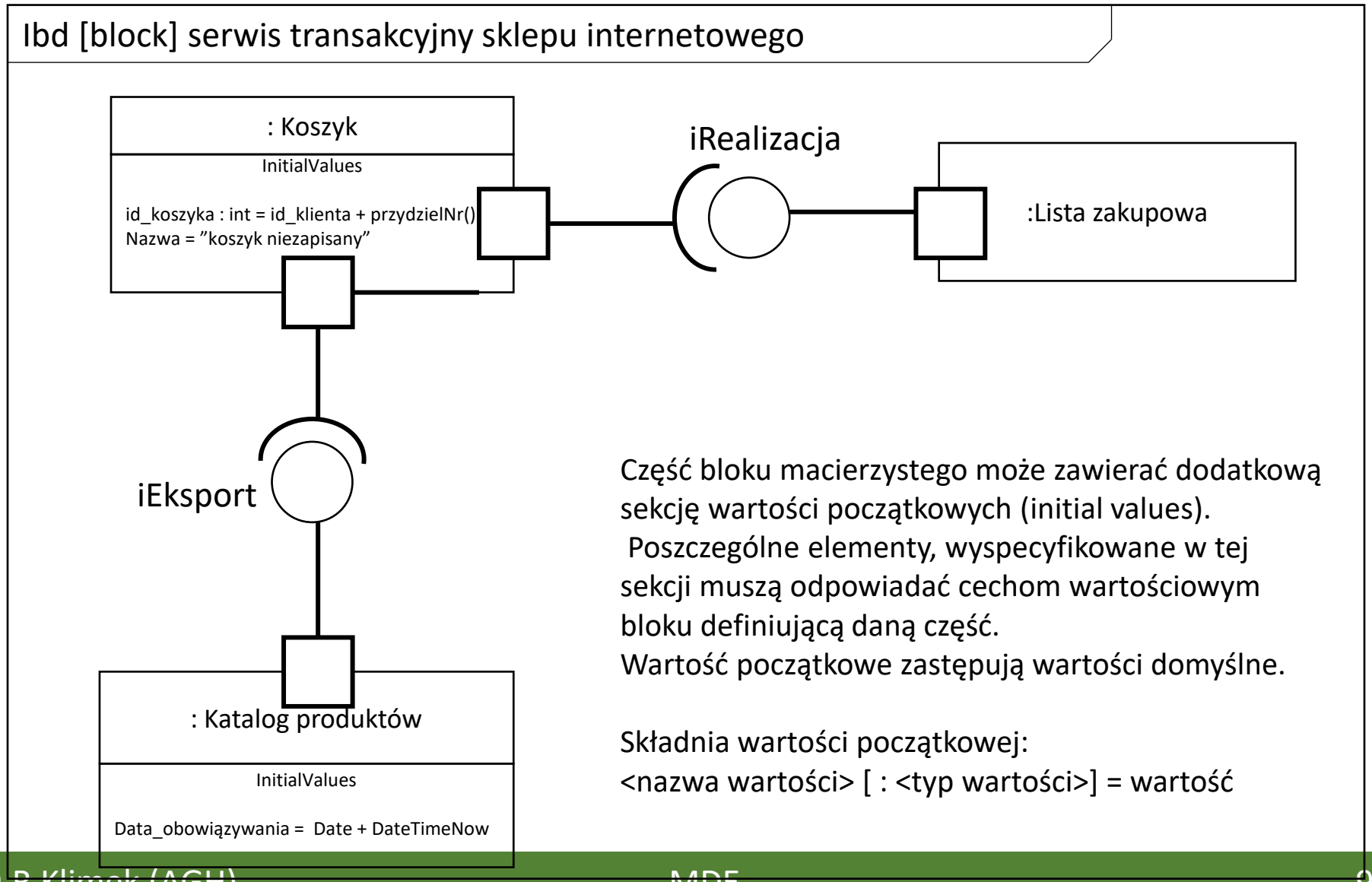
- sekcja definiowania portów standardowych (standard ports)
- sekcja definiowania portów transmisyjnych (flow ports)



W każdej z sekcji zamieszcza się listę portów danego typu. Składnia każdego portu jest taka sama jak składnia pojedynczego bloku.

Jeżeli definiowany jest port sprzężony należy dodatkowo umieścić ograniczenie {cojugated}

Ibd – wartość początkowa



Część bloku macierzystego może zawierać dodatkową sekcję wartości początkowych (initial values).
Poszczególne elementy, wyspecyfikowane w tej sekcji muszą odpowiadać cechom wartościowym bloku definiującą daną część.
Wartość początkowe zastępują wartości domyślne.

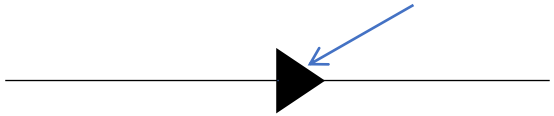
Składnia wartości początkowej:
<nazwa wartości> [: <typ wartości>] = wartość

Przepływ zasobów

Język SysML umożliwia modelowanie

- przepływu informacji (jak UML)
- bytów fizycznych

na **ibd** zaznaczany jako przepływ zasobów (item flow)

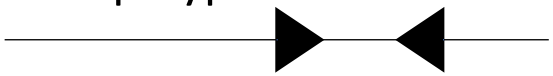


- składnia nazwy:

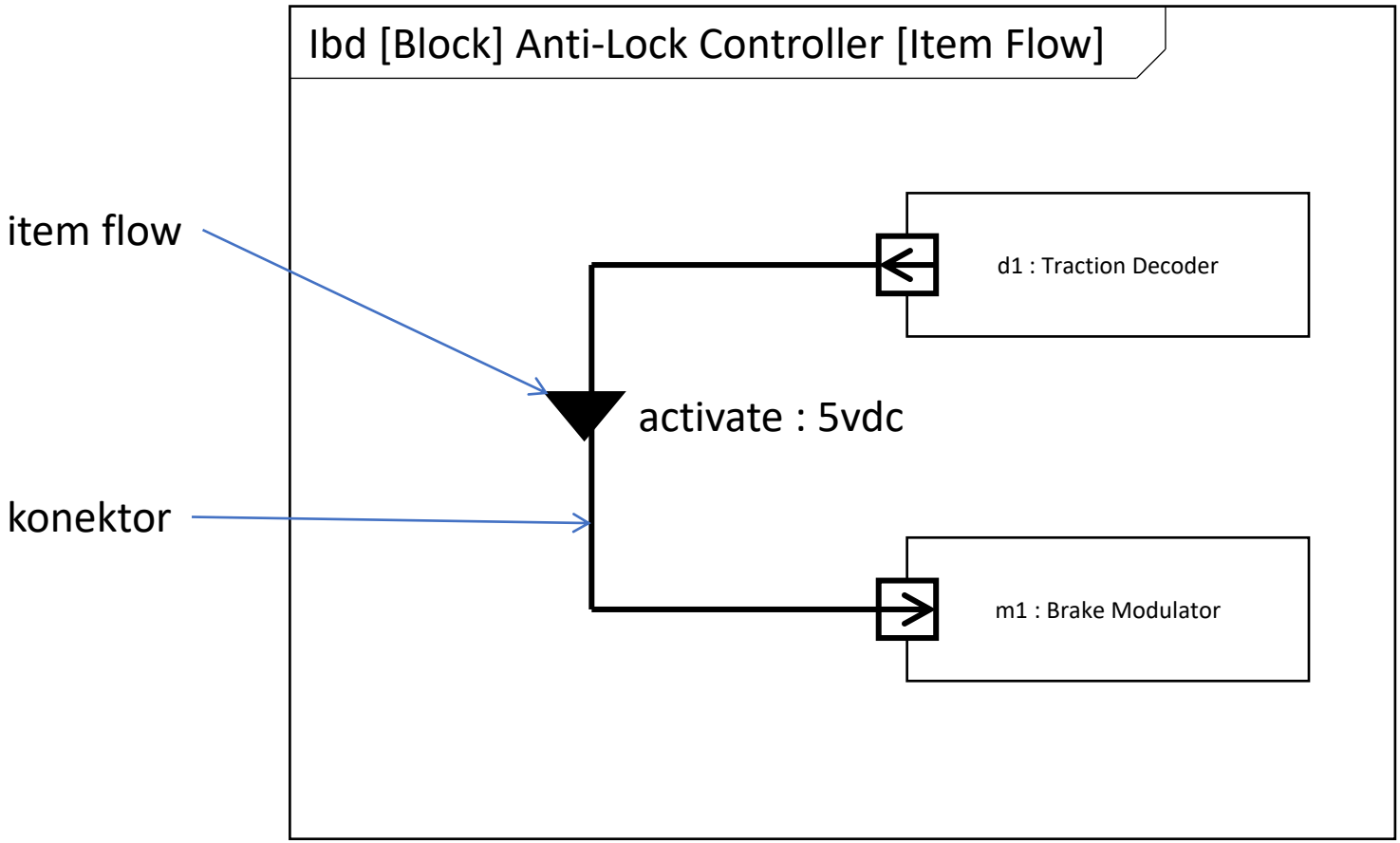
  nazwaPrzepływu : rodzaj Zasobu

  : rodzajZasobu (dla anonimowego)

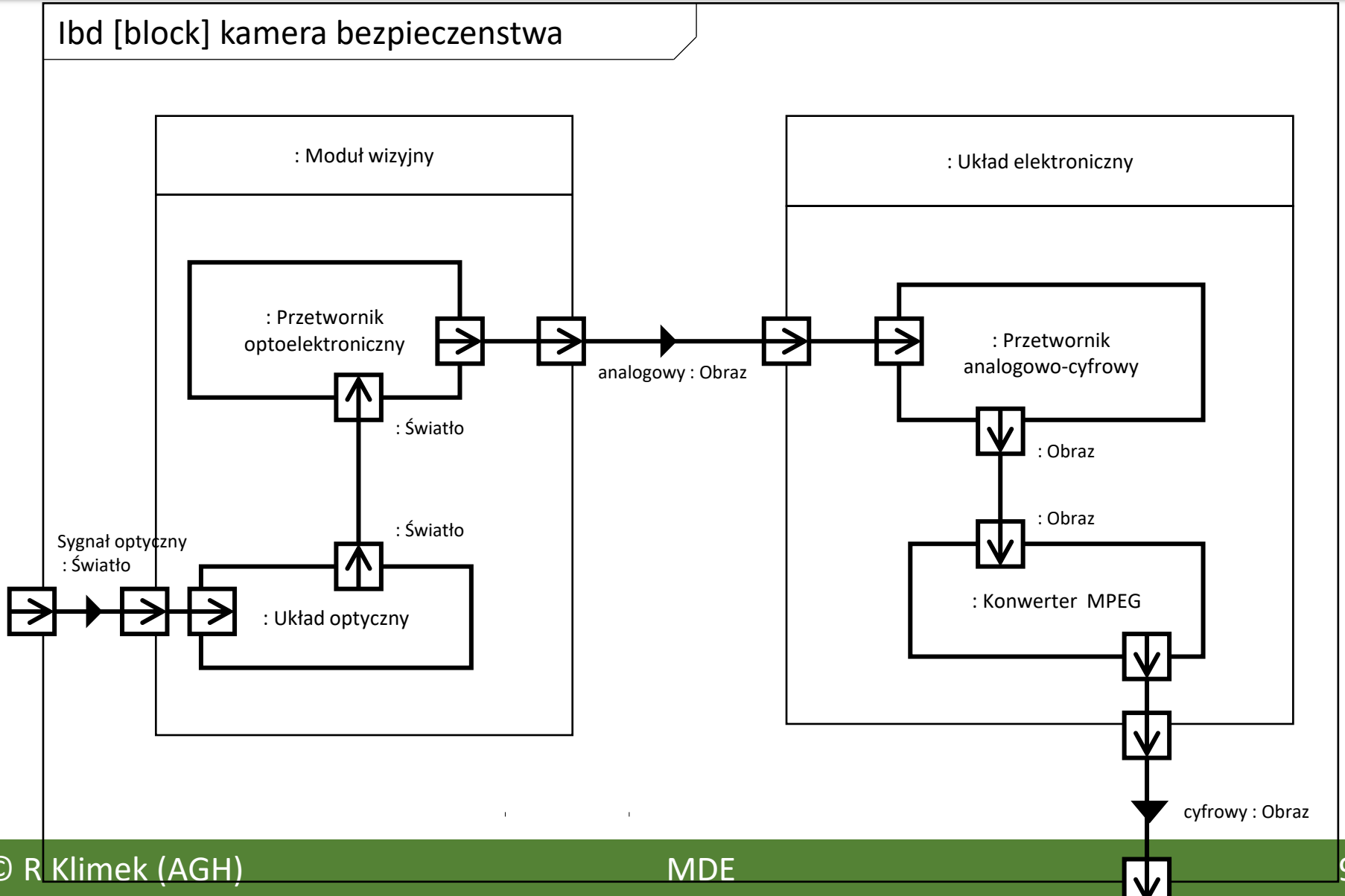
- specyfikuje rodzaj transmitowanego zasobu i kierunek jego przepływu
- pojedynczemu konektorowi można przypisać kilka zasobów o dowolnych kierunkach



Przepływ zasobów – przykład

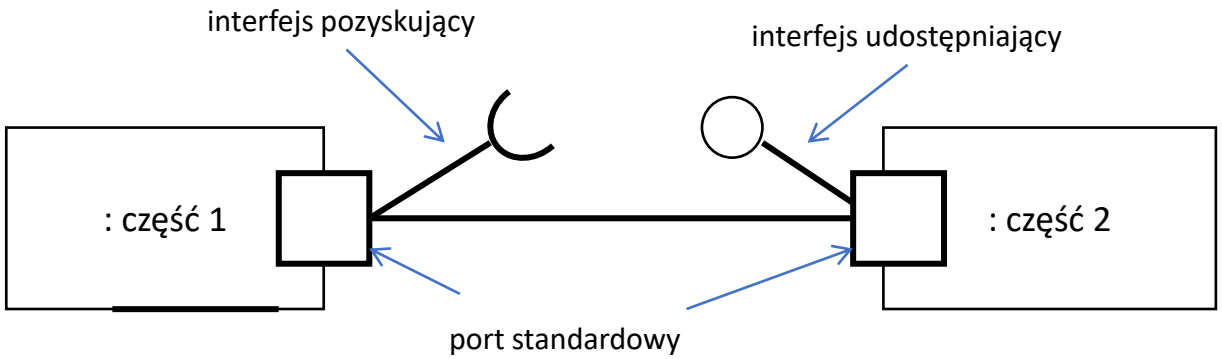


Przepływ zasobów – przykład

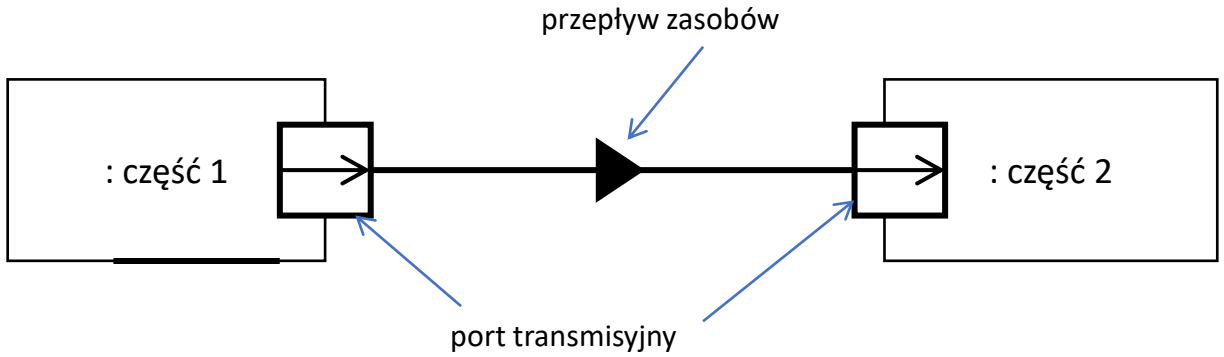


Porty, interfejsy

Porty standardowe



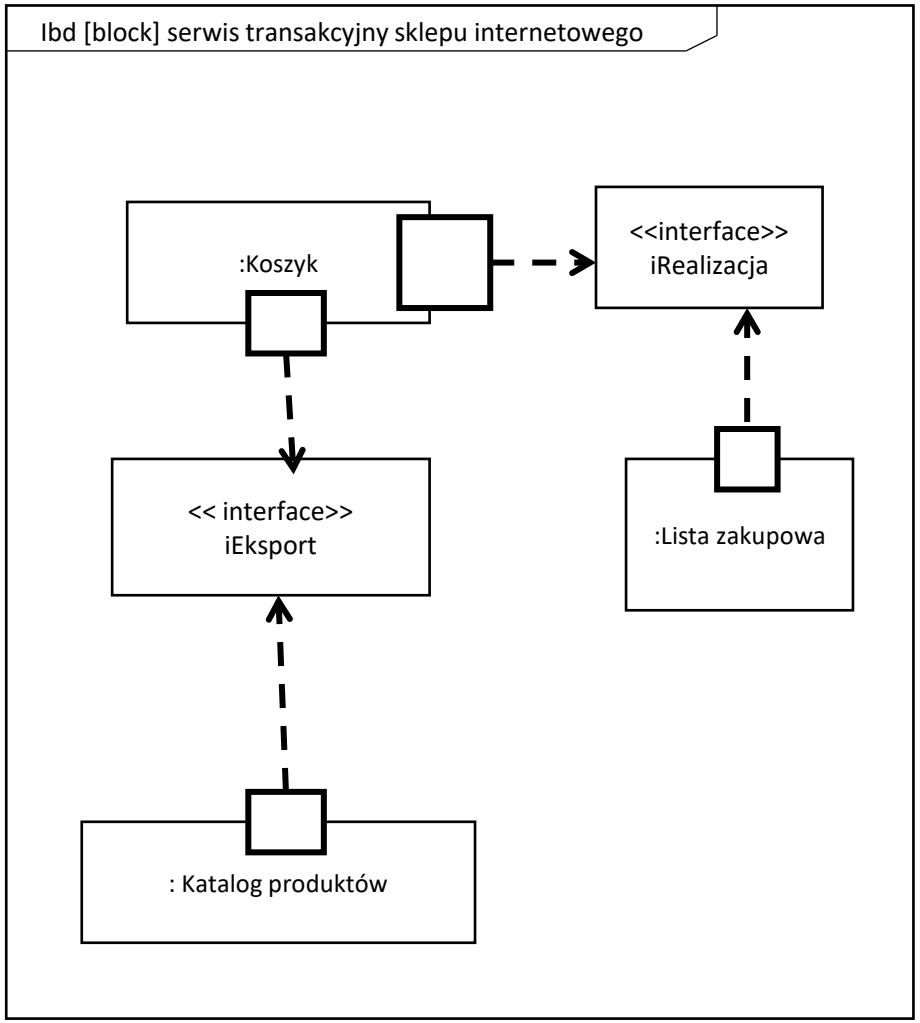
Porty transmisyjne

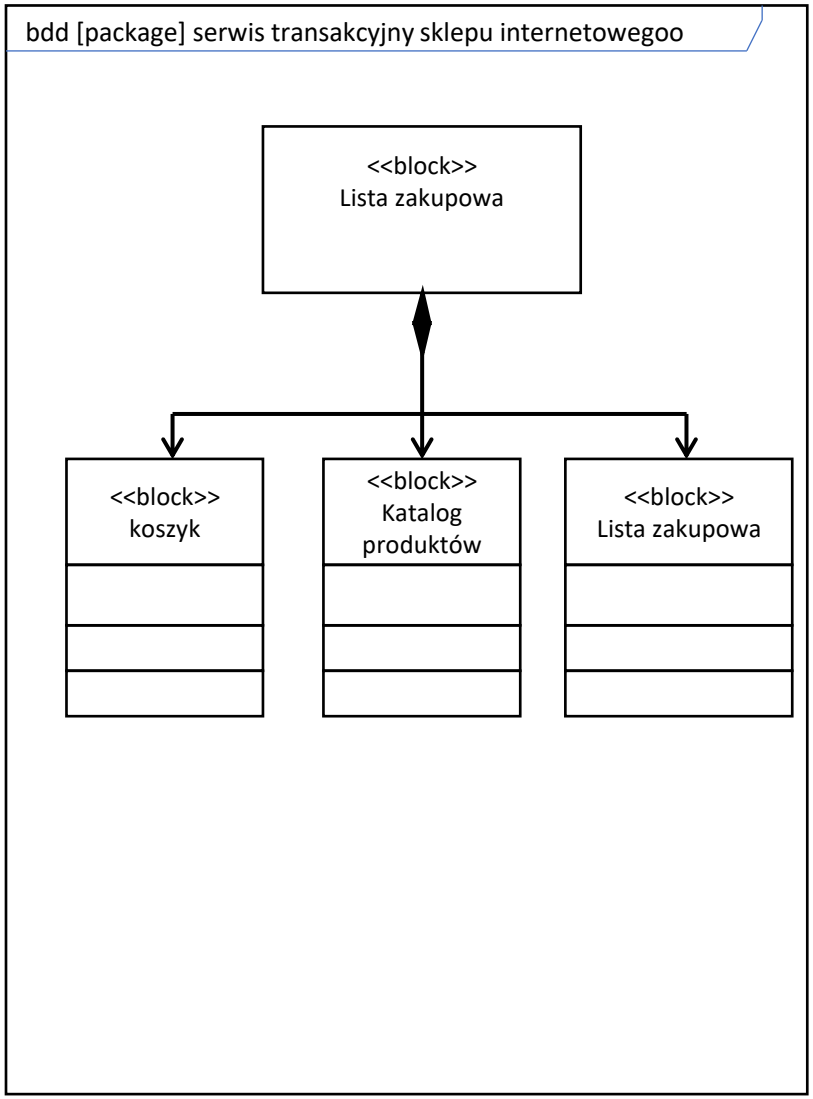


Referencje do bloków i części

- Zastosowanie bloków asocjacyjnych pozwala na odwołanie się do bloków i części, które nie są elementami składowymi modelowanego bloku.
- Przywoływany blok/część może jawnie wyszczególniać wszystkie cechy za pomocą sekcji
- Używanie referencji nie implikuje używania portów

Referencje do bloków i części – przykład



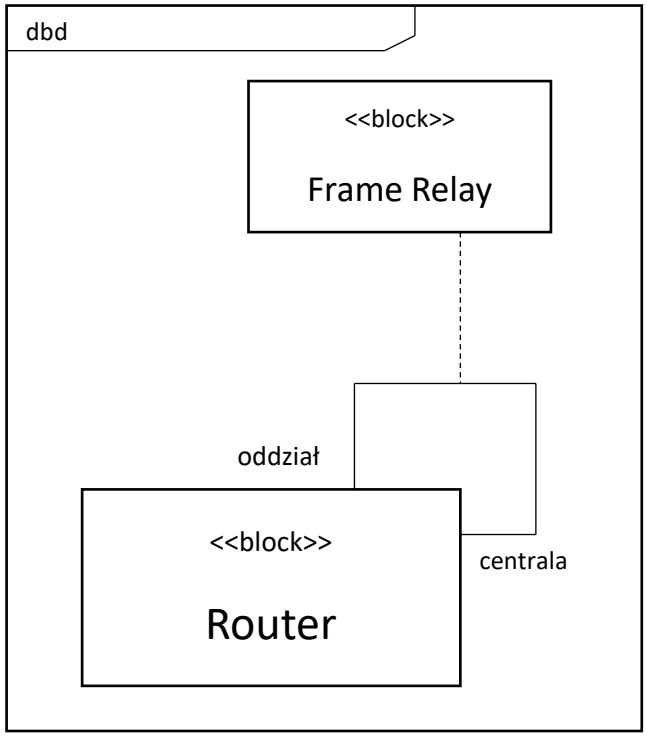


Węzeł bloku asocjacyjnego

Aby zastosować bloki asocjacyjne na diagramie **bdd** należy na diagramie **ibd** umieścić węzły asocjacyjne (participant properties).

Węzeł bloku asocjacyjnego reprezentuje pojedynczą końcówkę asocjacji (może być traktowany jako szczególny przypadek przywołania).

Węzeł bloku asocjacyjnego – przykład



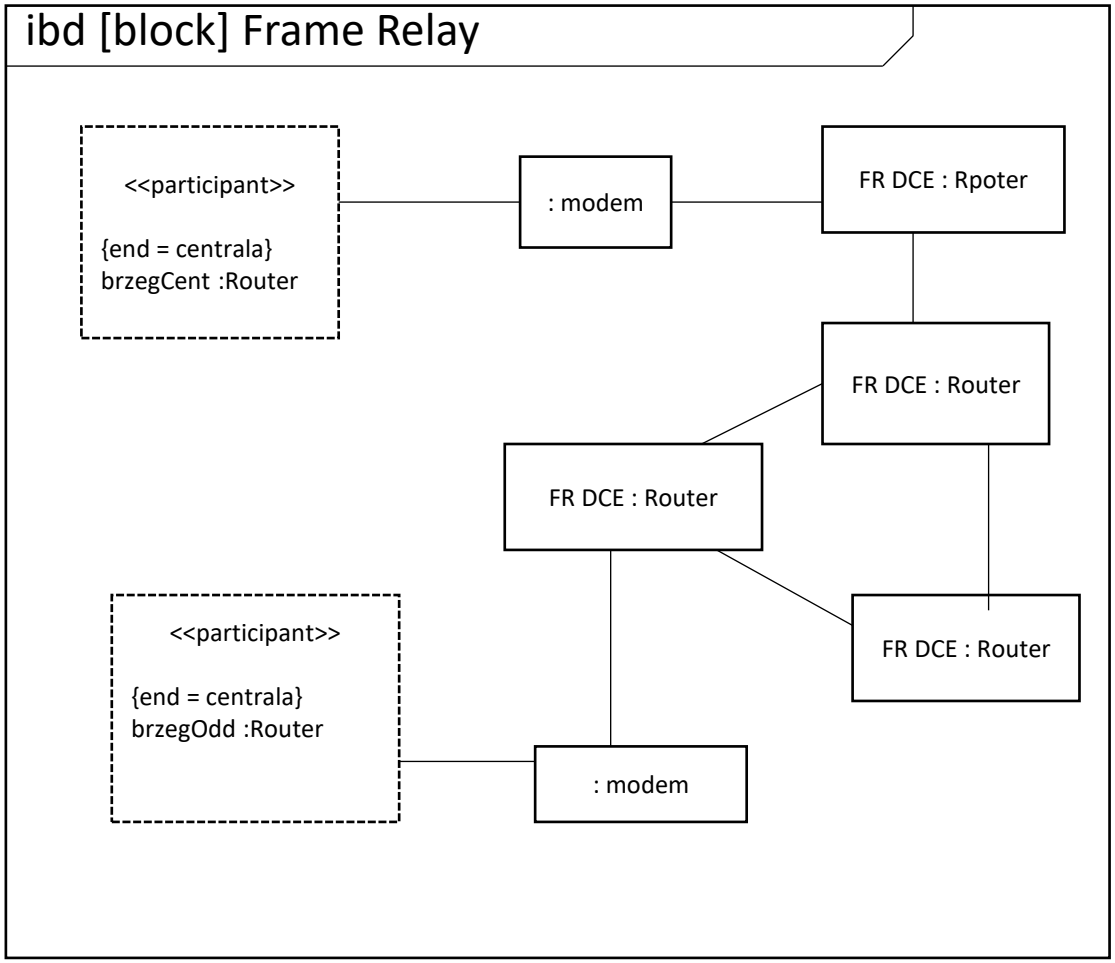
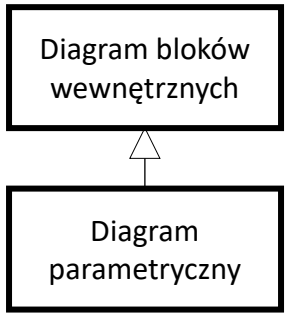
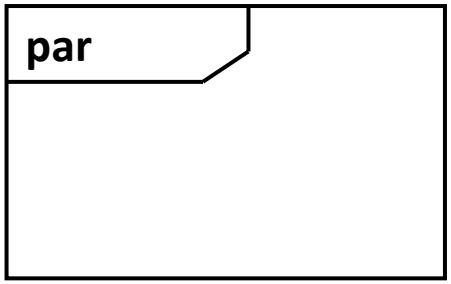


Diagram parametryczny (par)

Diagram parametryczny





- Diagram parametryczny stanowi sformalizowaną odmianę diagramu bloków wewnętrznych.
- Koncentruje się na ograniczeniach cech bloków określonych na diagramie definiowania bloków.
- Jest niezbędny do reprezentowania relacji matematycznych lub ograniczeń w SysML
- Ograniczenia są wyrażane jako równania, których parametry są przypisywane kategoriom modelowania struktury systemu.

Diagram parametryczny

Stosowanie diagramów parametrycznych ma na celu

- wspomaganie studiów nad różnymi wersjami produktu/procesu pod kątem analizy i wyboru wariantów
- wspomaganie analizy i optymalizcji projektu (symulacja)
- określanie
  - wydajności
  - niezawodności
  - cech fizycznych systemu

Diagram parametryczny

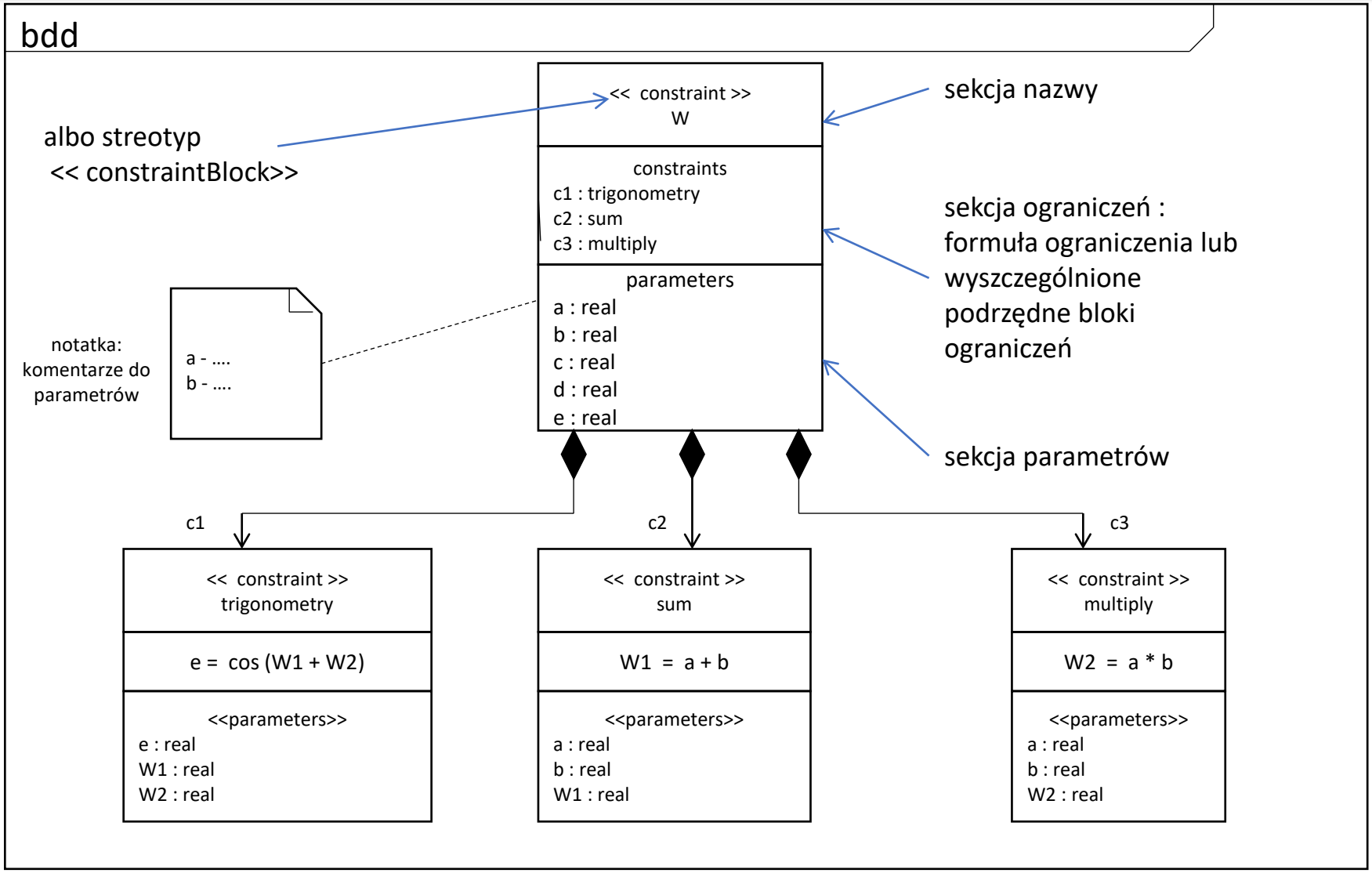
Punktem wyjścia do konstruowania diagramu parametrycznego jest zdefiniowanie **bloków ograniczeń** (constraint block)

- bloki ograniczeń umieszczane są na diagramach definiowania bloków **bdd**
- zastosowanie bloku ograniczeń w określonym kontekście przyjmuje postać **cechy ograniczającej** (constraint property)
- połączone konektorami cechy ograniczające tworzą diagram parametryczny
- jeżeli przewidywana jest analiza wariantowa, wówczas do diagramów **par** wprowadza się **funkcje celowe,** które operują na cechach wartościowych (miara efektywności) poszczególnych bloków

Blok ograniczeń

- stanowi zapisane w postaci formalnej równania matematyczne
- wskazuje na ograniczenia i reguły występujące w strukturze sytemu, oraz parametry niezbędne do wykonania tych formuł
- parametry w SysMl są odpowiednikiem atrybutów z diagramu klas UML
- bloki ograniczeń zawierają jedynie formalny opis (niezależny od kontekstu), mogą być więc użyte ponownie (w innych projektach sytemach – **reused**)
- SysMl nie posiada własnego języka ograniczeń – korzysta się z zewnętrznych formalnych języków ograniczeń (OCL, Shernatron, Alloy, MathML), albo opisuje nieformalnie
- ograniczenie może być zawarte w każdym elemencie (blok, pakiet) i może być zależne od czasu

Blok ograniczeń – definiowanie



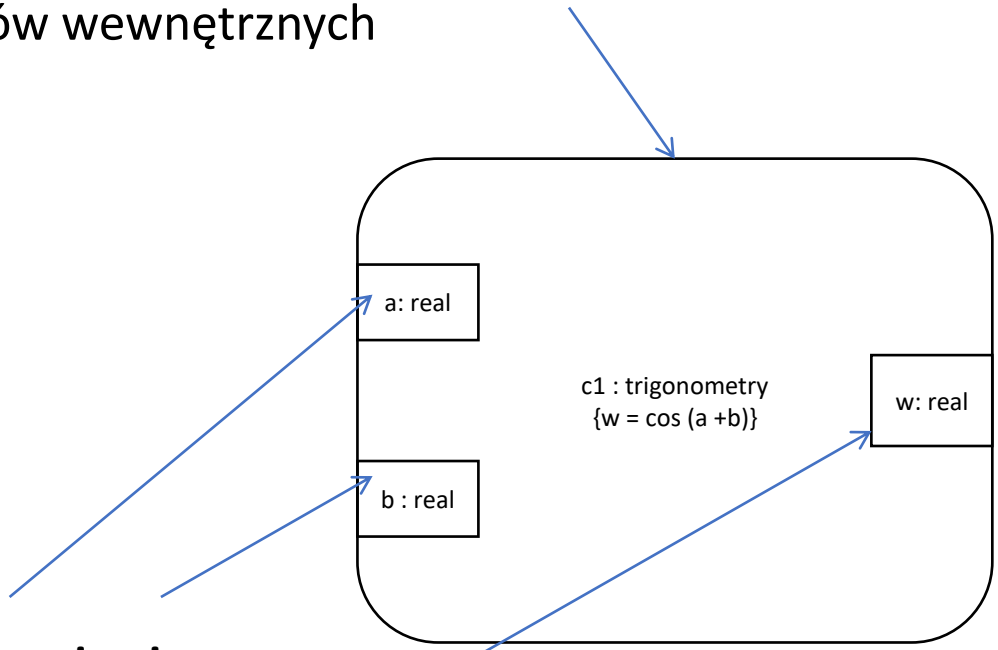
Tworzenie bloku parametrycznego

Cztery etapy tworzenia diagramu parametrycznego:

- zdefiniowanie bloku ograniczeń na podstawie reguł diagramu definiowania bloków
- utworzenie właściwego diagramu (lub wielu diagramów)
- przypisanie wartości cechom ograniczającym
- przeprowadzenie właściwej analizy i/lub studiów alternatywnych wariantów

Cecha ograniczająca

Cecha ograniczająca (constraint property) jest to konkretne zastosowanie bloku ograniczającego na diagramie parametrycznym, podobnie jak części są zastosowaniem bloku na diagramie bloków wewnętrznych



parametry ograniczające wejściowe i wyjściowe (constraint parameters) odpowiadają parametrom bloku ograniczającego

Diagram parametryczny

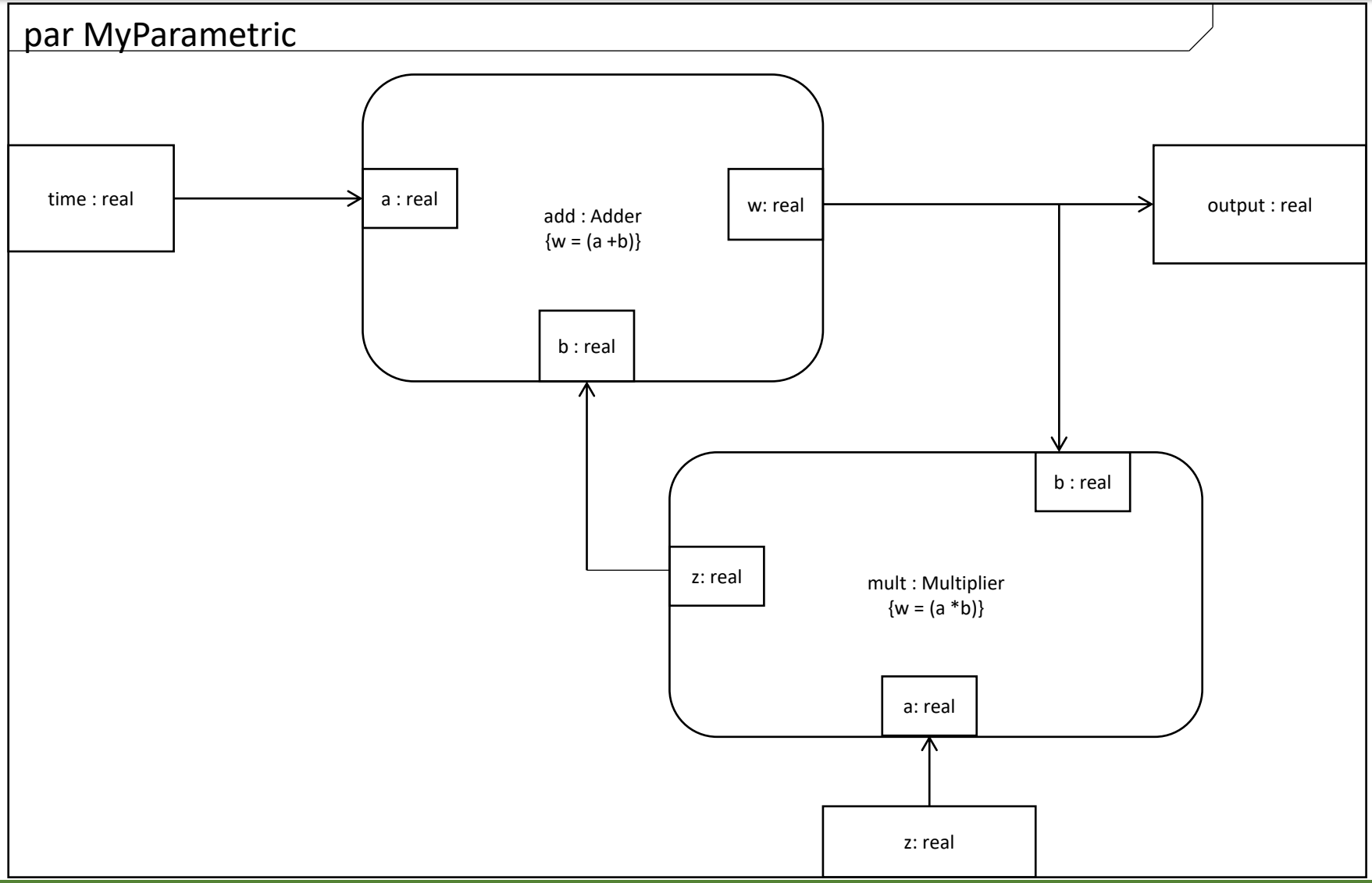
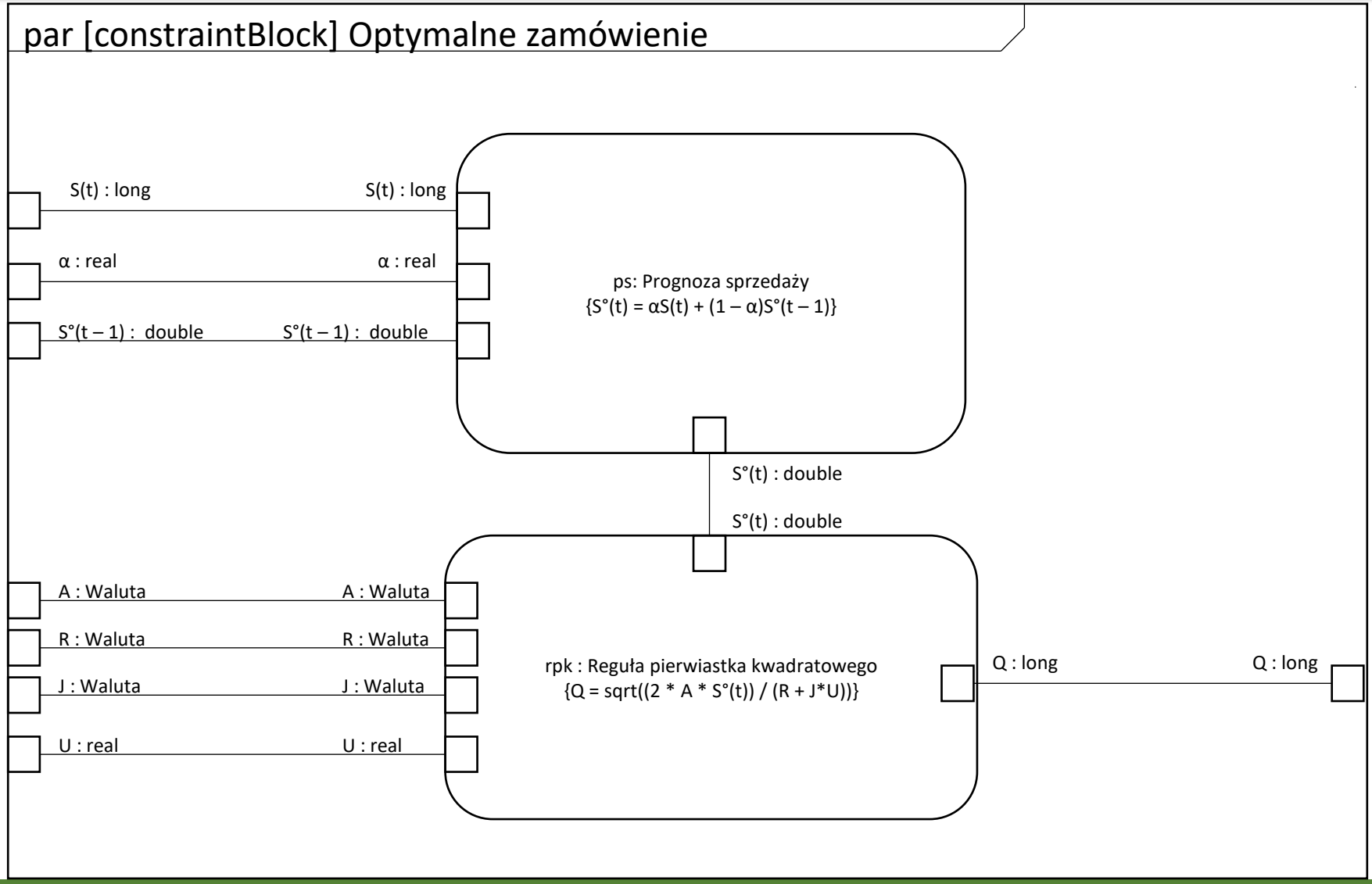


Diagram parametryczny



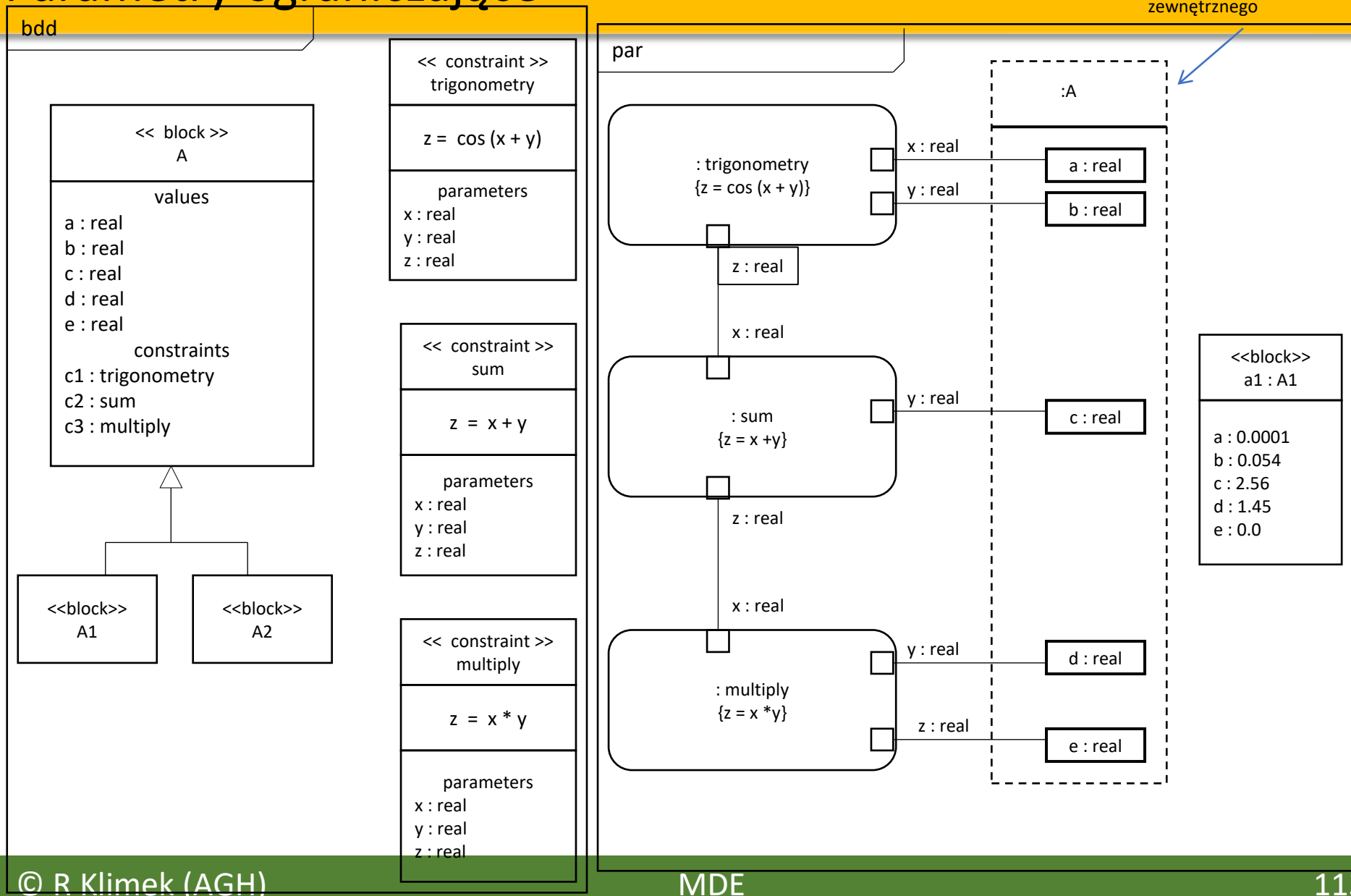
Parametry ograniczające

Aby przeprowadzić poprawną analizę procesów i produktów parametrom ograniczającym należy nadać dobrany zestaw wartości

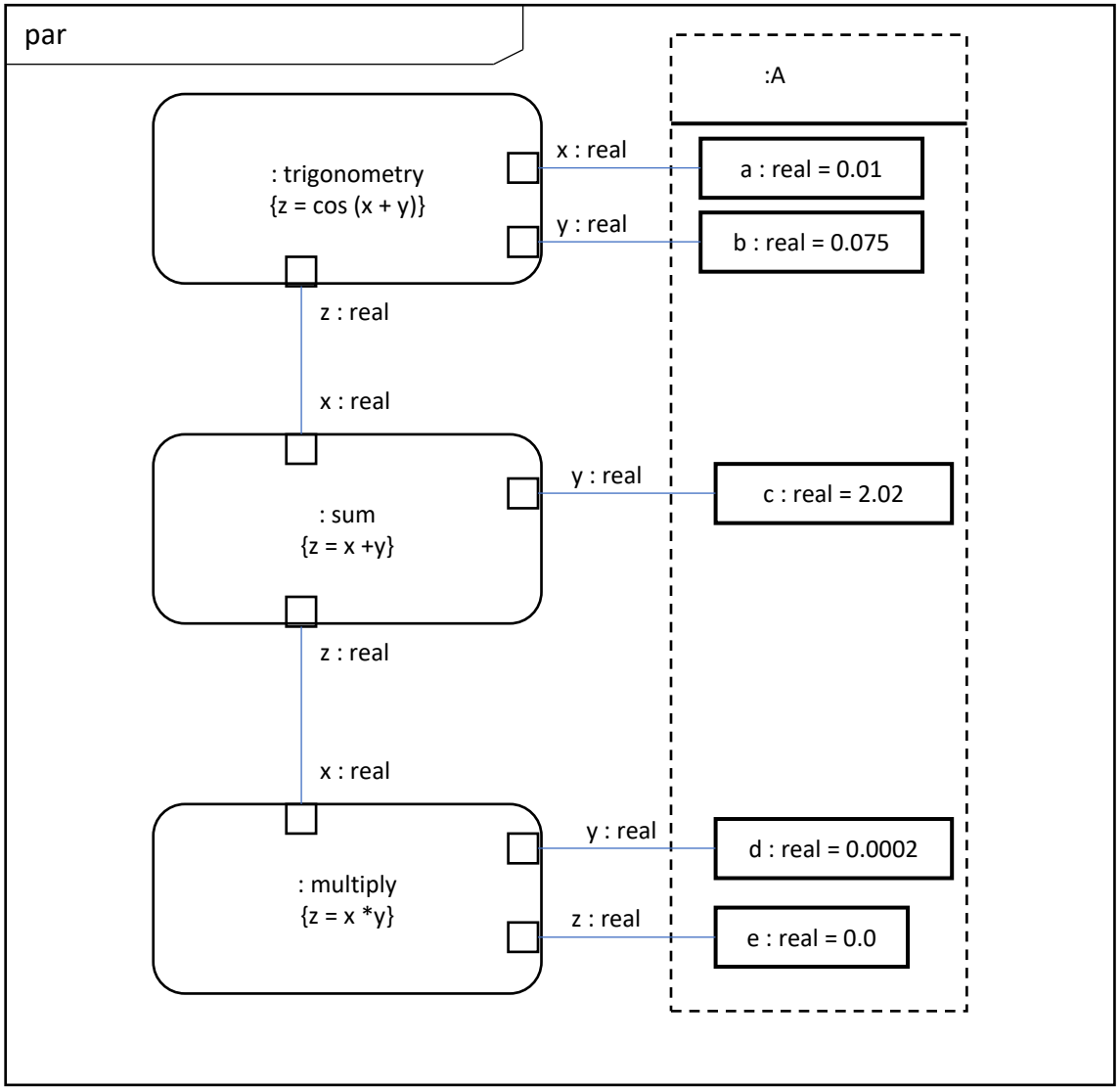
- jawnie (na diagramie parametrycznym)
- dynamicznie, w czasie rzeczywisty, przez zastosowane narzędzie analityczne

Konieczne jest wyspecyfikowanie źródła parametrów ograniczających (z jakich części, bloków cech ograniczających pochodzą)

Parametry ograniczające



Parametry ograniczające



Nadanie wartości parametrom ograniczającym w sposób jawny na diagramie par

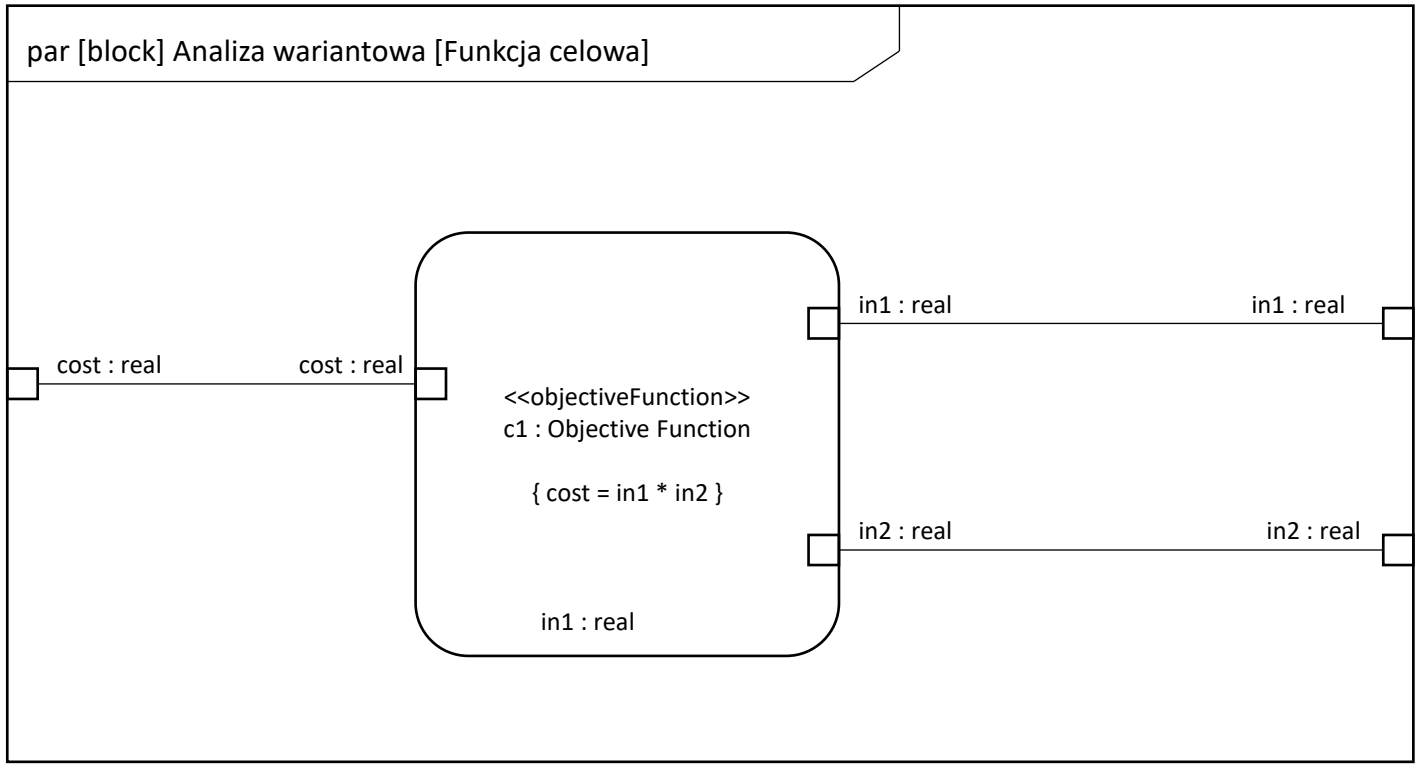
Funkcje celowe

W celu przeprowadzenia analizy wariantowej należy dostarczyć różne zestawy wartości wejściowych parametrów ograniczających.

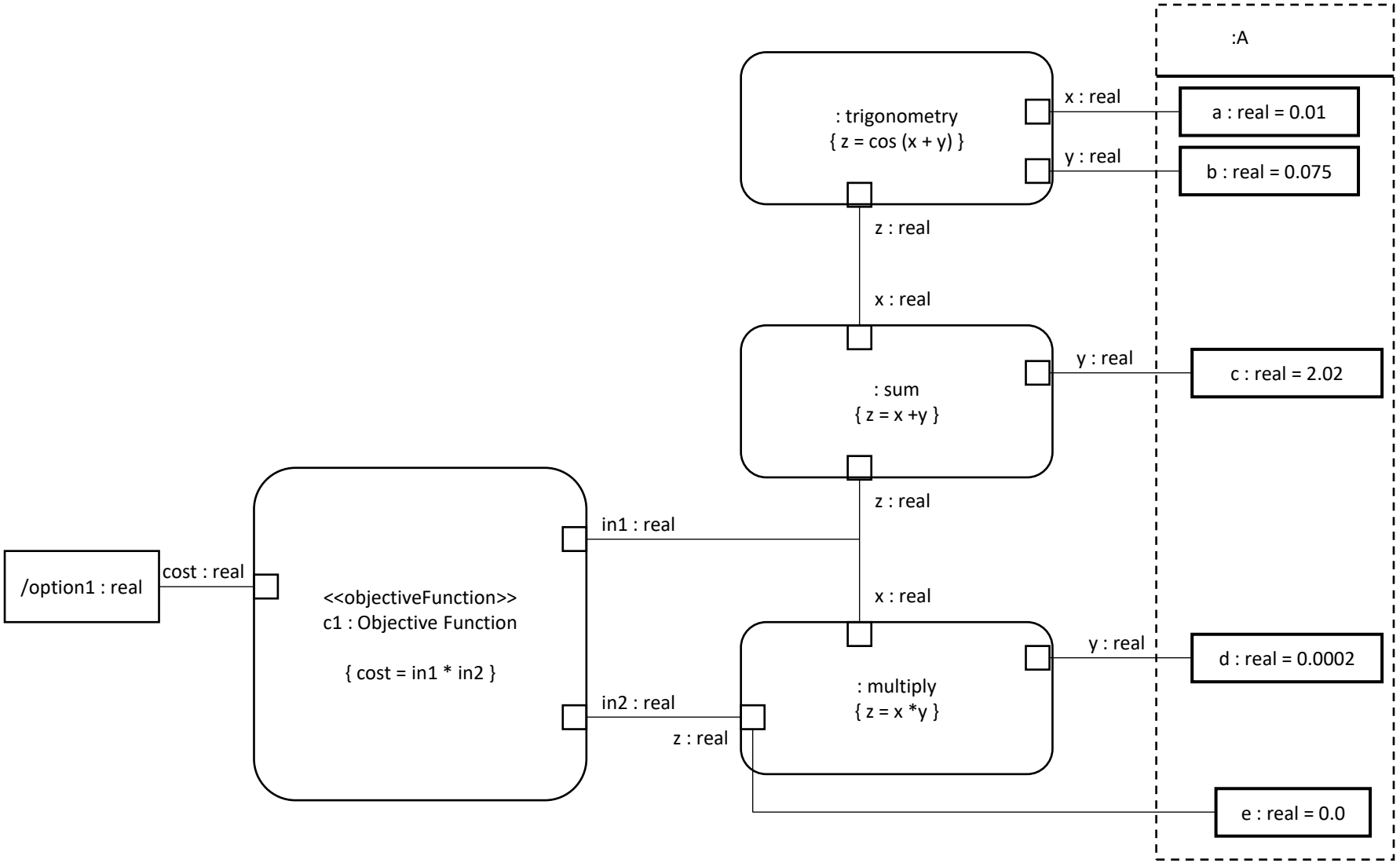
Za generowanie zestawu alternatywnych wejściowych parametrów ograniczających odpowiada **funkcja celowa** (objective function), inaczej nazywana optymalizacją albo funkcją kosztów.

Funkcja celowa jest szczególnym przypadkiem bloku ograniczeń, stosowany w połączeniu z miarami efektywności do przetwarzania wartościowań wejściowych parametrów ograniczających.

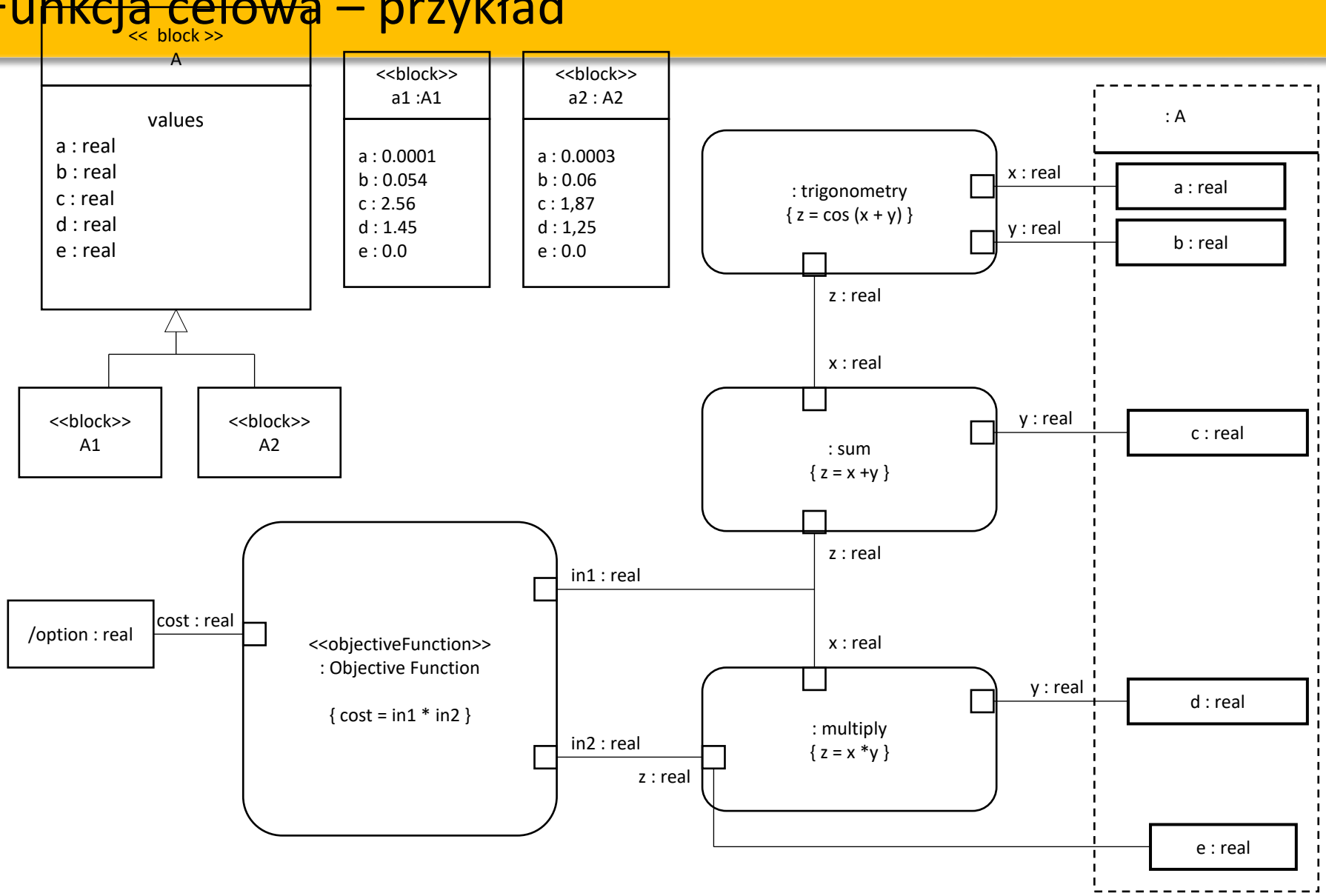
Funkcja celowa – przykład



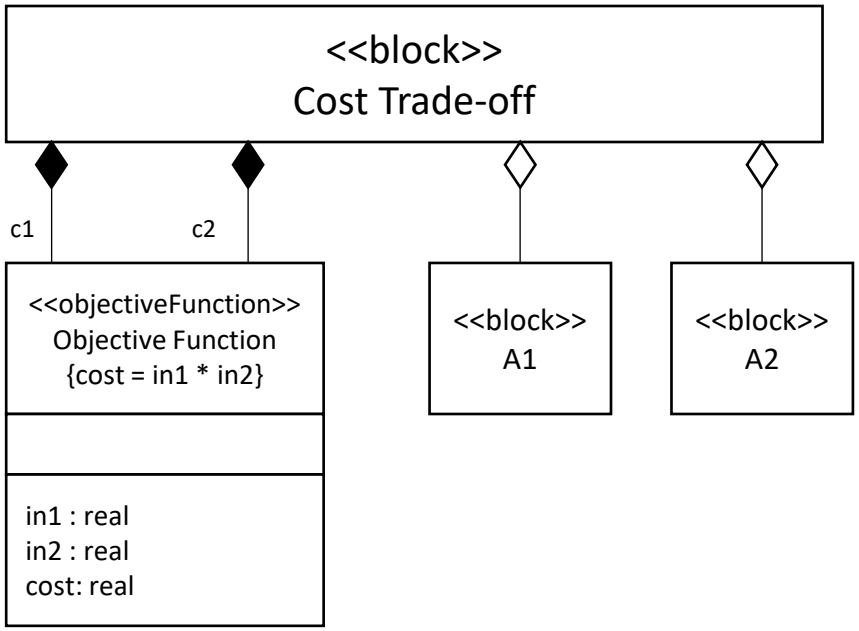
Funkcja celowa – przykład

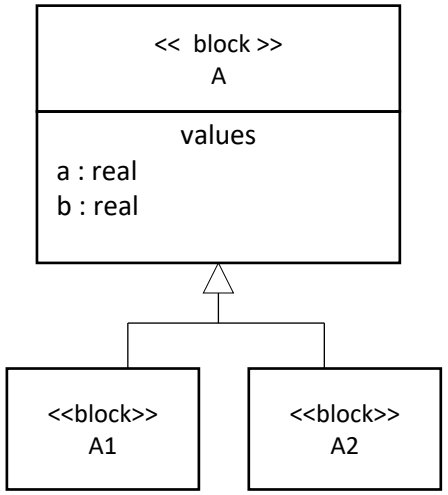


Funkcja celowa – przykład

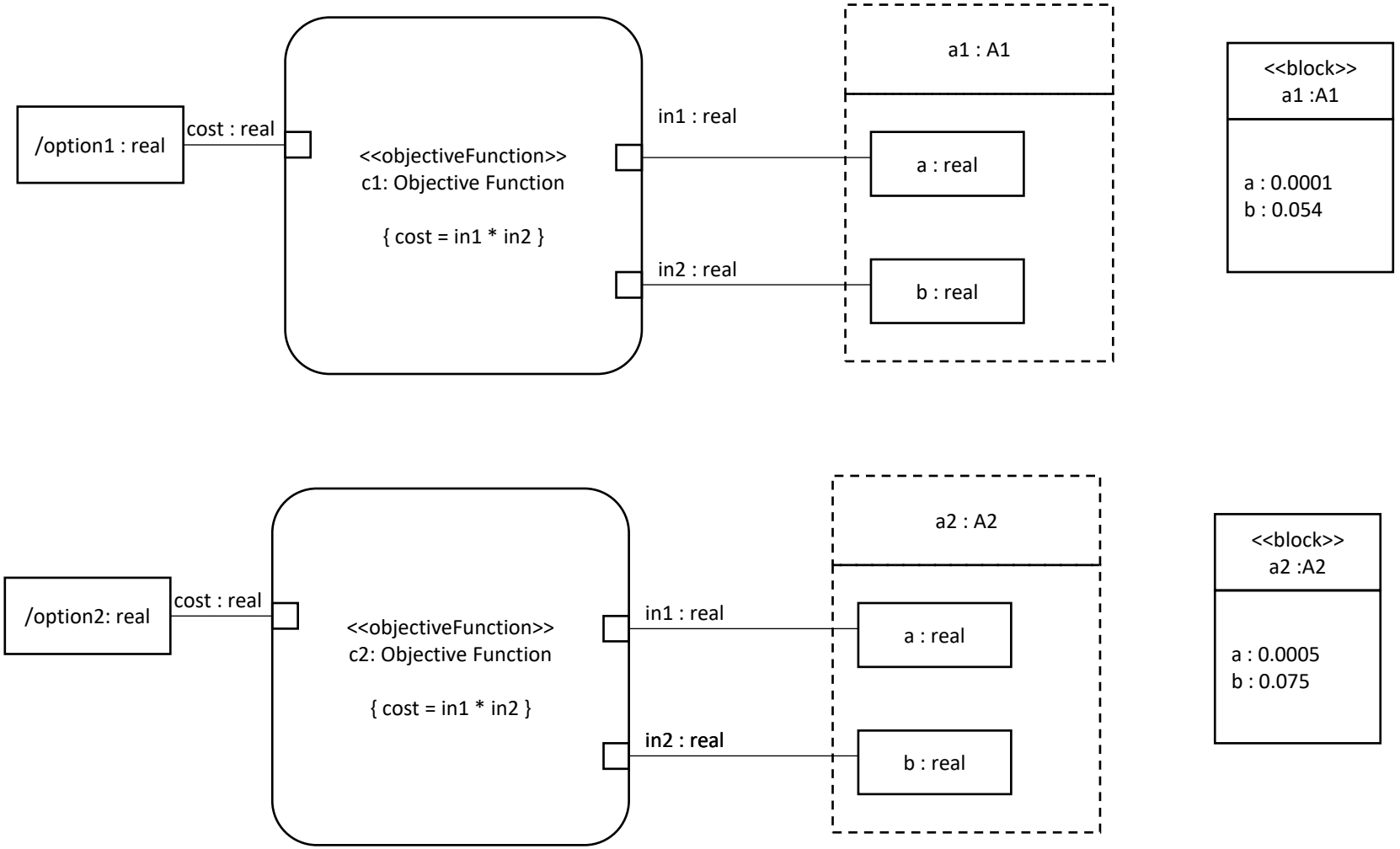


Alternatywa wariantowa



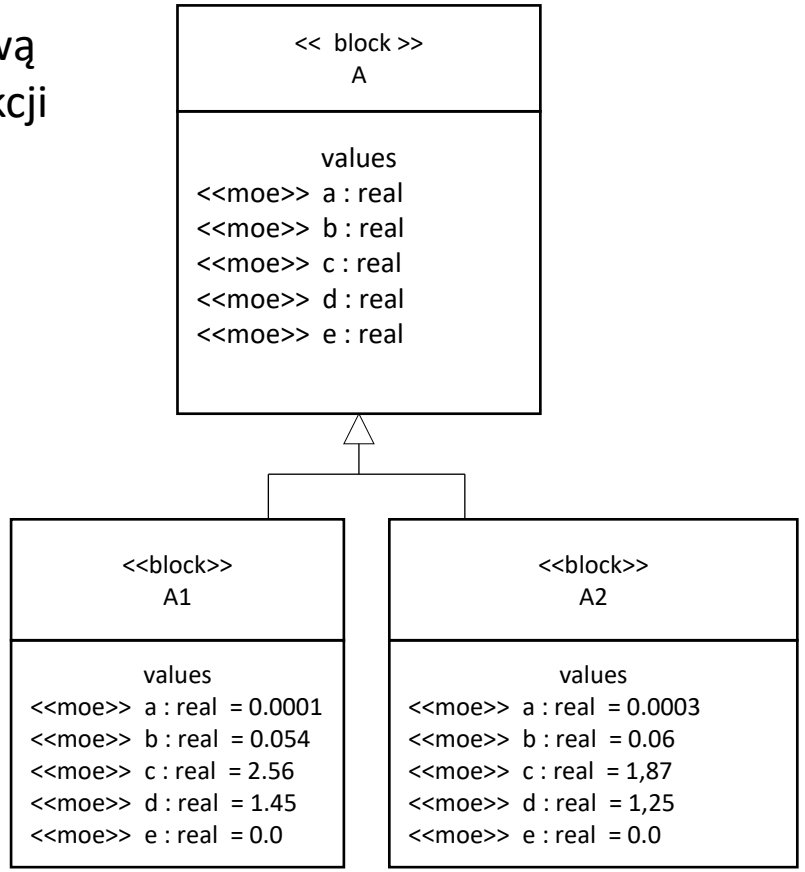


Alternatywa wariantowa



Miara efektywności

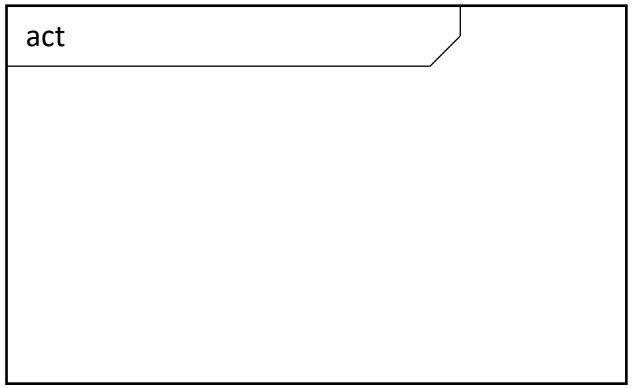
Parametrom bloków, które pełnią kluczową rolę z punktu widzenia zastosowania funkcji celowej, można przypisać stereotyp <<moe>> - measures of effectiveness. Są to parametry przydatne do analizy wariantowej



Rozszerzony diagram aktywności (act)

Rozszerzony diagram czynności – graficzne przedstawienie sekwencyjnych i/lub współbieżnych przepływów sterowania oraz danych pomiędzy uporządkowanymi ciągami czynności, akcji oraz obiektów. Odpowiada diagramowi czynności z UML, jednak jest istotnie zmodyfikowany.

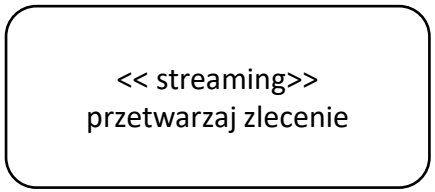
Używany jest do modelowania zachowania algorytmów, procesów i współdziałania poszczególnych elementów systemu i jego środowiska



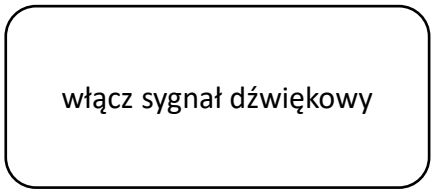
Operator sterowania zarządzający wartościami kontrolnymi

Przetwarzanie czynności

- **ciągłe** (standardowe)
- **strumieniowe** (do SysML włączono elementy wspierające modelowanie systemów strumieniowych)

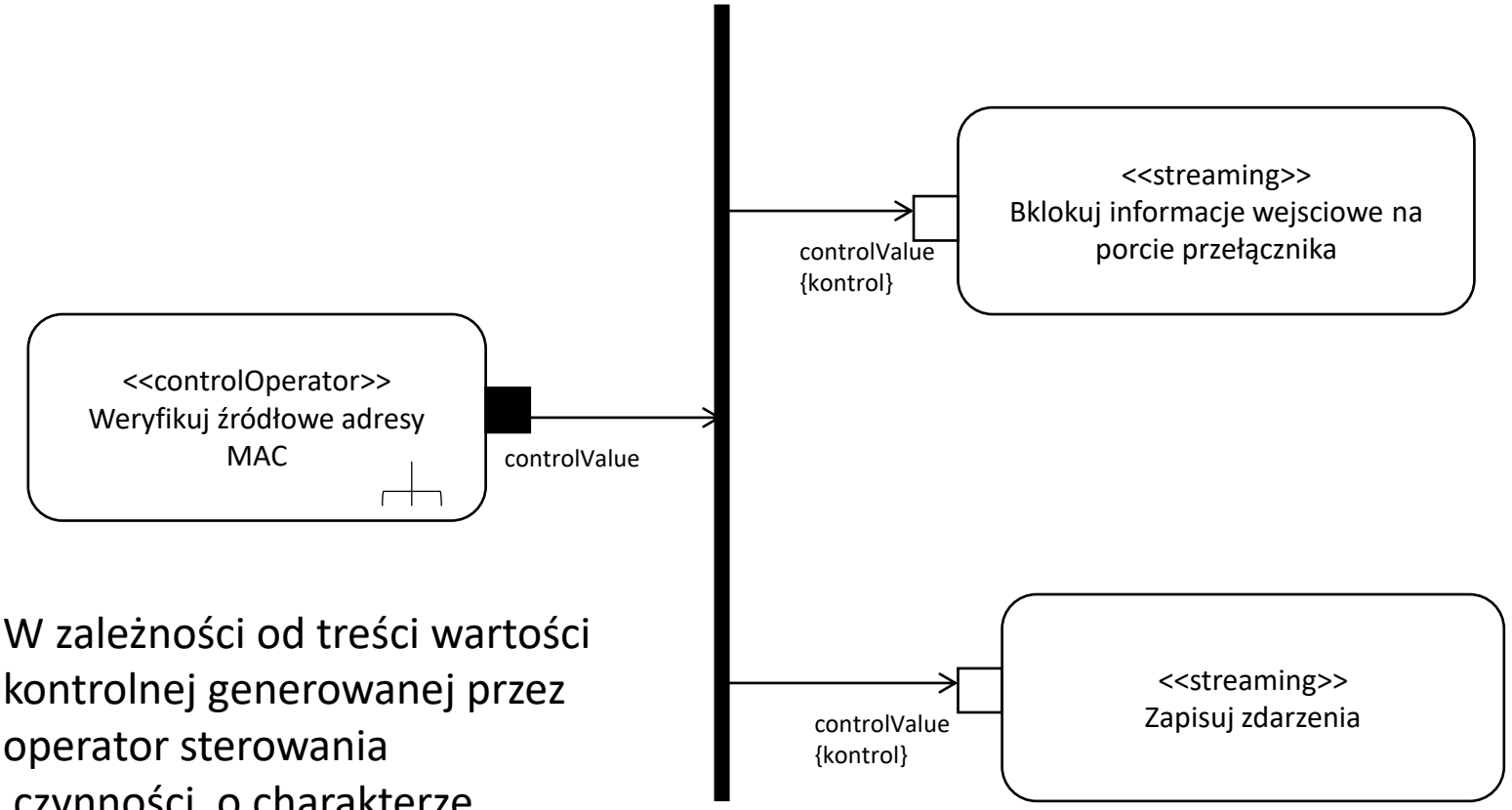


Czynności uczestniczące w przetwarzaniu strumieniowy



Czynności uczestniczące w przetwarzaniu ciągłym

Wartości kontrolne



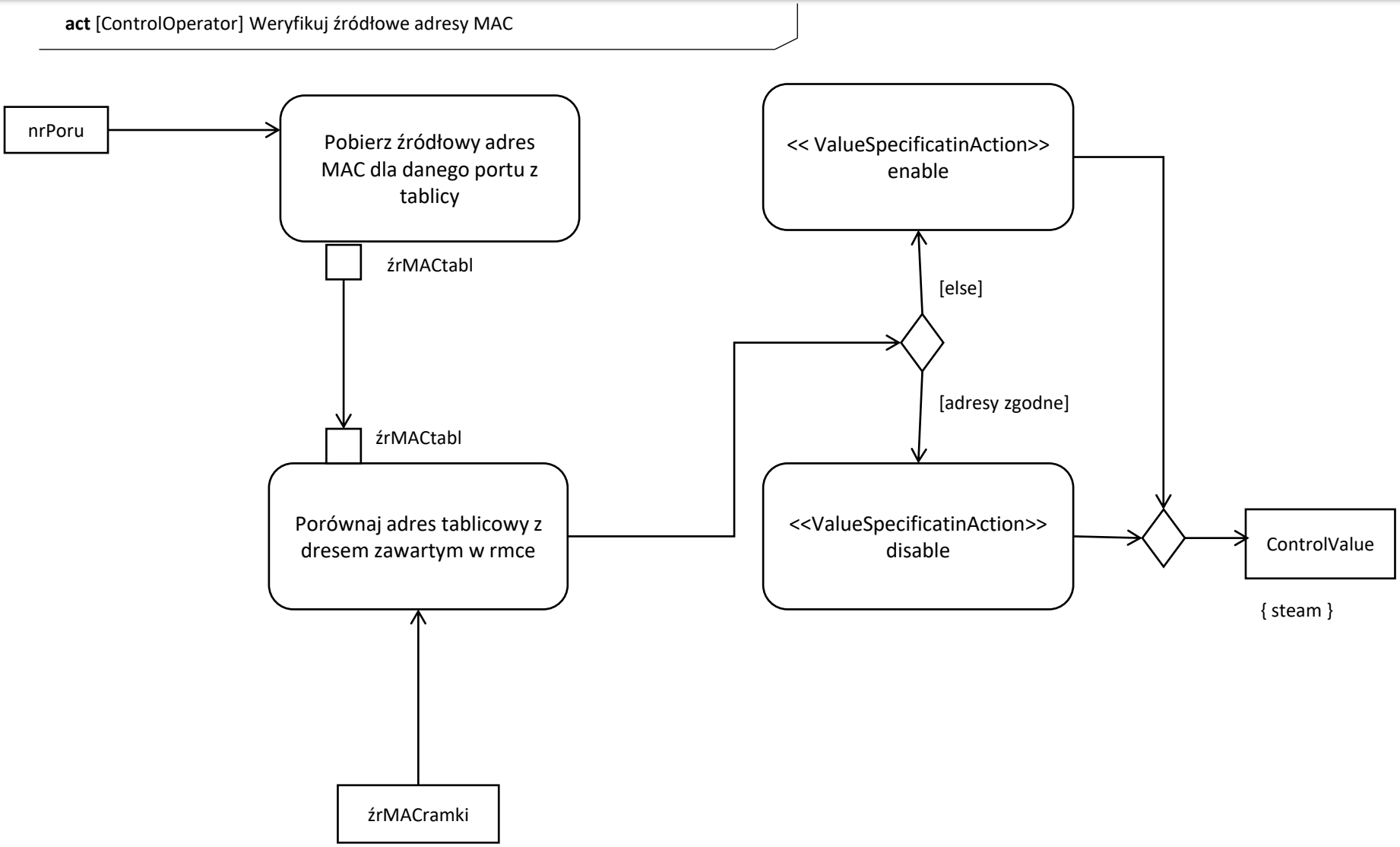
W zależności od treści wartości kontrolnej generowanej przez operator sterowania czynności o charakterze strumieniowym mogą zostać aktywowane albo dezaktywowane

Operator sterowania

- Operator do właściwego funkcjonowania potrzebuje dwóch parametrów:
  - nrPortu
  - źrMACramki

- Operator najpierw wykonuje czynność pobierz źródłowy adres, a następnie porównuje adres tablicowy z adresem zawartym w ramce
- Jeśli w tablicy znajduje się odpowiednik adresu zawartego w nagłówku ramki, wartość kontrolna przyjmuje wartość **disable**, w przeciwnym przypadku **enable**

Operator sterowania – przykład



Buforowanie danych i sterowania

- Przekaźniki danych mogą mieć ograniczoną zdolność buforowania nadchodzących danych i wartości kontrolnych.
- Nie ma to zasadniczego znaczenia w przypadku systemów nieoperujących na strumieniach danych – aby upewnić się, że czynności otrzymają odpowiedni zestaw danych wejściowych można się posłużyć znanym z UML buforem centralnym
- W systemach strumieniowych czynność może obsługiwać dane przychodzące w czasie rzeczywistym, w tym celu wejściowe przekaźniki danych zaopatrzono w bufor wewnętrzny.

Buforowanie danych i sterowania

Przekaźniki danych (z punktu widzenia przechowywania danych):

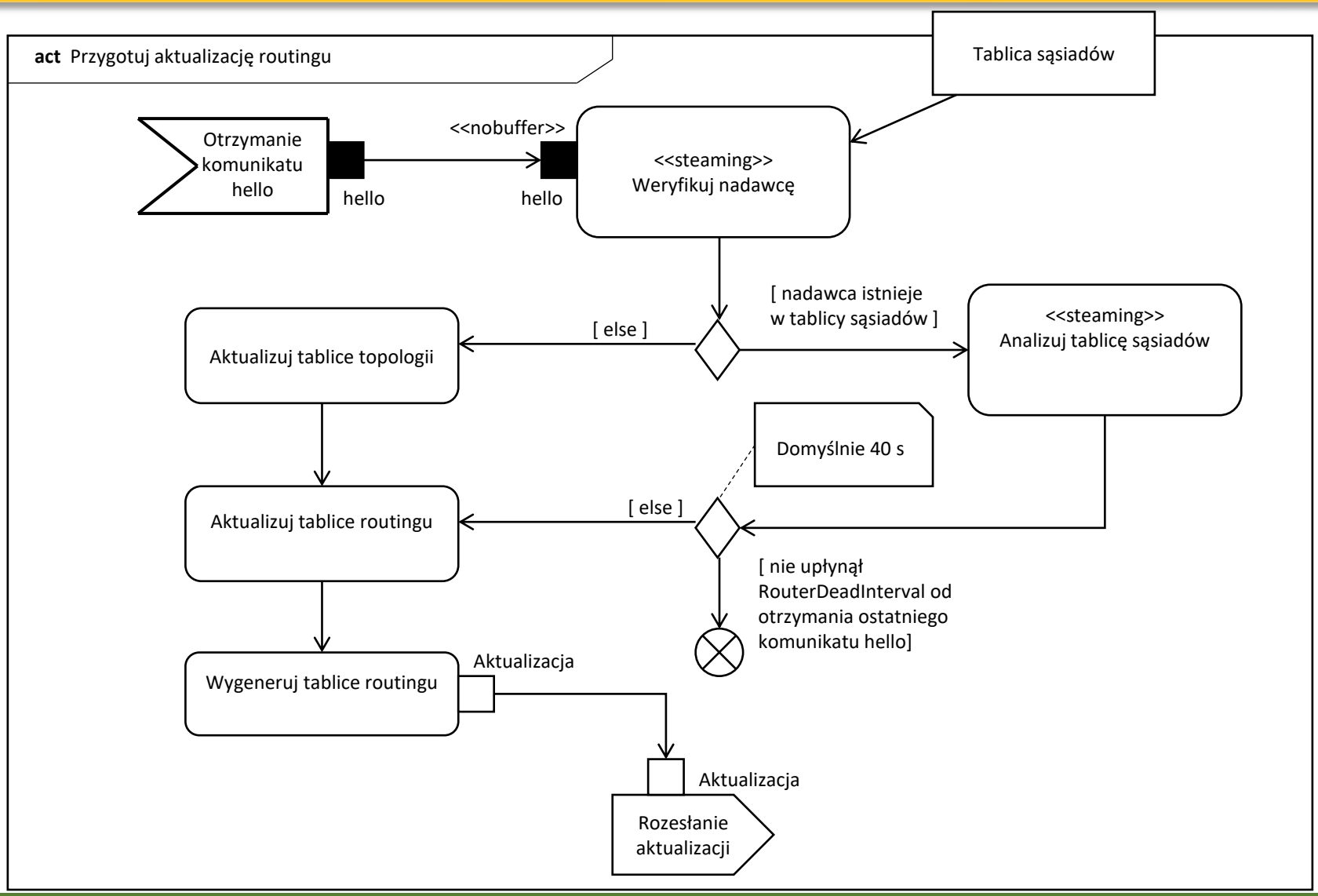
- niebuforujące przekaźniki danych **no buffer**
- nadpisujące przekaźniki danych **overwrite**

Przekaźnik niebuforujący

Użycie wskaźnika niebuforującego wskazuje na możliwość odmowy przyjmowania danych przez czynność

w przypadku przypisania stereotypu <<nobuffer>> do wejściowego przekaźnika danych, kasuje on dane przychodzące, o ile czynność nie jest ich w stanie od razu obsłużyć

Przekaźnik niebuforujący



Przekaźnik nadpisujący

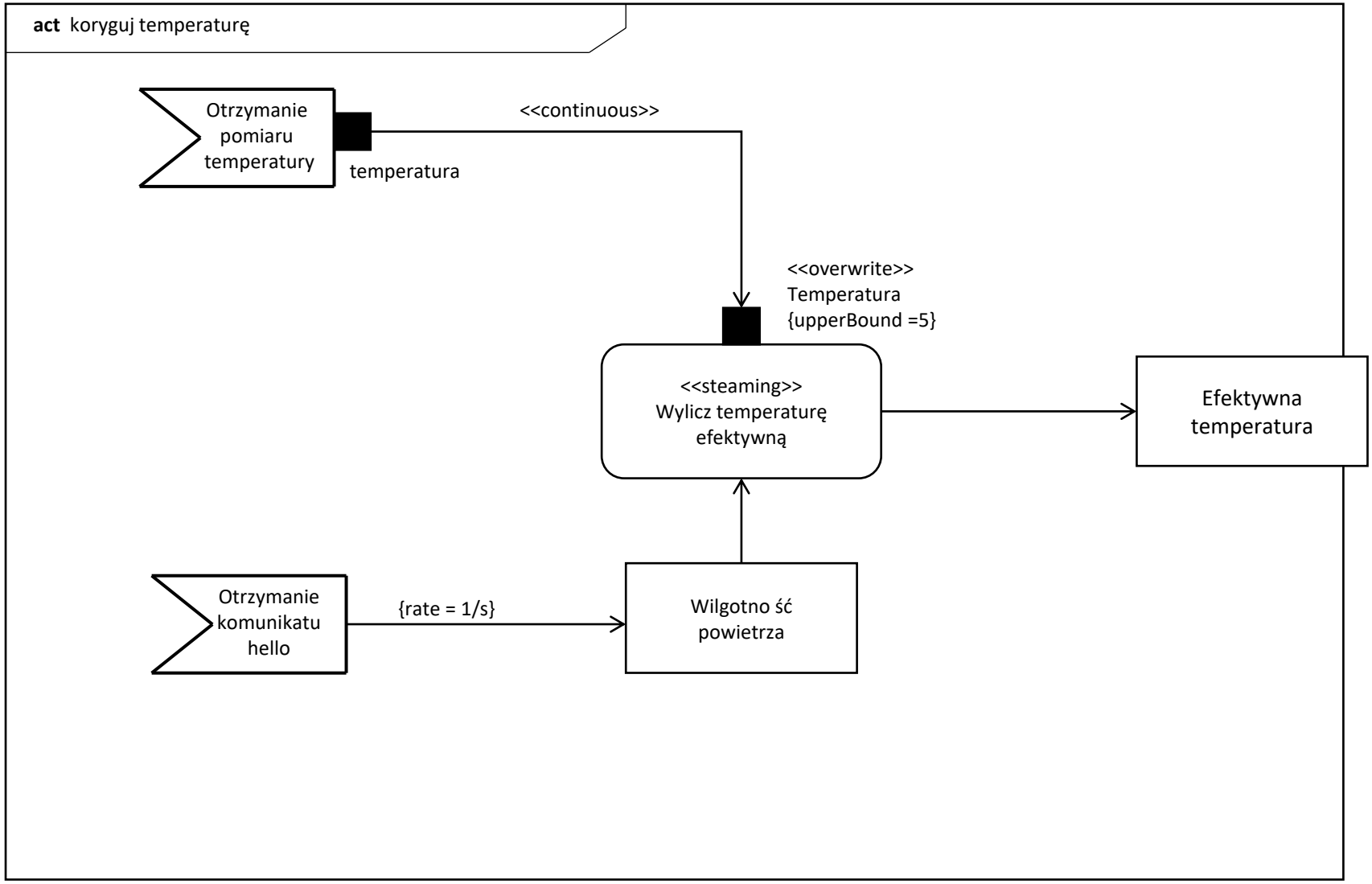
- Przekaźnik nadpisujący to bardziej złożona odmiana przekaźnika niebuforującego
  - przyjmuje się istnienie wewnętrznego bufora o określonej wielkości
  - W momencie osiągnięcia maksymalnego rozmiaru bufora, napływające dane/sterowanie zastępują dane/sterowanie, które są w tym momencie w buforze

Przekaźnik nadpisujący

Cechy nadpisującego przekaźnika danych:

- Rozmiar bufora (upperBound)
  - domyślnie: 1
  - wartość 0 stworzy przekaźnik niebuforujący
- Mechanizm obsługi kolejki (ordering)
  - Domyślnie FIFO

Przekaźnik nadpisujący



Przekaźnik opcjonalny

Stereotyp <<optional>> związany jest z wymagalnością przekaźników danych oraz parametrów czynności, wówczas:

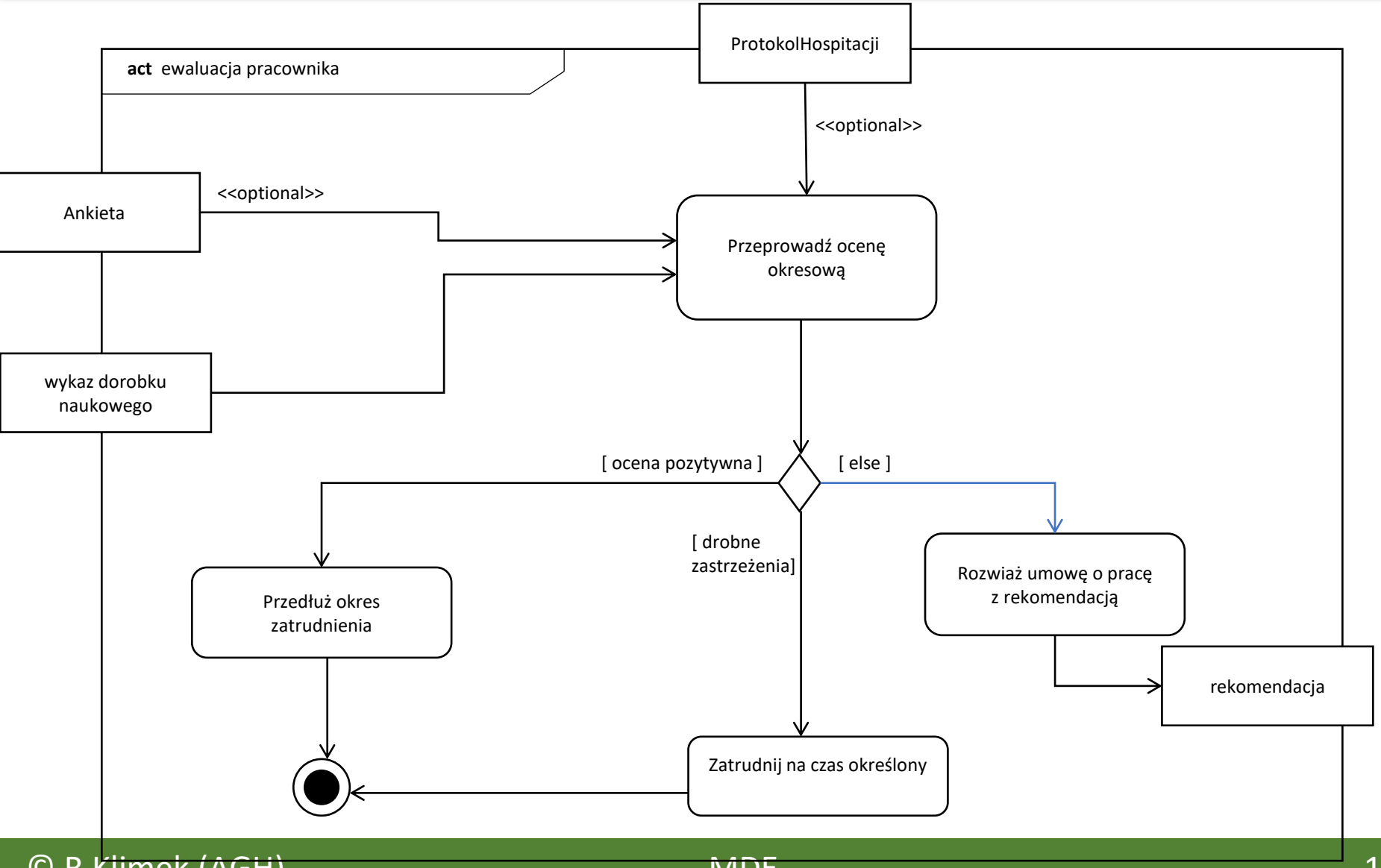
- dane z przekaźnika lub parametru nie są niezbędne do rozpoczęcia/ zakończenia czynności
- brak stereotypu oznacza, że przepływ danych jest obligatoryjny

Przekaźnik opcjonalny

Znaczenie parametru opcjonalnego można rozszerzać poprzez przypisanie do stereotypowanej kategorii modelowania dodatkowego ograniczenia

- powinno zawierać minimalną liczbę jednostek danych/sterowania (wymaganą przez daną czynność) poprzedzone słowem kluczowym *lowerBound*
- wartość *lowerBound* nie może być większa od rozmiaru bufora wewnętrznego *upperBound*

Przekaźnik opcjonalny – przykład



Przepustowość

Przepustowość jest ściśle związana ze strumieniowym przepływem danych

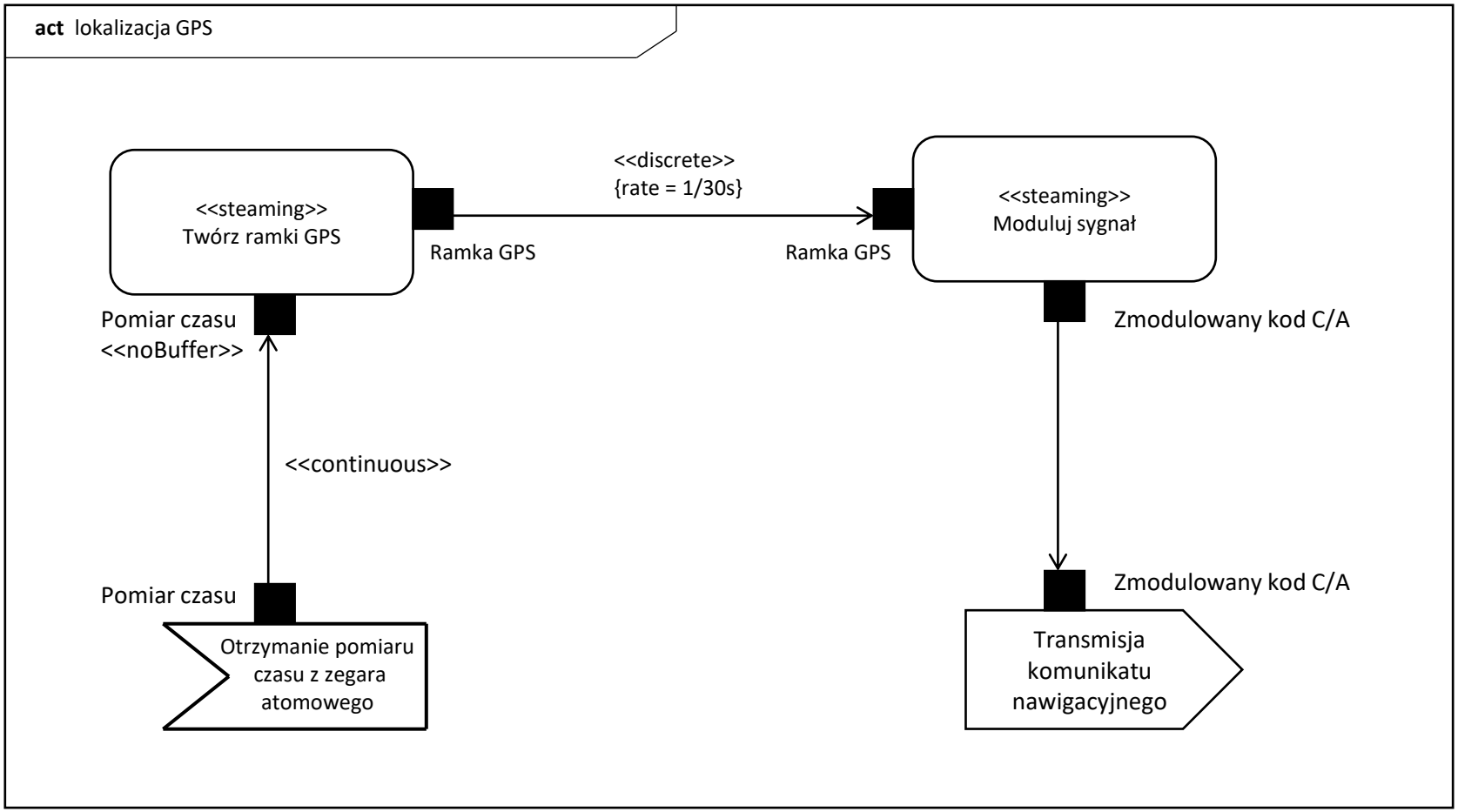
Przepustowość definiuje liczbę obiektów/wartości transmitowanych przez przepływ w jednostce czasu(dodanie słowa kluczowego *rate.*

Przepustowość dotyczy przepływu danych i przepływu obiektów

Definiując przepustowość należy określić charakter przepływu strumieniowego:

- dyskretny <<discrete>> jeżeli można określić takt pomiędzy jednostkami danych
- ciągły <<continuous>> jeżeli czas pomiędzy przesyłanymi danymi jest nieistotny

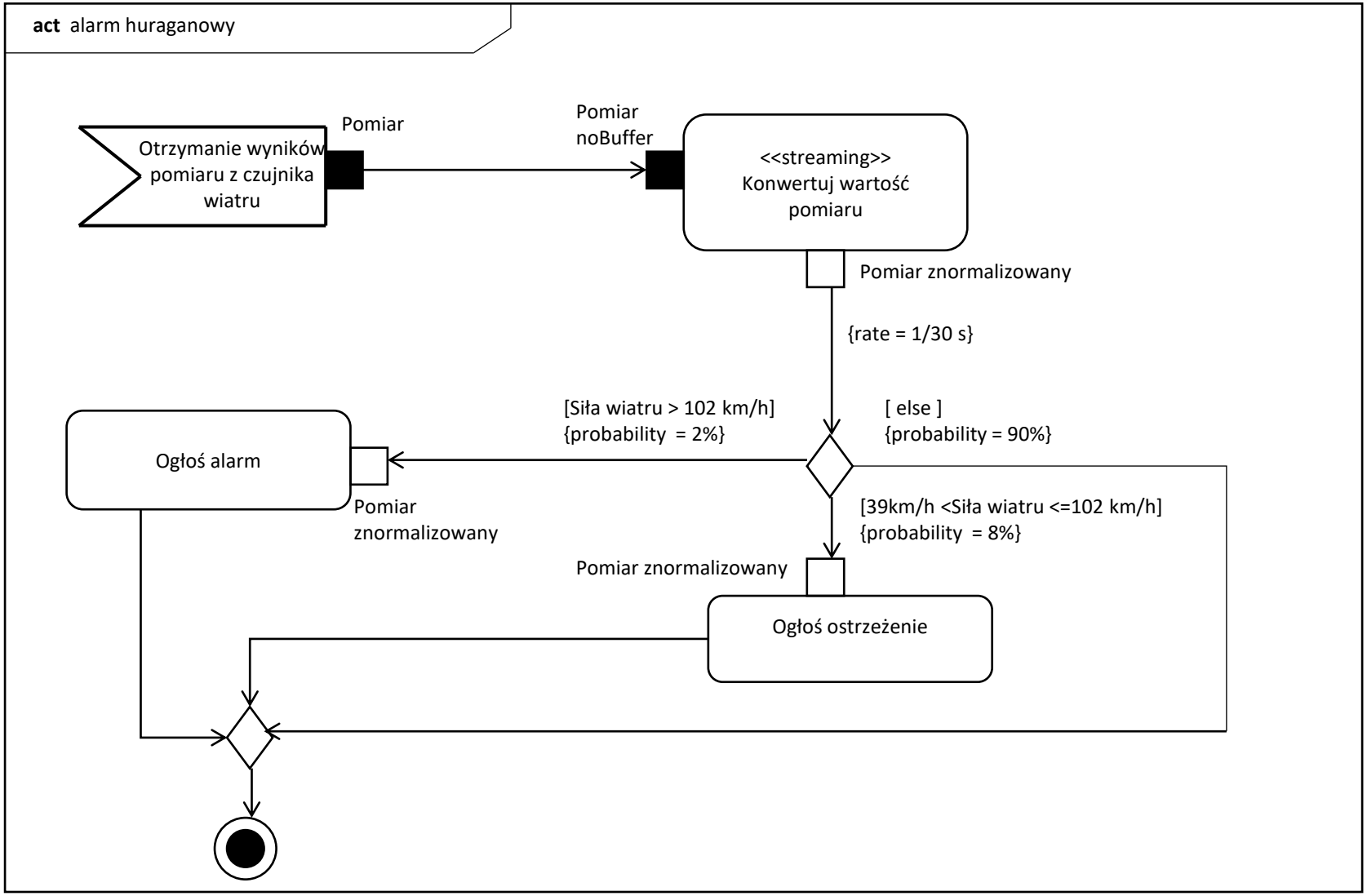
Przepustowość – przykład



Prawdopodobieństwo

- Przepływy wyjściowe bloku decyzyjnego mogą zostać zrealizowane z określonym prawdopodobieństwem. (dodanie słowa kluczowego *probability*)
- Prawdopodobieństwa wyjściowe sumują się do 1 (100%)
- Zasada ta odnosi się również do przepływu obiektów i zestawów przekaźnikowych (jeżeli obiekt posiada co najmniej dwa przepływy wyjściowe)

Parametr opcjonalny



Prewarunki i postwarunki

SysML pozwala na określenie lokalnych warunków wstępnych *preconditions* i końcowych/wyjściowych *postconditions* realizowanych czynności

- warunki wstępne – lista kryteriów, które muszą być spełnione, aby zainicjować wykonywanie czynności
- warunki końcowe – lista kryteriów, które powinny być spełnione, na skutek wywołania czynności

Prewarunki i postwarunki – definiowanie

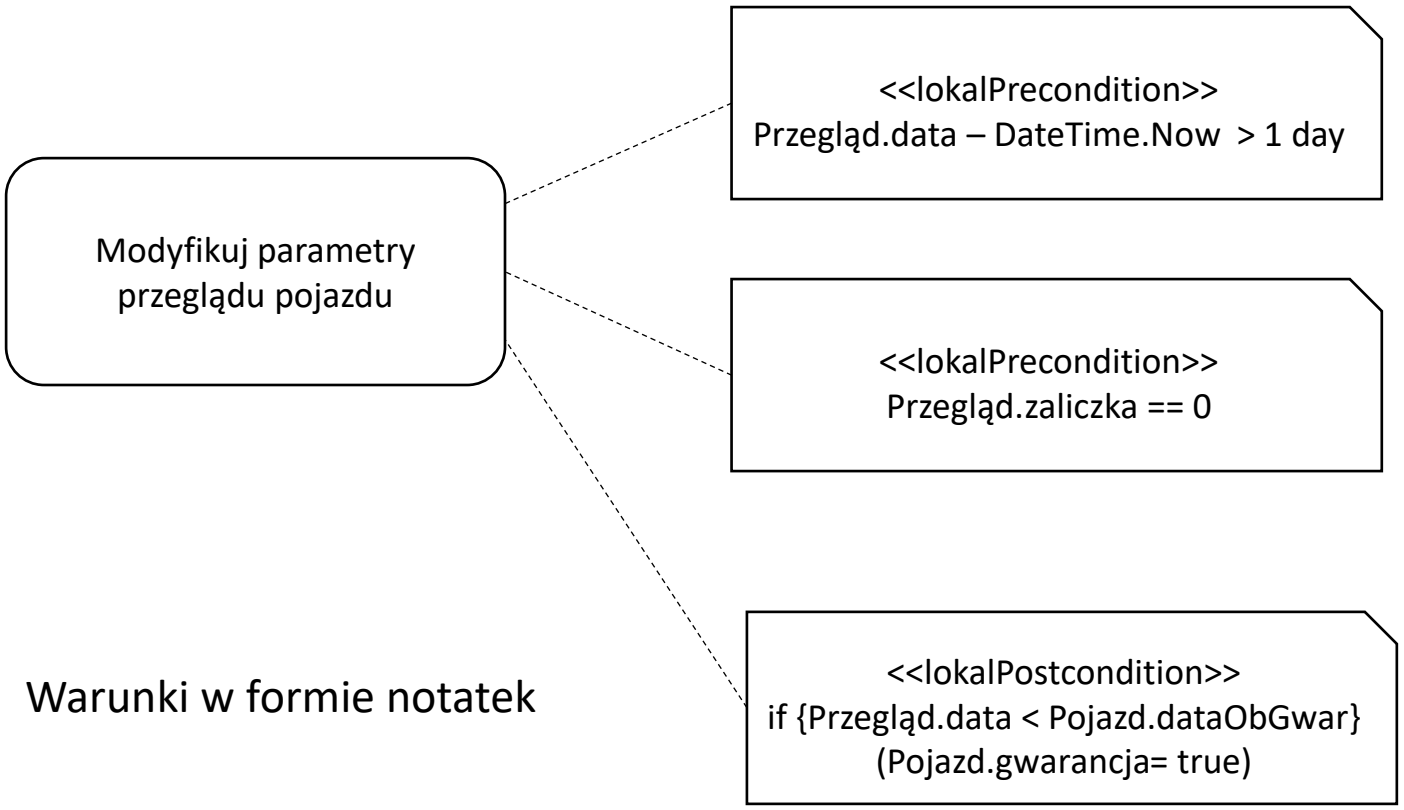
Definiowanie warunków

- nieformalnie (opisowo)
- w dowolnym języku ograniczeń (np. OCL)

Warunki umieszczane są na diagramie

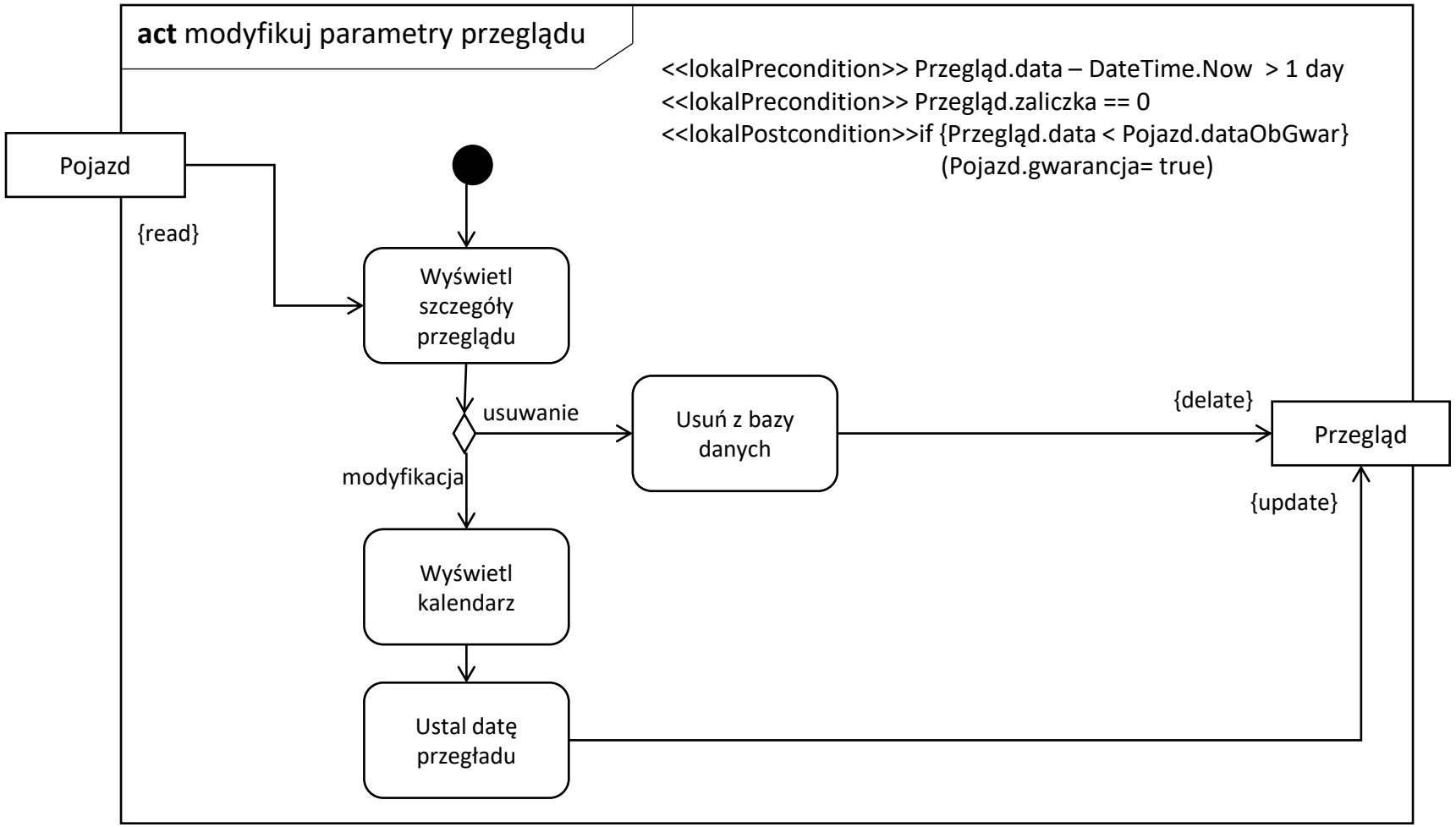
- w formie notatek, jeżeli dotyczą konkretnej czynności
- w prawym górnym rogu diagramu, jeżeli dotyczą całego diagramu

Prewarunki i postwarunki – przykład



Warunki w formie notatek

Prewarunki i postwarunki – przykład



Blokowa notacja czynności

- Czynności mogą zostać pokazane także jako stereotypowane bloki (mogą być wówczas umieszczane bezpośrednio na diagramach struktury, np. w celu wyspecyfikowania alokacji)

- Możliwość pokazania hierarchii czynności za pomocą agregacji (dekompozycja czynności)
  - Zakończenie czynności-całości jest możliwe po zakończeniu wykonywania jej czynności-całości
  - Liczebność pod-czynności domyślnie jest przyjmowana jako 0..* - w szczególnym przypadku żadna z nich może nie być wykonana

Blokowa notacja czynności