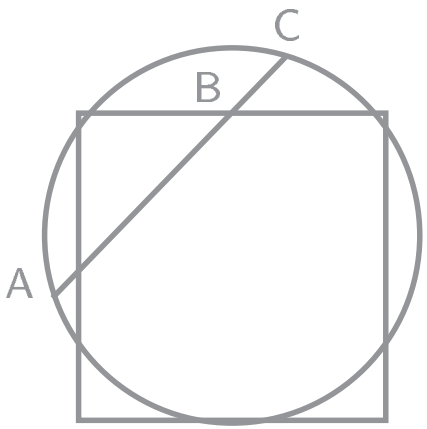
Inżynieria oprogramowania

Radosław Klimek

2015-23



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

1 Metodologia OMT

Metodologia OMT (Object-modeling Technique)

Pieter BRUEGHEL: *Ślepcy prowadzący ślepców*

Metodologia OMT

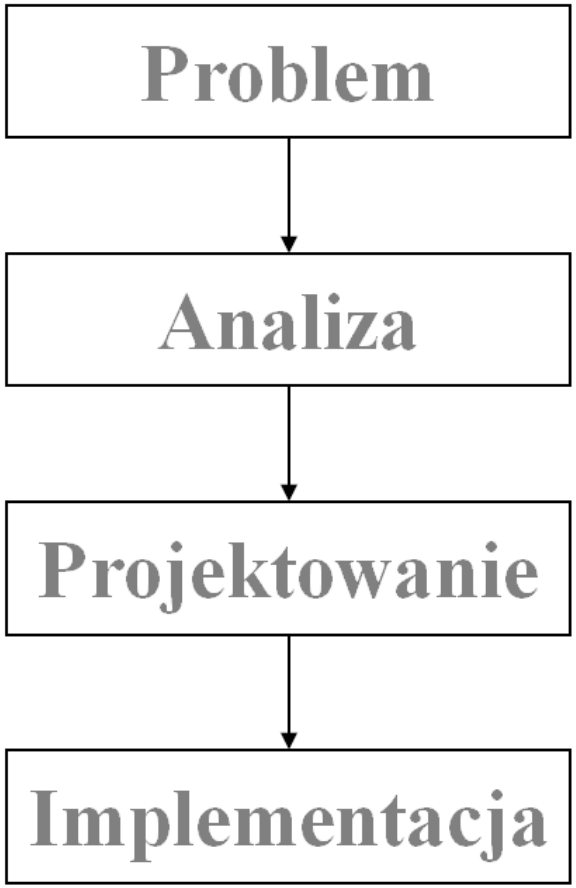
Metodyka analizy i projektowania obiektowego **OMT** zaproponowana poprzez Jamesa Rumbaugh i innych autorów:

- metodyka wprowadza trzy perspektywy: schemat (klas) obiektów, schemat dynamiczny i schemat funkcjonalny (wariant modelu przepływu danych),
- charakteryzuje się sporym poziomem abstrakcji, nie schodzi zbytnio na poziom projektowania i implementacji,
- użytkowana terminologia jest intuicyjna, choć nie zawsze jasna,
- OMT jest jednym z bezpośrednich poprzedników UML.

James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, William Lorensen. *Object-Oriented Modeling and Design*. Prentice Hall 1994.

Podstawowe założenia OMT

- Postawienie problemu.
- Zrozumienie wymagań – **analiza**.
- Planowanie rozwiązania – **projektowanie**.
- **Implementacja** w języku programowania.



Charakterystyka OMT

Podejście projektowe bazujące na OMT opiera się na procesie modelowania dziedziny aplikacji:

- identyfikuje i adoptuje byty świata rzeczywistego jako elementy modelu domeny,
- koncentruje się tylko na istotnych właściwościach rozwiązywanego problemu,
- kolejne etapy projektowania uszczegóławiają model wprowadzony w fazie analizy.

Charakterystyka OMT

Podejście projektowe bazujące na OMT opiera się na procesie modelowania dziedziny aplikacji:

- identyfikuje i adoptuje byty świata rzeczywistego jako elementy modelu domeny,
- koncentruje się tylko na istotnych właściwościach rozwiązywanego problemu,
- kolejne etapy projektowania uszczegóławiają model wprowadzony w fazie analizy.
- **Nie wprowadza ograniczeń wynikających z nieelastyczności języków programowania – końcowa realizacja może być wyrażona w języku obiektowym, proceduralnym, implementacji struktury bazy danych, jak również implementacji sprzętowej.**

Analiza systemowa

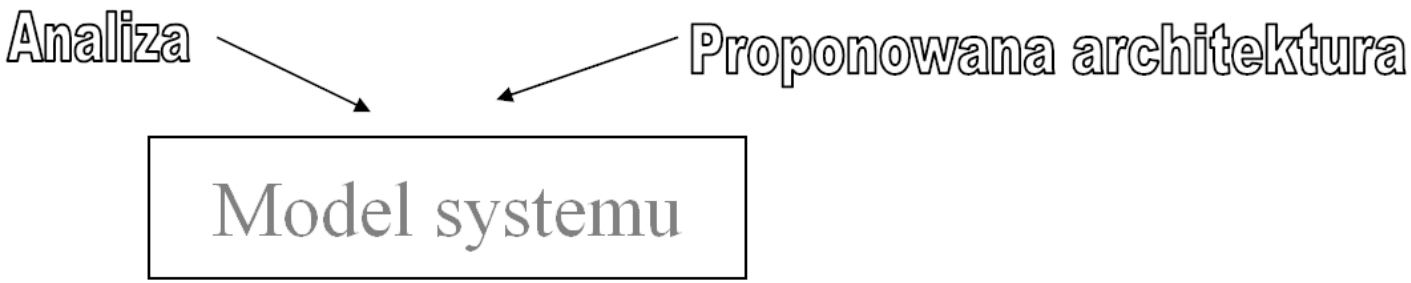
Założenia i cele:

- koncentracja na tym, **co** ma być zrobione, a nie **jak** ma być zrobione,
- modelowanie sytuacji świata rzeczywistego i wskazanie najważniejszych ich właściwości,
- model powinien zostać zweryfikowany przez ekspertów aplikacji, a nie programistów,
- nie używa pojęć związanych z implementacją,
- posługuje się pojęciami z zakresu dziedziny aplikacji, a nie struktur danych,
- stanowi punkt wyjścia kolejnych etapów tworzenia oprogramowania.

Projektowanie systemowe

Założenia i cele:

- ustalenie architektury systemu, np. **wielowarstwowa**,
- dekompozycja systemu na podsystemy,

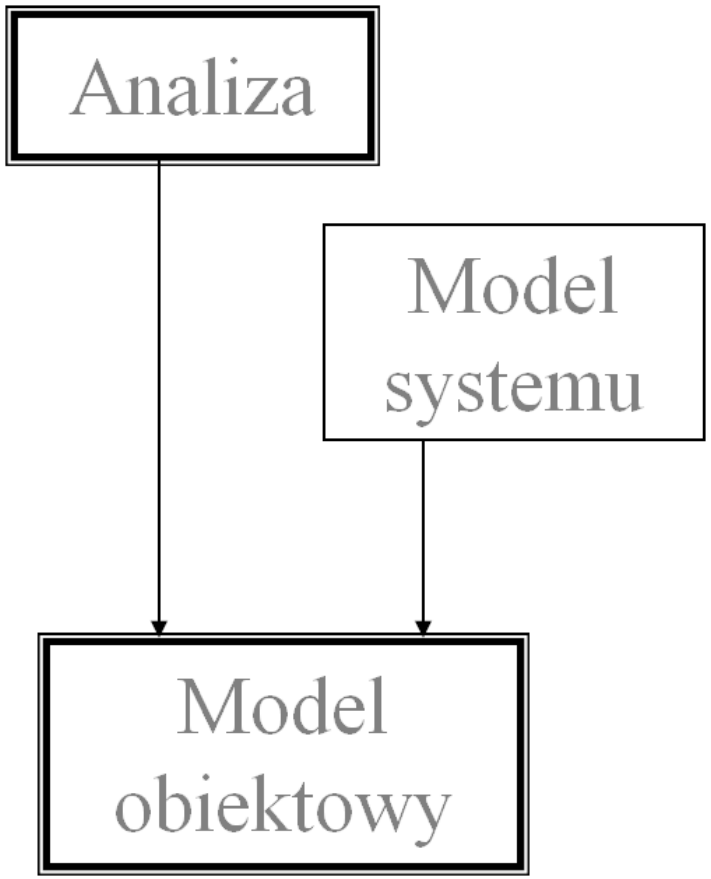


- wybór strategii rozwiązania problemu,
- decyzje odnośnie wydajności, skalowalności i optymalności wybranych wskaźników,
- rozsądna alokacja zasobów.

Projektowanie obiektowe

Zasady:

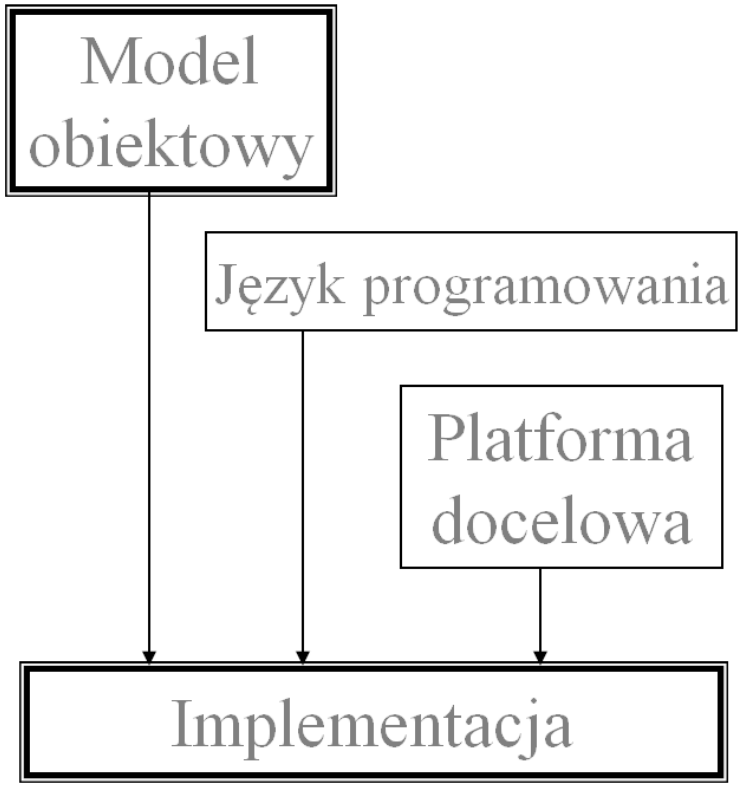
- budowa i projektowanie modelu bazującego na modelu domeny, ale wzbogaconym o szczegóły implementacyjne,
- myślenie w kategoriach struktur danych i algorytmów, optymalizacja wybranych właściwości,
- obiektowy model dziedziny i model komputerowy są te same, ale usytuowane w innych wymiarach (używają tej samej notacji).



Implementacja

Zasady:

- translacja modelu obiektowego z poprzedniego kroku do języka programowania, bazy danych lub implementacji sprzętowej,
- implementacja powinna być automatyczna z minimum nakładu pracy,
- opwinna być elastyczna i rozszerzalna.



Pierwsze wnioski

- Zachowanie struktury modeli pomiędzy etapami projektowania – w kolejnych krokach uszczegóławiane są elementy modelu dziedziny.
- Zróżnicowane są tylko poziomy abstrakcji.
- Część struktur danych wprowadzana jest sztucznie (np.: lista, drzewo, mapa) i nie ma odpowiednika w rzeczywistości.
- Realizuje podejście „top-down” – nie jest metodą iteracyjną.

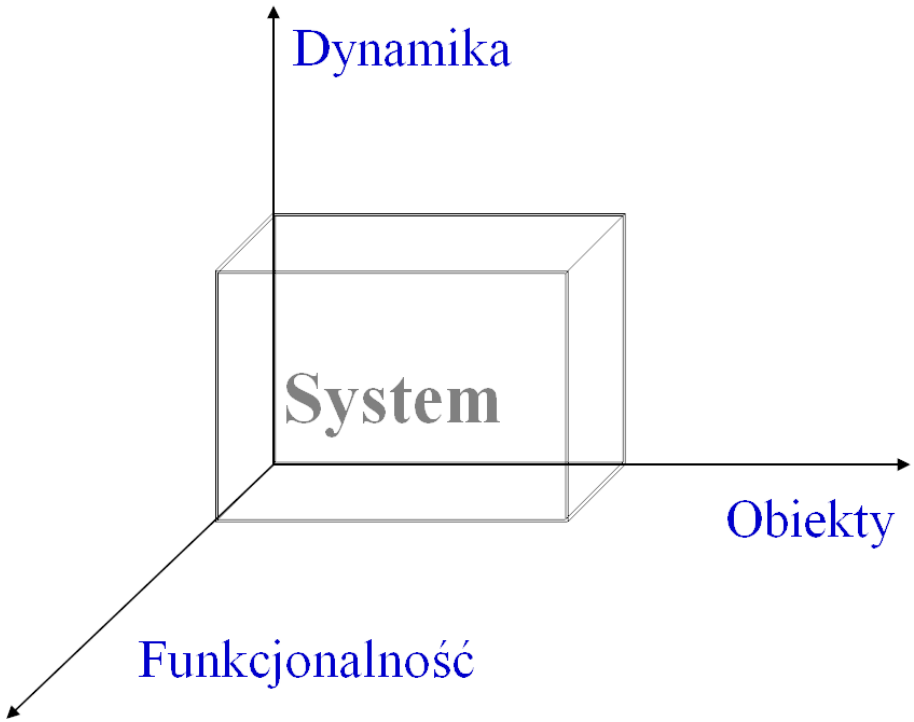
Pierwsze wnioski

- Zachowanie struktury modeli pomiędzy etapami projektowania – w kolejnych krokach uszczegóławiane są elementy modelu dziedziny.
- Zróżnicowane są tylko poziomy abstrakcji.
- Część struktur danych wprowadzana jest sztucznie (np.: lista, drzewo, mapa) i nie ma odpowiednika w rzeczywistości.
- Realizuje podejście „top-down” – nie jest metodą iteracyjną.
- **Nie obejmuje fazy testowania aplikacji**.

Modele w OMT

Założenia:

- budowane są trzy modele:
  1. model obiektowy,
  2. model dynamiki,
  3. model funkcjonalny,
- powyższe modele są wzajemnie **ortogonalne** – opisują ten sam system z różnych, komplementarnych perspektyw,
- modele budowane są w fazie **analizy**.



Model obiektowy

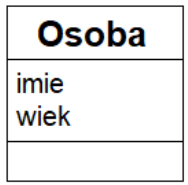
Założenia:

- wyraża statyczną strukturę obiektów, powiązań pomiędzy nimi i zobowiązań,
- określa podmioty operowania dla modeli dynamiki i modeli funkcjonalnych,
- najbardziej stabilny model procesu projektowania,
- składa się z diagramu obiektów w którym:
  - obiekty są wierzchołkami grafu,
  - krawędzie obrazują relacje pomiędzy obiektami.

Obiekty i klasy

Definicja

Klasa – grupuje obiekty o podobnych właściwościach, zachowaniu, relacji do pozostałych klas i podobnej semantyce. Klasa – grupuje obiekty o podobnych właściwościach, zachowaniu, relacji do pozostałych klas i podobnej semantyce.

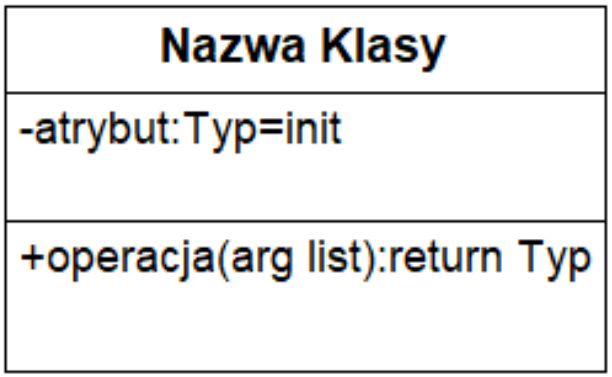


Definicja

Obiekt – rzeczownik, który ma sens w kontekście aplikacji. Obiekt "wie" do jakiej klasy przynależy.

Atrybuty i operacje

- **Atrybuty** – pamiętają stan obiektu klasy:
  - każdy atrybut posiada wartość dla każdej instancji obiektu,
  - nazwy atrybutów są unikalne w obrębie klasy,
  - atrybutami są typy proste nie posiadające identyfikacji.
- **Operacje** – funkcje lub transformacje, które może być wykonywana przez obiekt lub na obiekcie:
  - posiada powiązany obiekt jako obowiązkowy argument,
  - jest operatorem polimorficznym,
  - może posiadać dodatkowe argumenty.



Połączenia i związek

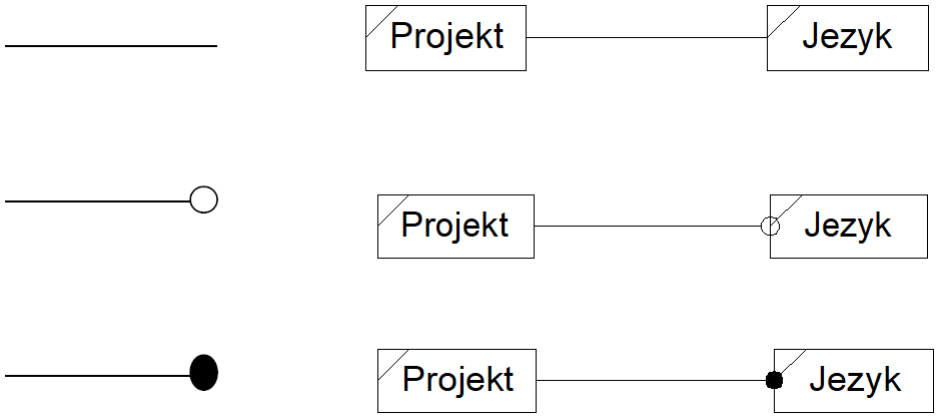
- Wyodrębnione elementy modelu dziedziny łączone są w **grafy instancji** obiektów i **diagramy klas**.
- **Połączenie** – fizyczne lub pojęciowe połączenie pomiędzy **instancjami** obiektów.

- **Związek** – grupa połączeń o jednakowej strukturze i semantyce – łączy **klasy** obiektów. Przeważnie są one dwukierunkowe.

Związki mnogościowe binarne

Specyfikują jak wiele instancji obiektów jednej klasy może być połączonych z pojedynczą instancją innej klasy:

- pojedyncze
- opcjonalne
- wielokrotne

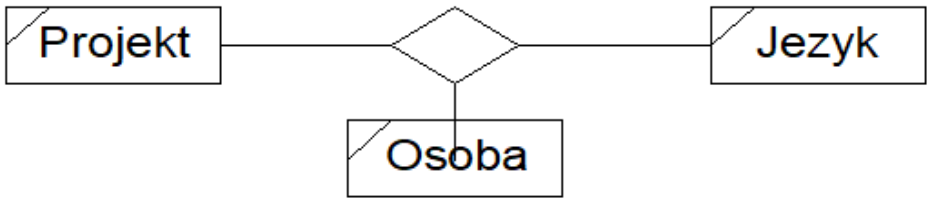


Na końcu powiązania może wystąpić liczba określająca jego liczebność.

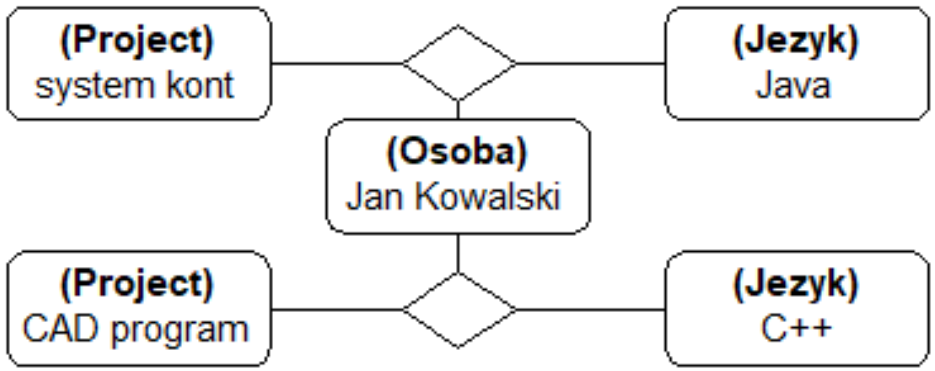
Związki wyższego rzędu

Powiązanie "ternary" jest jednostką niepodzielną – bez straty informacji nie daje się przedstawić jako zestaw powiązań binarnych.

- Diagram klas:

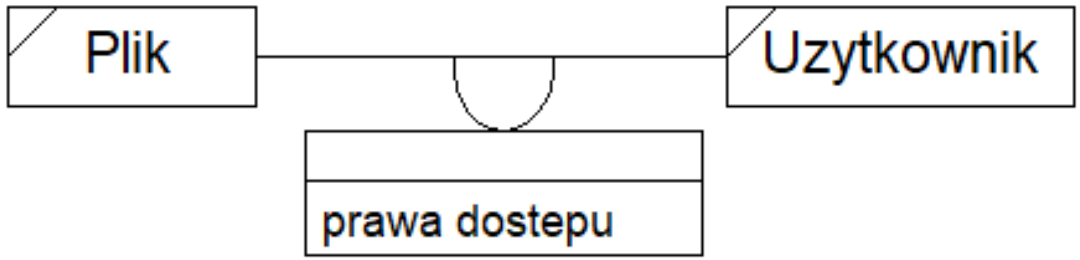


- Diagram obiektów:

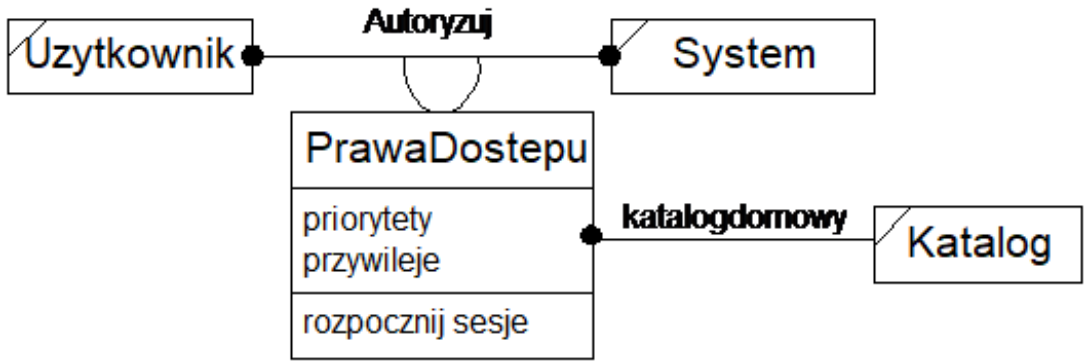


Atrybuty połączenia

- Podobnie jak **atrybut** jest własnością klasy, tak **atrybut połączenia** jest właściwością związku pomiędzy klasami.
- Atrybut połączenia jako własność:



- Atrybut połączenia jako klasa:

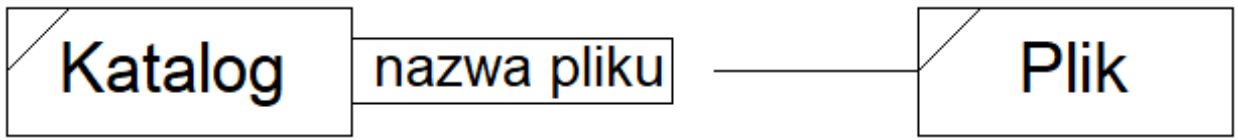


Kwalifikacja

Definicja

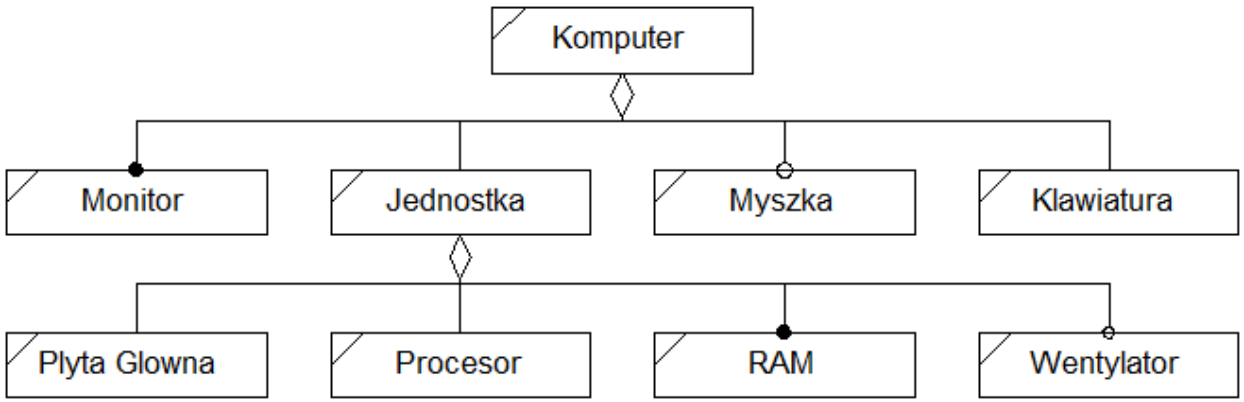
Kwalifikacja – specjalny atrybut redukujący wielokrotność powiązania – zmniejsza ilość kombinacji powiązania.

- Stosowany dla powiązań typu: jeden-do-wielu i wiele-do-wielu.
- Stanowi kontekst w którym nazwa nabiera sensu.
- Stosowana jest głównie przy przeszukiwania zbiorów.



Agregacja

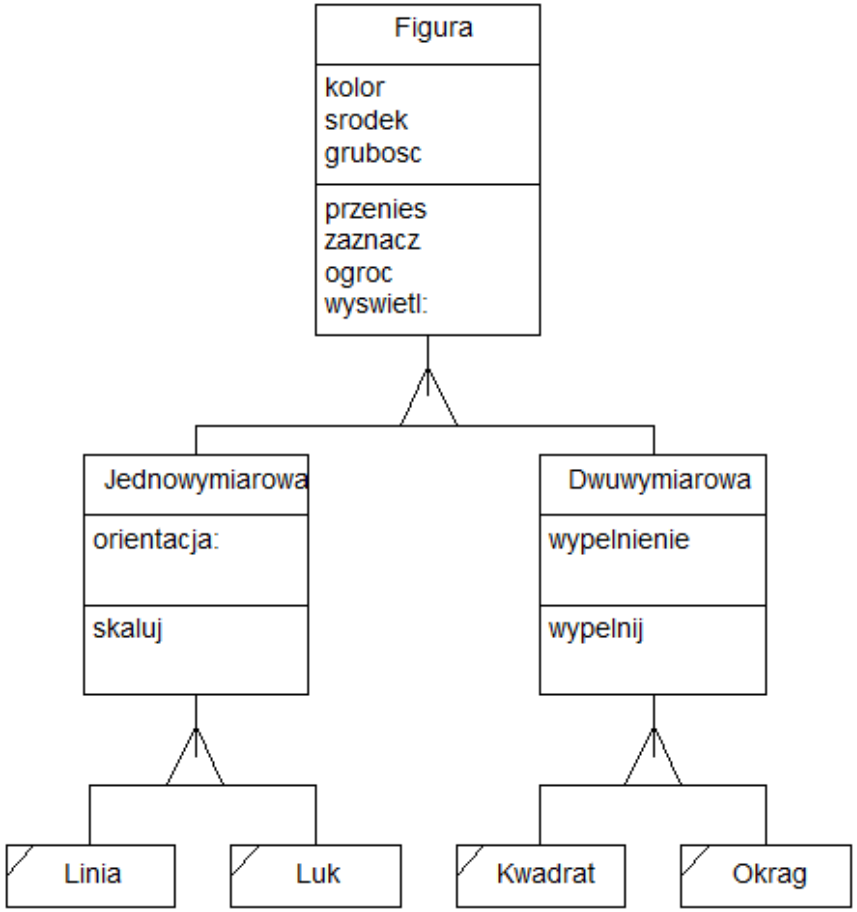
- Specjalna forma powiązania.
- Agregacja jest relacją zawierania i cechuje ją:
  - przechodniość,
  - antysymetria.



- Agregacja podkreśla, że jeden obiekt jest częścią innego – wyraża zależność ”ma”.

Dziedziczenie i generalizacja

- Dzielenie funkcjonalności z zachowaniem indywidualności klasy.
- Instancja klasy jest jednocześnie instancją wszystkich swoich przodków.
- Przechodniość pomiędzy poziomami dziedziczenia.
- Zastosowanie przy budowie modelu pojęciowego i implementacji.
- Dziedziczenie może być rozpatrywane jako rozszerzenie lub jako restrykcja klasy bazowej.



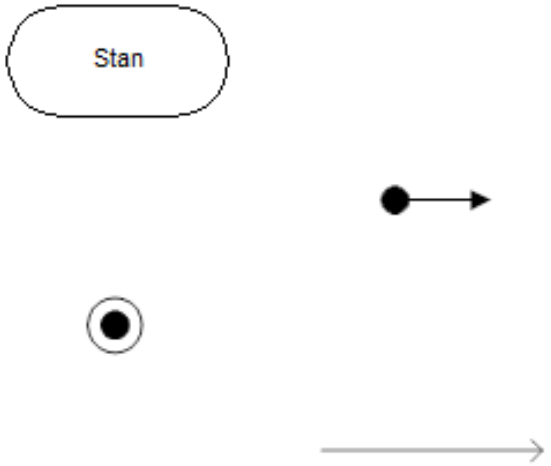
Model dynamiki

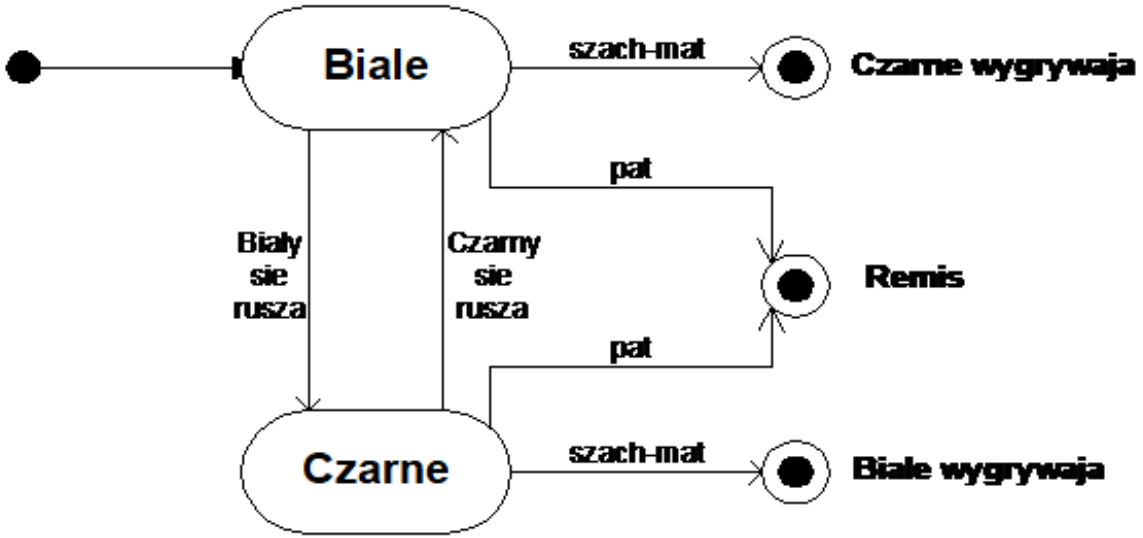
- Obrazuje zmiany stanu systemu w czasie, dotyczy aspektu sterowania systemem.
- Operuje na modelu obiektowym i opisuje aspekt "logiki" obiektu.
- Ustala zbiór możliwych wartości dla atrybutów obiektu.
- Łączy zdarzenia ze stanami obiektów.
- Składa się z diagramów stanów w którym:
  - jeden diagram (maszyna stanowa) odnosi się do jednej klasy o istotnej dynamice,
  - wierzchołki są zbiorem osiąganych stanów (wartości atrybutów i powiązań danej klasy),
  - krawędzie obrazują zmiany stanów w odpowiedzi na zdarzenia (operacje modelu obiektowego) – zjawisko zachodzące w czasie i przestrzeni.

Diagram stanów

Elementy

- stan (aktualne wartości atrybutów i powiązań),
- stan początkowy,
- stan końcowy,
- zdarzenie lub akcja.





Model funkcjonalny

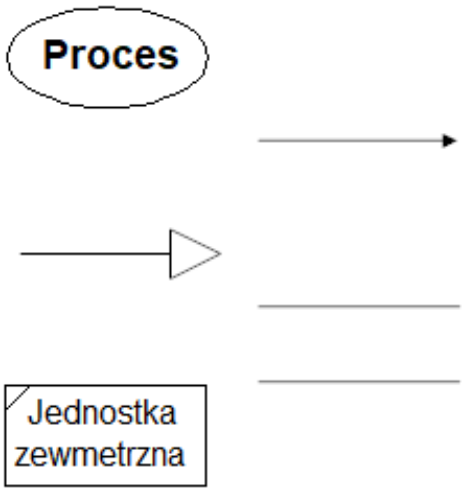
Cele i założenia:

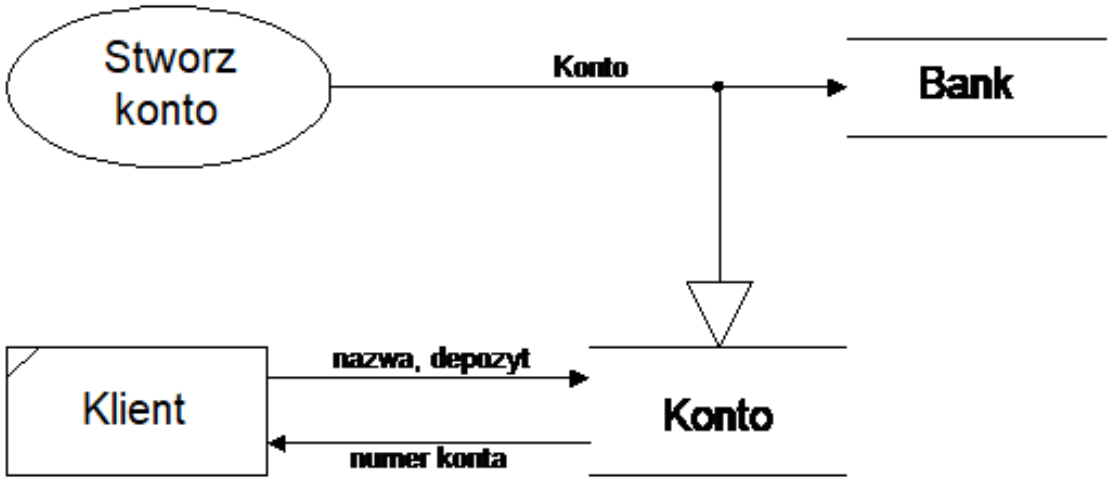
- obrazuje transformację wartości danych **modelu obiektowego**,
- specyfikuje rezultat obliczeń bez podawania jak i kiedy został on osiągnięty,
- uwydatnia znaczenie **operacji** modelu obiektowego i **zdarzeń** modelu dynamiki,
- uwzględnia: dane wejściowe, wyjściowe i tymczasowe wartości obliczane przez system,
- składa się z diagramów przepływu danych w którym:
  - wierzchołki – procesy transformujące dane,
  - krawędzie obrazują przepływ danych.

Diagram przepływu danych (DFD)

Elementy:

- proces (np. funkcja),
- przepływ danych,
- ujście danych,
- magazyn danych,
- jednostka zewnętrzna.





DFD – składniki diagramu

- **Proces** – zazwyczaj jest to funkcja obliczająca pewne wartości i zwracająca je do systemu.
- **Przepływ danych** – łączy wyjście jednego procesu z wejściem następnego, nie zmienia wartości przekazywanych danych, dopuszcza rozgałęzienia i selekcję danych.
- **Jednostka zewnętrzna (aktor)** – obiekt aktywny, który steruje przepływem danych.
- **Magazyn danych** – przechowuje dane w celu późniejszego ich wykorzystania.

Powiązania pomiędzy modelami

Każdy z modeli opisuje jeden z aspektów systemu, ale zawiera odwołania do pozostałych modeli:

- model obiektowy opisuje struktury danych na których operuje model funkcjonalny i dynamiki,
- operacje w modelu obiektowym korespondują ze zdarzeniami z modelu dynamiki i funkcjami z modelu funkcjonalnego,
- model dynamiki obrazuje strukturę logiki obiektu, opisuje zależności pomiędzy wartościami atrybutów obiektu i transformacjami jego stanu,
- model funkcjonalny dostarcza funkcji wywoływanych w operacjach i zdarzeniach oraz ustala ograniczenia w wartościach obiektu.

OMT a inne standardy modelowania

- **Unified Modeling Language** (UML) adoptuje bez większych modyfikacji wszystkie 3 modele proponowane w OMT.
- **Model Driven Architecture** (MDA) jest rozwinięciem stosu czynności tworzenia oprogramowania z OMT.
- Podobnie jak "The Booch Methodology" – OMT koncentruje się głównie na fazie analizy i projektowania.

Większy przykład

Stacja kontroli lotów

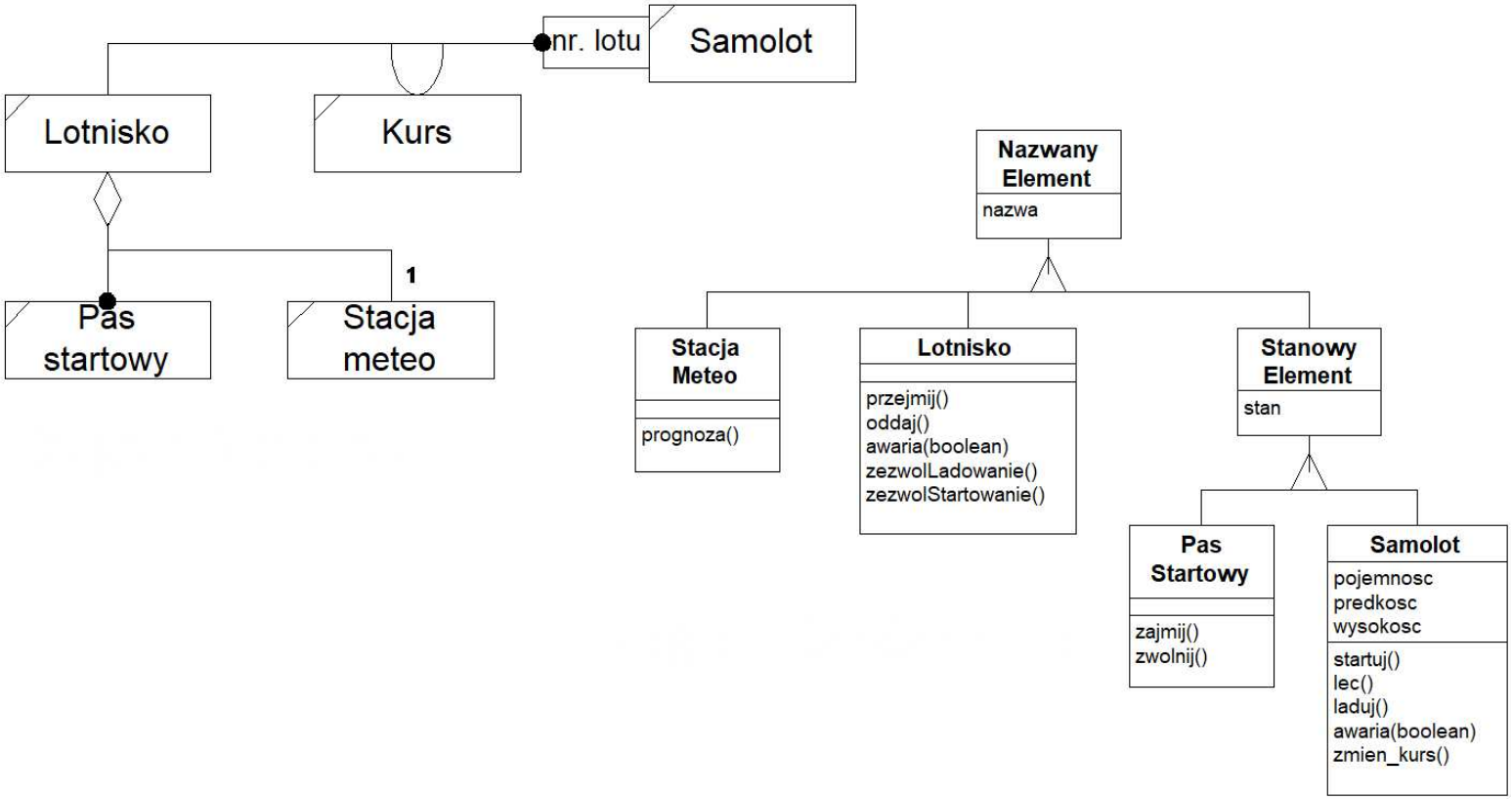
Wymagania ogólne:

- zamodelowanie symulatora lotniska i stacji kontroli lotów,
- lotnisko przyjmuje i startuje samoloty,
- start i lądowanie samolotu zależy od warunków pogodowych i stanu dostępnych pasów,
- stacja kontroli lotów kontroluje także loty tranzytowe.

Obiekty dziedziny i ich odpowiedzialność

- Lotnisko – obiekt fizyczny posiadający strukturę organizacyjną i procedury obsługi.
- Samolot – obiekty wymieniane i kontrolowane przez stacje, w danej placówce identyfikowany jest przez numer lotu.
- Pasy startowe – wpływają na zdolności przyjmowania i startowania samolotów.
- Stacja meteo – obiekt pomocniczy lotniska, modeluje warunki meteorologiczne.

Diagramy klas

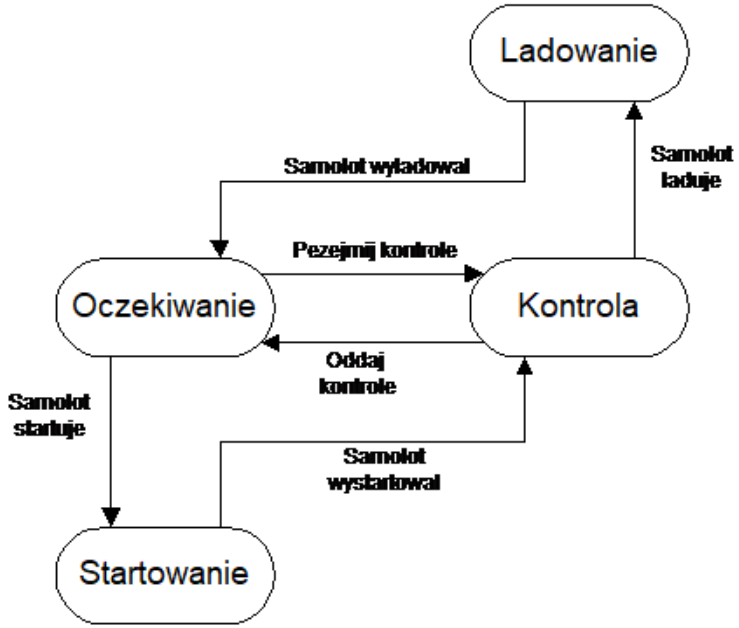


W diagramie dziedziczenia wyodrębniono klasy bazowe ”Nazwany Element” i ”Stanowy Element”.

Diagramy stanów

Wyodrębnione stany **lotniska**:

- **Lądowanie** – kontrola nad samolotem podczas lądowania,
- **Startowanie** – kontrola nad samolotem podczas startowania,
- **Kontrola** – nadzór nad samolotami w strefie,
- **Oczekiwanie** – stan spoczynkowy.



Wyodrębnione stany **samolotu**:

- Startuje
- Leci
- Ląduje

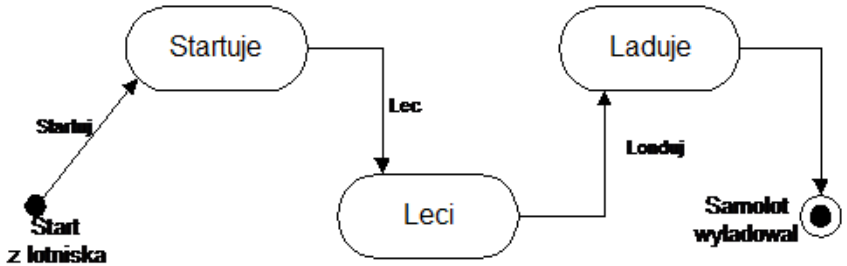
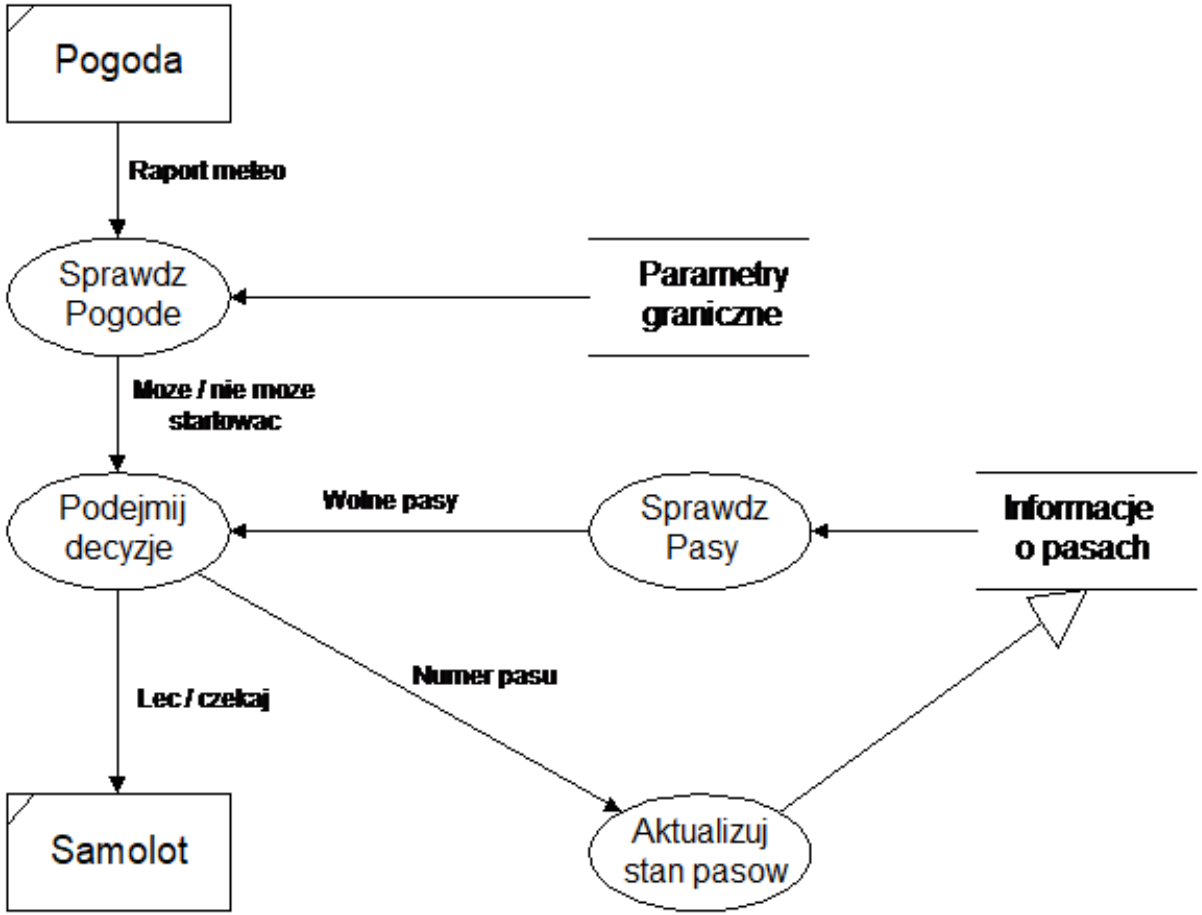


Diagram funkcjonalny

Przedstawiony diagram przykładowy obrazuje proces decyzyjny startowania samolotu:



Wnioski

- OMT opisuje w sposób kompletny proces tworzenia oprogramowania.
- Wydłuża czas życia systemu informatycznego poza czas życia technologii.
- Wprowadza duży stopień abstrakcji w początkowej fazie projektowania, co pozwala na uniezależnienie się od języka programowania i platformy systemowej.
- Większość wiodących obecnie standardów oprogramowania wywodzi się, bądź bezpośrednio adoptuje OMT (np. UML, MDA).
- Kładzie mały nacisk na proces implementacji.

Wnioski

- OMT opisuje w sposób kompletny proces tworzenia oprogramowania.
- Wydłuża czas życia systemu informatycznego poza czas życia technologii.
- Wprowadza duży stopień abstrakcji w początkowej fazie projektowania, co pozwala na uniezależnienie się od języka programowania i platformy systemowej.
- Większość wiodących obecnie standardów oprogramowania wywodzi się, bądź bezpośrednio adoptuje OMT (np. UML, MDA).
- Kładzie mały nacisk na proces implementacji.
- Nie uwzględnia procesu testowania i zarządzania cyklem życia aplikacji.