Modelowanie obiektowe

ZPO 2009/2010

Sprawy organizacyjne

- dr Wojciech Tylman, Katedra Mikroelektroniki i Technik Informatycznych PŁ
- B 18, lp., p. 56
- www.dmcs.p.lodz.pl
- tyl@dmcs.p.lodz.pl
- godziny przyjęć: WWW

Organizacja zajęć

- Wykład: 15h (1h co tydzień)
- Laboratorium/projekt: 15h (1h co tydzień)

- Zaliczenie przedmiotu: zaliczenie wykładu i laboratorium/projektu, oceny brane z tą samą wagą

Modelowanie

- Jest ważne przy tworzeniu wysokiej jakości oprogramowania
- Jest przydatne przy tworzeniu i analizie działania organizacji
- Modelujemy aby:
  - Zrozumieć system
  - Określić pożądaną strukturę i zachowania
  - Określić architekturę i móc ją zmieniać
  - W celu zarządzania ryzykiem

Model

- Model jest uproszczeniem rzeczywistości
- Uproszczenie pozwala pominąć nieistotne (w danym momencie, aspekcie) szczegóły
- Jednocześnie pomaga uwypuklić kwestie istotne

Zasady modelowania

- Model powinien odzwierciedlać rzeczywistość
- Wybór modelu ma wpływ na rozwiązanie problemu – zarówno od strony metody, jak i jakości rozwiązania
- Każdy model może mieć różny poziom szczegółowości
- Zazwyczaj jeden model nie wystarcza. Kilka modeli to najlepsze rozwiązanie jeśli obiekt modelowany nie jest trywialny

Podejście do modelowania

- Strukturalne. Rozwinęło się w oparciu o strukturalne języki oprogramowania i zaowocowało znaczną liczbą odmiennych rozwiązań. Różnice między rozwiązaniami istotnie ograniczyły użyteczność modelowania strukturalnego.
- Przeprowadzono dwie próby unifikacji:
  - Grupa robocza CRIS (Comparative Review of Information Systems Methodologies)
  - EuroMethod

Podejście do modelowania

- Zorientowane obiektowo. Jest wynikiem wzmożonego zainteresowania językami orientowanymi obiektowo. W latach '89-'94 ponad 50 różnych rozwiązań było rozwijanych, jednak, w przeciwieństwie do modelowania strukturalnego, zbiegły się one w jedno.

Podejście do modelowania

- Najistotniejsze podejścia które złożyły się na ostateczne rozwiązanie to:
  - OMT (Object Modeling Technique), Rumbaugh 1991
  - OOAD (Object Oriented Analysis and Design), Booch 1991
  - OOSE (Object Oriented Software Engineering), Jacobson 1992

W stronę UML

- Prace nad UML rozpoczęły się w 1994, kiedy Rumbaugh i Booch, obaj zatrudniani przez Rational Software Corporation, rozpoczęli prace nad unifikacją OMT i OOAD. Rezultat, Unified Method (UM) 0.8, został zaprezentowany w '95. W tym samym roku, Jacobson dołączył do Rational Software Corporation i wzbogacił UM elementami własnego OOSE, co zaowocowało UM 0.9 i UM 0.91 (oba w '96). Od tego momentu język ten jest znany jako **UML**.

Rozwój UML

- Wysiłki Rational Software Corporation zostały szybko wsparte przez istotne firmy, między innymi: IBM, DEC, HP, Oracle, Unisys, Microsoft. Doprowadziło to do dalszego rozwoju - wersji 1.0 w 1997. Wersja ta została później przekazana do Object Management Group (OMG). Wersja 1.1 powstała w tym samym roku. Była to wersja oficjalna do 2001 (wersja 1.4). Wersja 1.5 stała się oficjalna w 2003.

UML 2.0

- Wersja 2.0 została wprowadzona w 2003. Jest to pierwsza istotna zmiana standardu, wprowadzająca liczne nowe diagramy i kategorie modelowania.

Diagramy UML

- Model w UML jest graficzną reprezentacją systemu
- Reprezentacja składa się z logicznie połączonych diagramów
- Wersja 2.0 zawiera 13 typów diagramów
- Istotnym pojęciami są pojęcia **klasyfikatora** (classifier) – abstrakcyjnej kategorii która uogólnia kolekcję wystąpień mających te same cechy, oraz **wystąpienia** (instance) – egzemplifikacji klasyfikatora

Widoki UML

- Projektowanie systemu wymaga współpracy znacznej liczby osób, mających różne kompetencje i zakresy odpowiedzialności (kierownicy, projektanci, programiści, klienci itd.)
- Każda osoba widzi system z innego punktu widzenia
- UML rozwiązuje ten problem przez wykorzystanie 5 różnych widoków systemu

Widoki UML

- Widok przypadków użycia (use case view) – określa zakres i oczekiwaną funkcjonalność systemu na wysokim poziomie abstrakcji
- Widok dynamiczny (dynamic view) – opisuje zachowania (dynamikę) wystąpień w systemie
- Widok logiczny (logical view) – opisuje statykę systemu
- Widok implementacyjny (implementation view) – używany przez programistów, opisuje komponenty systemu
- Widok wdrożenia (deployment view) – opisuje sprzęt wymagany przez komponenty

Mechanizmy rozszerzeń

- Mimo że UML zawiera szeroki wachlarz rozwiązań i elementów, może nie odpowiadać w pełni pewnym domenom modelowania
- Z tego powodu zawiera on mechanizmy rozszerzeń
- Istnieją 3 typy mechanizmów rozszerzeń:
  - Stereotyp (stereotype)
  - Ograniczenie (constraint)
  - Metka (tagged value)

Stereotypy

- Umożliwiają utworzenie nowej kategorii modelowania w oparciu o kategorie istniejącą
- Stereotypy mogą być:
  - tekstowe – nazwa zawarta w cudzysłowach: << >> i umieszczona na stereotypowanym elemencie
  - graficzne – odpowiedni symbol graficzny jest umieszczony na stereotypowanym elemencie
- OMG rekomenduje dużą grupę stereotypów standardowych

Ograniczenia

- Sa wyrażeniami określającymi warunki mające zastosowanie do ograniczanego elementu
- Mogą być wyrażone w języku naturalnym, jako formuła matematyczna lub w OCL (Object Constraint Language) – specjalnym języku do ograniczeń obiektowych
- Są umieszczane w nawiasach: { }, przy ograniczanym elemencie

Metki

- Umożliwiają tworzenie nowych własności
- Mają postać nazwa-wartość
- Są umieszczane w nawiasach: { }

Diagramy przypadków użycia

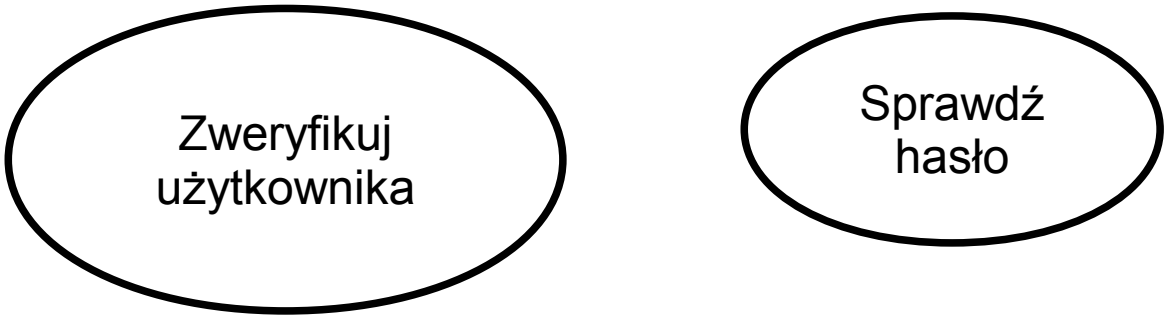
- Umożliwiają:
  - Identyfikację i dokumentację wymogów
  - Analizę zakresu aplikacji
  - Komunikację pomiędzy twórcami, właścicielami, klientami itd.
  - Opracowanie projektu przyszłego systemu, organizacji
  - Określenie procedur testowych dla systemu
- Są dwa typy diagramów przypadków użycia:
  - biznesowe
  - systemowe

Diagramy przypadków użycia

- Zawierają:
  - Przypadki użycia
  - Aktorów
  - Związki (relacje)

Przypadki użycia

- Specyfikacja sekwencji akcji (i ich wariantów) które system może wykonać poprzez interakcję z aktorami tego systemu
- Przypadek użycia jest spójnym fragmentem funkcjonalności systemu
- Nazwą jest rozkaz wypełnienia określonej funkcji. Nazwa jest umieszczona w owalu

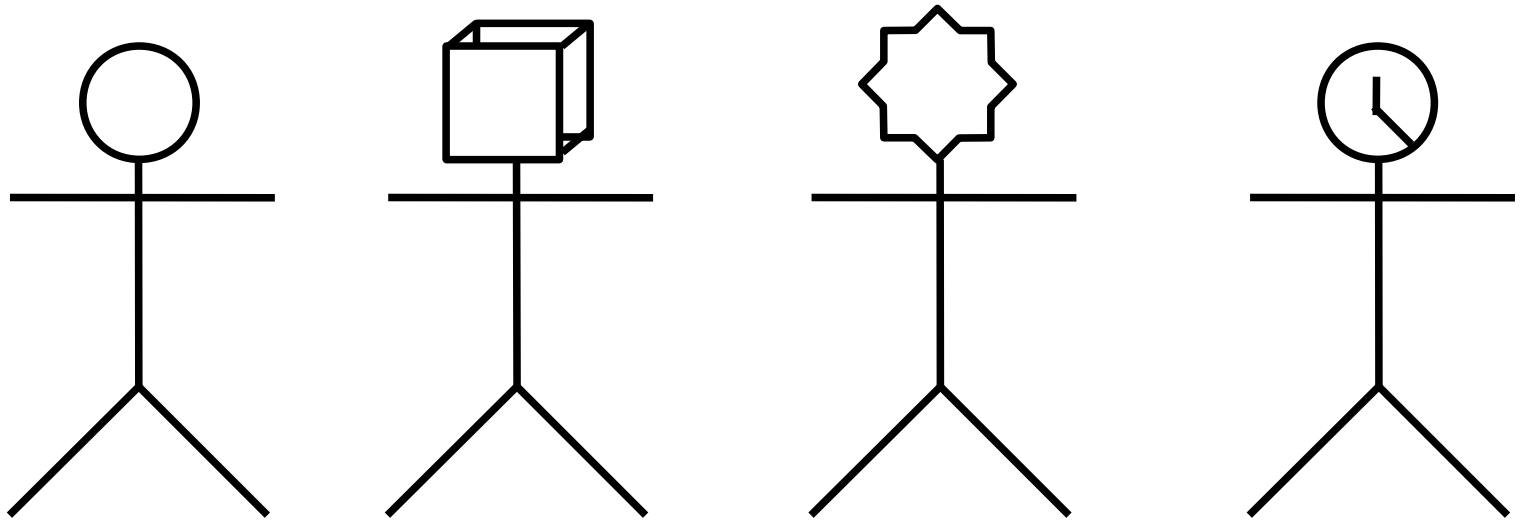


Aktor

- Aktor jest spójnym zbiorem ról odgrywanych przez użytkowników podczas interakcji z przypadkami użycia
- Aktorami mogą być:
  - Osoby (pojedyncza osoba, grupa, organizacja itp.)
  - Zewnętrzne systemy (programowe bądź sprzętowe)
  - Czas
- Nazwa to rzeczownik odzwierciedlający rolę odgrywaną w systemie
- Aktor może używać wielu przypadków użycia, przypadek użycia może być używany przez wielu aktorów

Stereotypy aktorów

- Klasyczny symbol aktora może być stereotypowany aby rozróżnić między różnymi typami aktorów



klasyczny / człowiek

system zewnętrzny

urządzenie

czas

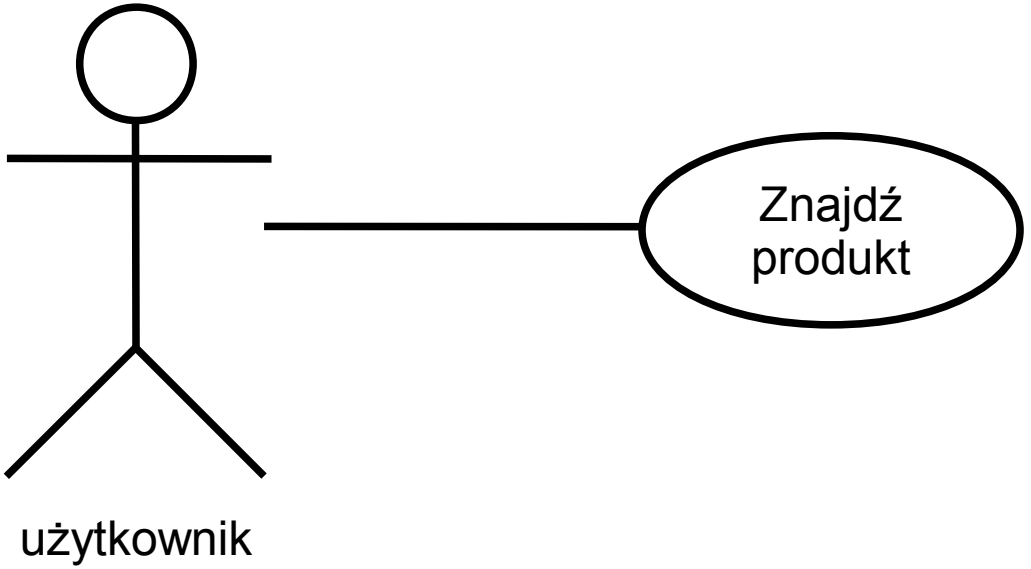
Związek (relationship)

- Wiąże ze sobą elementy diagramu (np. aktorów i przypadki użycia)
- Są 4 rodzaje związków:
  - Asocjacja (association)
  - Uogólnienie (generalisation)
  - Zależność (dependence)
  - Realizacja (realisation)

Asocjacja

- Asocjacja opisuje związek pomiędzy (dwiema lub więcej) wystąpieniami klasyfikatorów
- W diagramie przypadków użycia reprezentuje dwukierunkową komunikację pomiędzy aktorem i przypadkiem użycia
- Jej reprezentacją graficzną jest ciągła linia
- Zazwyczaj nie są nazywane

Asocjacja



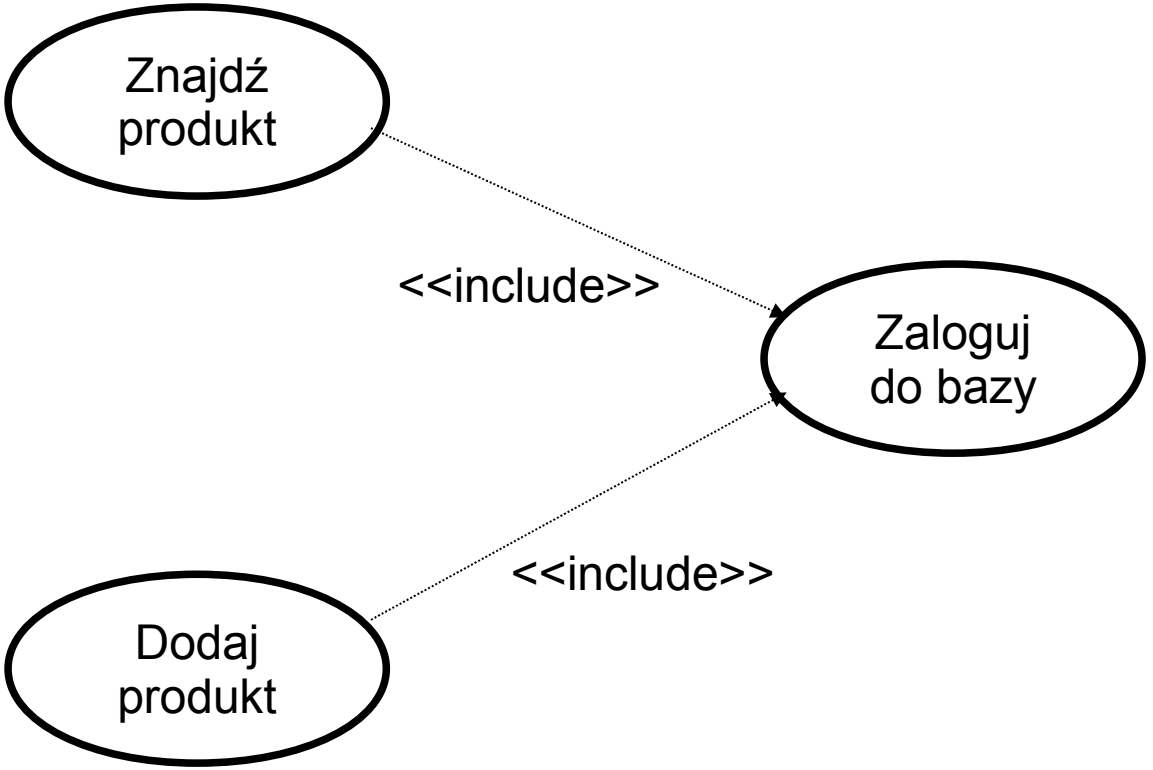
Zależność (dependency)

- Zależność jest związkiem pomiędzy dwoma elementami modelu gdzie zmiana w jednym elemencie (niezależnym one) ma wpływ na drugi element (zależny)
- Jest obrazowana jako linia przerywana zakończona otwartą strzałką
- W diagramach przypadków użycia zależność jest stereotypowana w:
  - Zależność <<include>>
  - Zależność <<extend>>

Zależność <<include>>

- Związek między przypadkiem zawierającym i zawieranym
- Przypadek zawierany jest wykonywany zawsze gdy wykonywany jest przypadek zawierający – i tylko wtedy
- Jest przydatna gdy kilka przypadków użycia zawiera tę sama wspólną część
- Strzałka skierowana jest od przypadku zawierającego do zawieranego

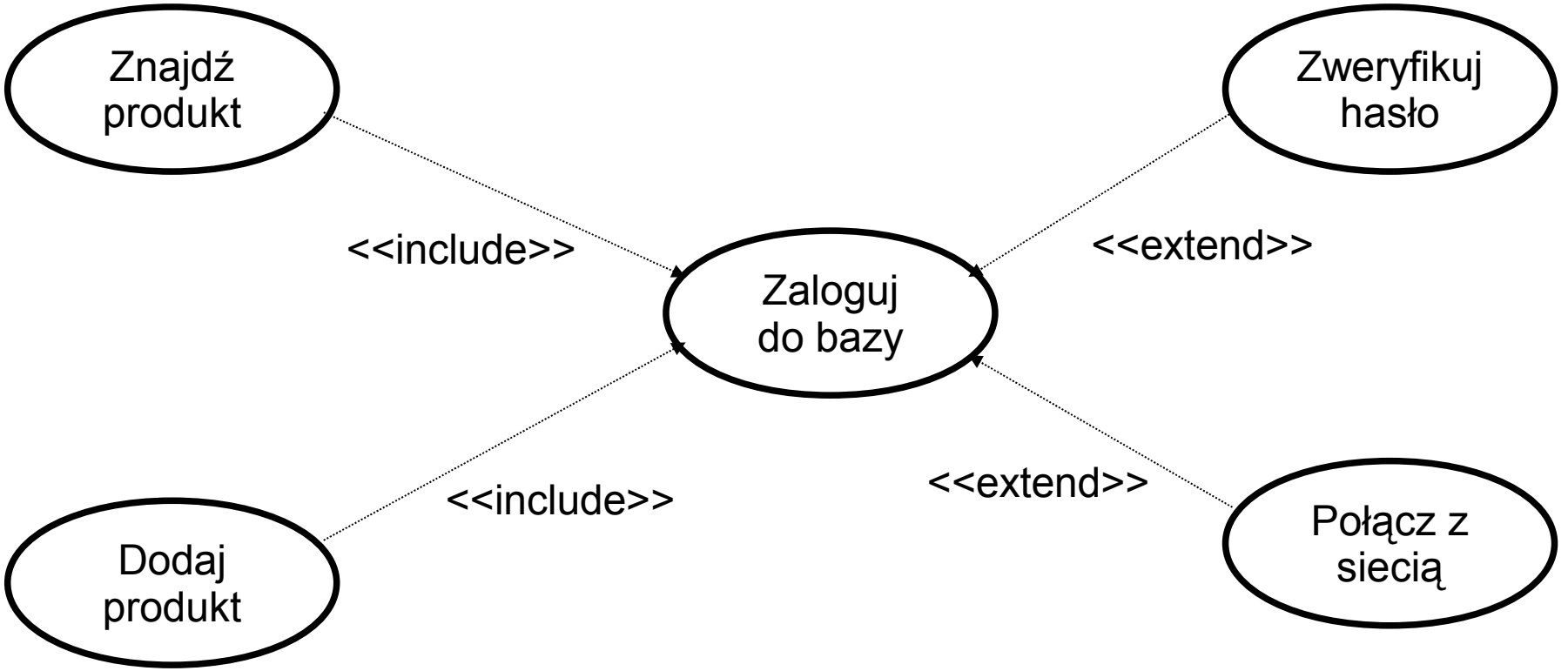
Zależność <<include>>



Zależność <<extend>>

- Zależność między przypadkiem podstawowym i przypadkiem który opcjonalnie może wprowadzić dodatkową funkcjonalność do przypadku podstawowego
- Jest przydatna gdy przypadek może, w pewnych warunkach, być uzależniony od innych przypadków
- Strzałka wskazuje od rozszerzenia do przypadku podstawowego

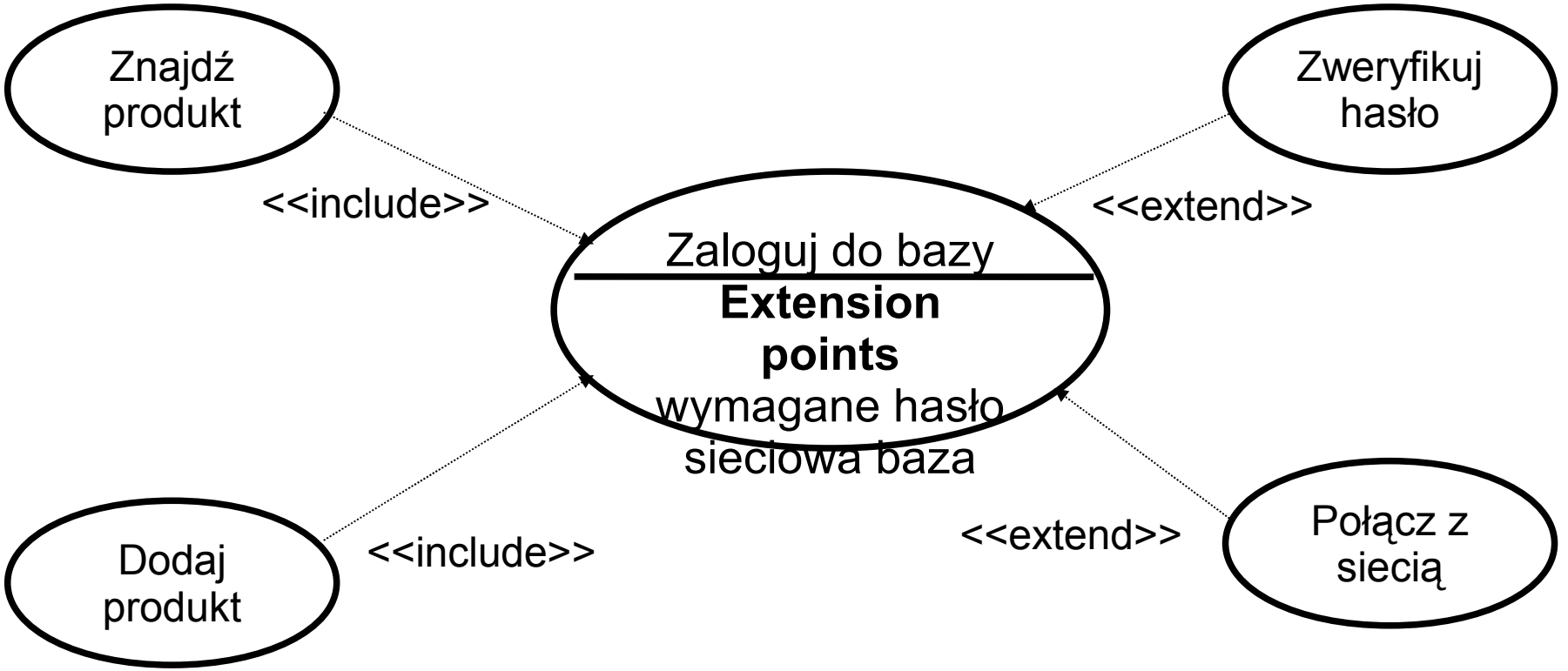
Zależność <<extend>>



Punkty rozszerzenia

- Jest możliwe określenie sytuacji (warunków) w których musi nastąpić dołączenie przypadków rozszerzających
- Są one wyszczególnione w rozszerzanym przypadku, pod poziomą linią

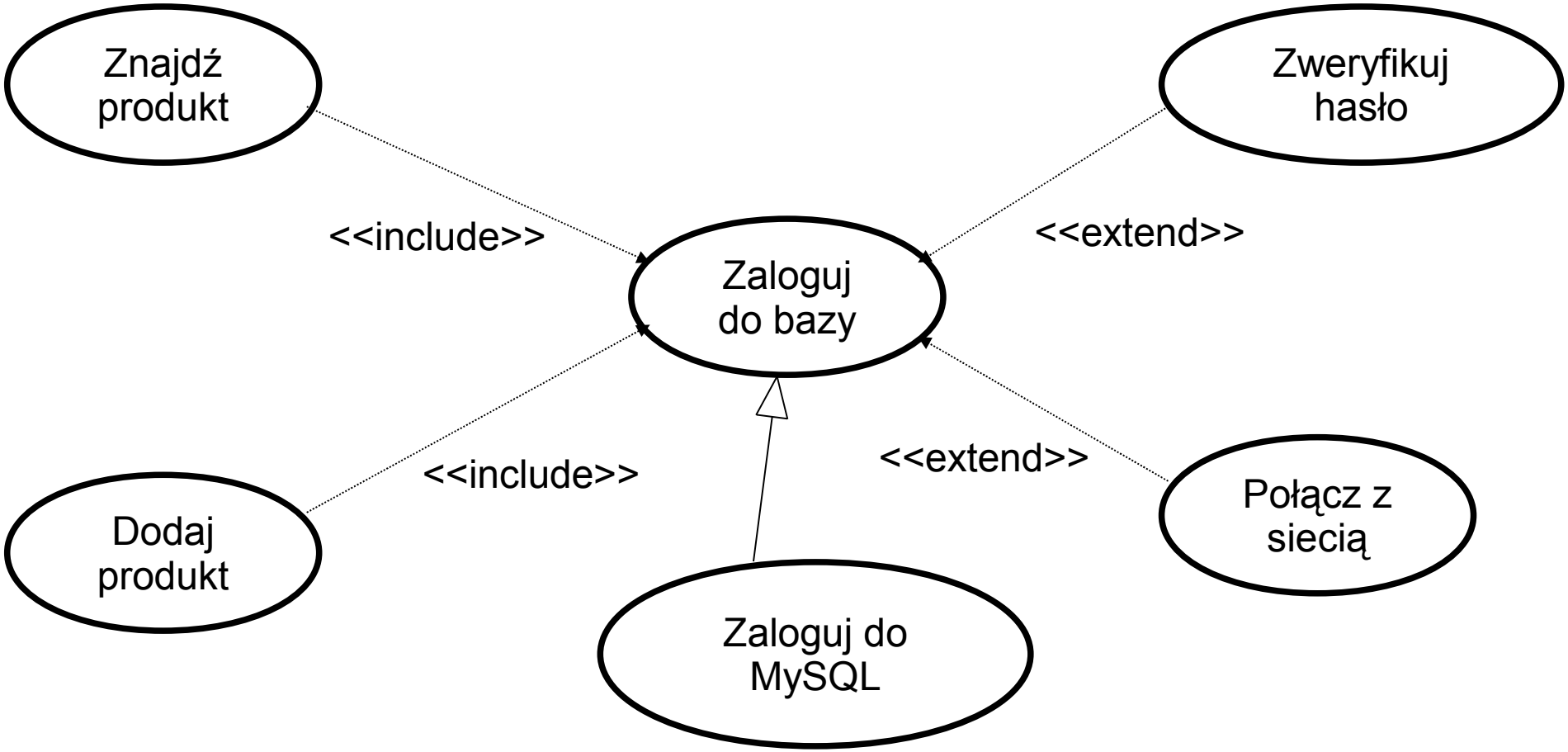
Punkty rozszerzenia



Uogólnienie

- Uogólnienie jest taksonomiczną relacją pomiędzy ogólnym i specjalizowanym klasyfikatorem
- Specjalizowany klasyfikator dziedziczy wszystkie cechy klasyfikatora ogólnego
- Jest obrazowana linią zakończoną zamkniętą strzałką, wskazującą w stronę klasyfikatora ogólnego

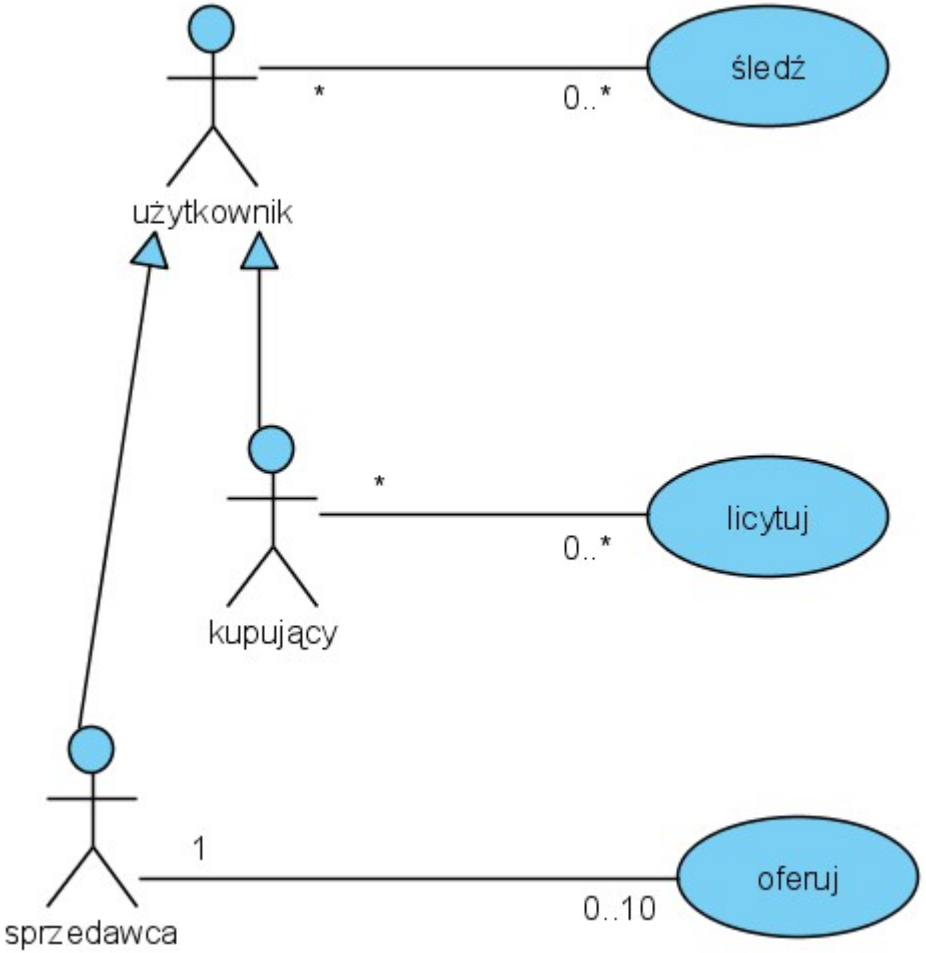
Uogólnienie



Liczebność

- Umożliwia określenie liczby elementów biorących udział w tej asocjacji, na każdym jej końcu
- Możliwe przypadki:
  - n (n > 0) dokładnie n
  - n..* (n ≥ 0) n lub więcej
  - n..m (m > n ≥ 0) od n do m
  - * wiele (nieznana liczba)
  - n, m, o..p, q (q > p...) lista wartości

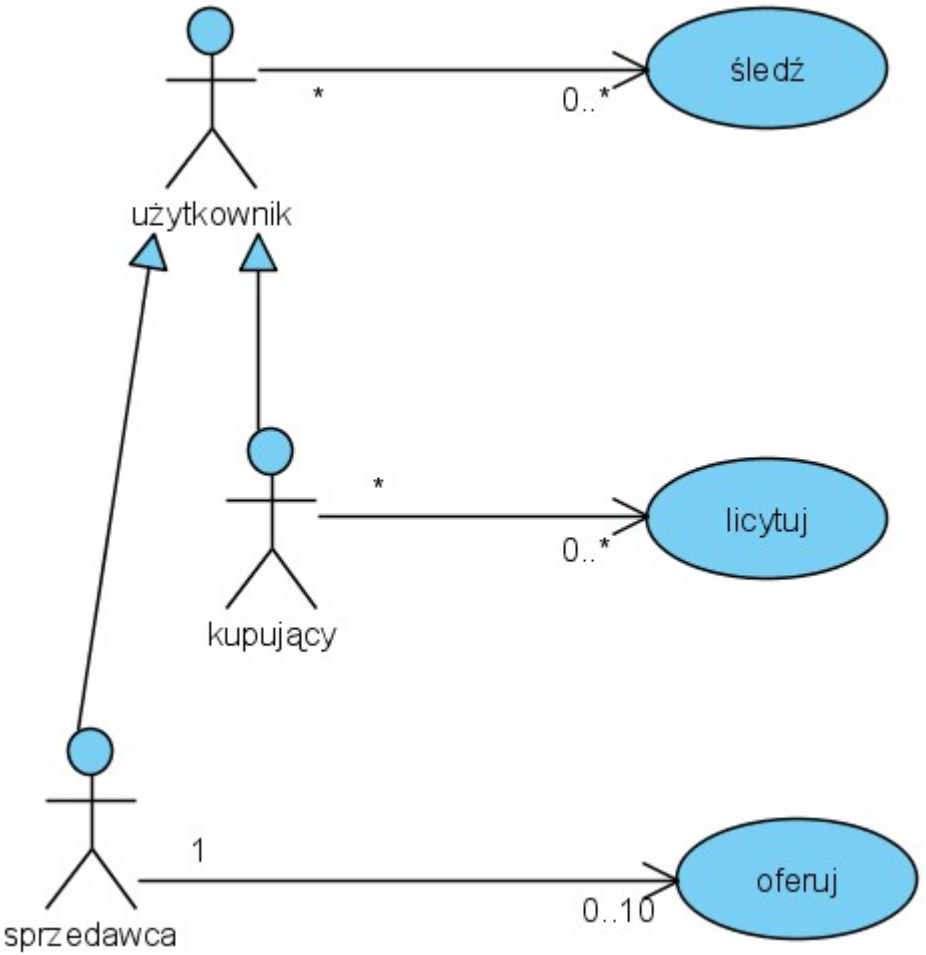
Liczebność


śledź
*
0..*
użytkownik
licytuj
*
0..*
kupujący
1
oferuj
0..10
sprzedawca

Skierowanie (navigability)

- Dwukierunkowa asocjacja może być wzbogacona o informację określającą stronę inicjującą komunikację
- Jest to zobrazowane przy pomocy strzałki
- Strzałka nie wskazuje kierunku przepływu danych

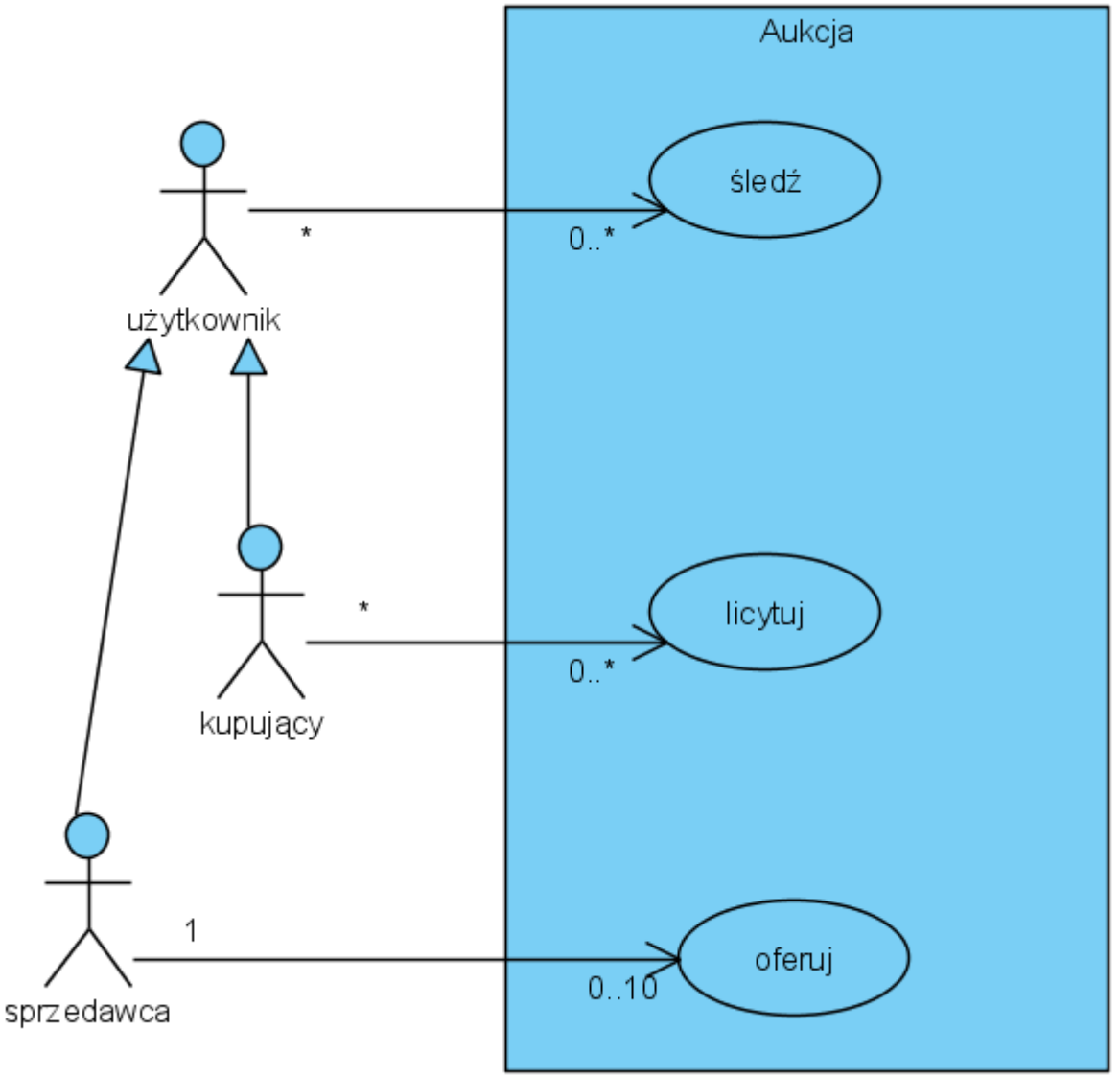
Skierowanie



System

- Przypadki użycia mogą zostać zgrupowane, tworząc kompletny system
- Zgrupowanie jest wyrażone poprzez umieszczenie przypadków wewnątrz nazwanego prostokąta

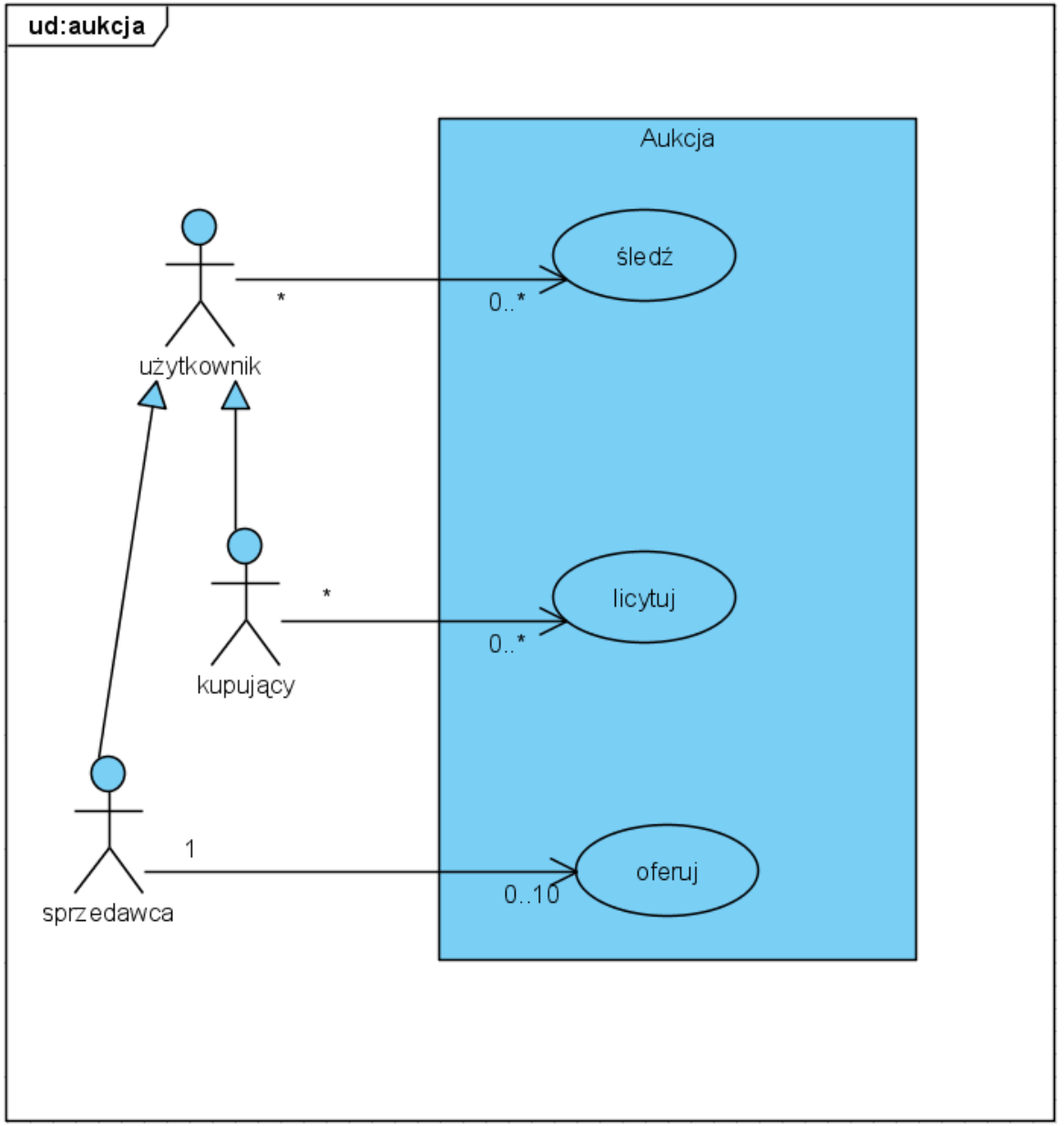
System


Aukcja
śledź
licytuj
oferuj
użytkownik
kupujący
sprzedawca
*
0..*
*
0..*
1
0..10

Rama

- Cały diagram (dowolnego typu) może być umieszczony w ramie
- Rama ma nagłówek (lewy górny róg) określający typ i nazwę diagramu
- Używanie ram zwiększa przejrzystość dokumentacji w przypadku dużych projektów

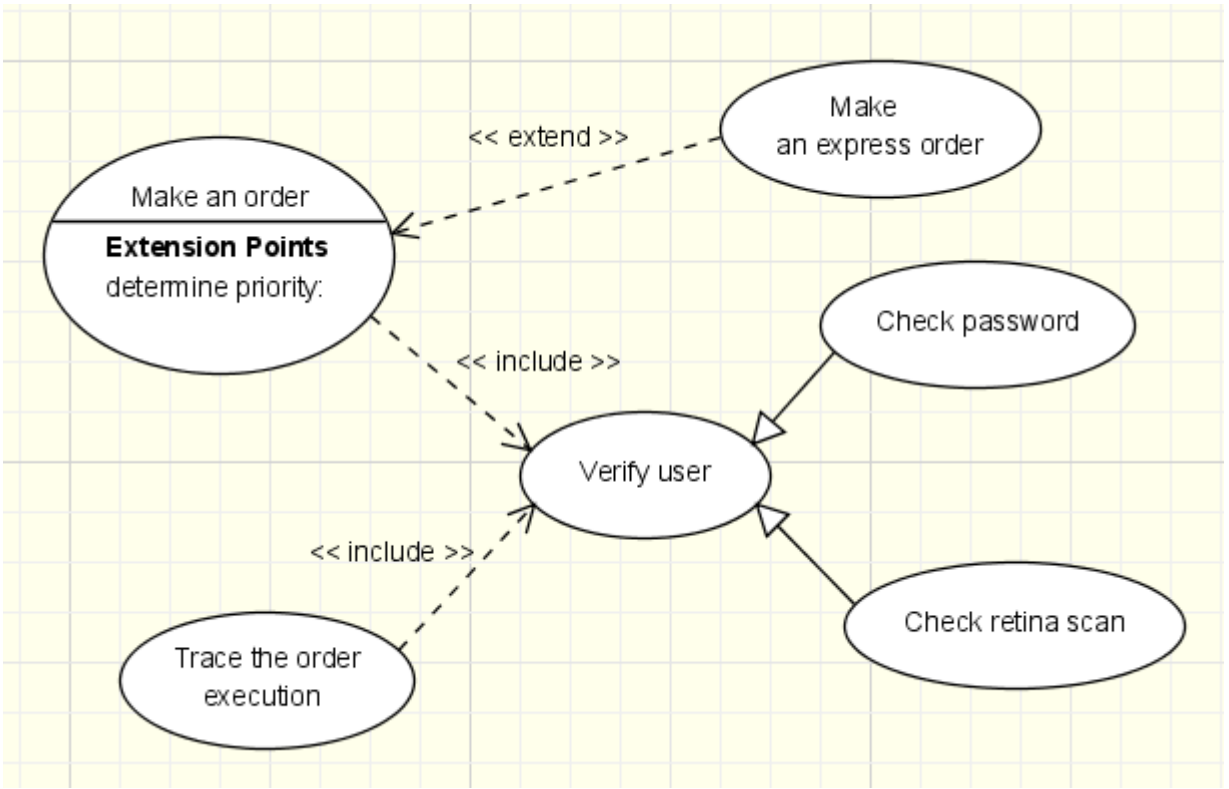
Rama


ud:aukcja
Aukcja
śledź
0..*
*
użytkownik
licytuj
0..*
*
kupujący
oferuj
0..10
1
sprzedawca

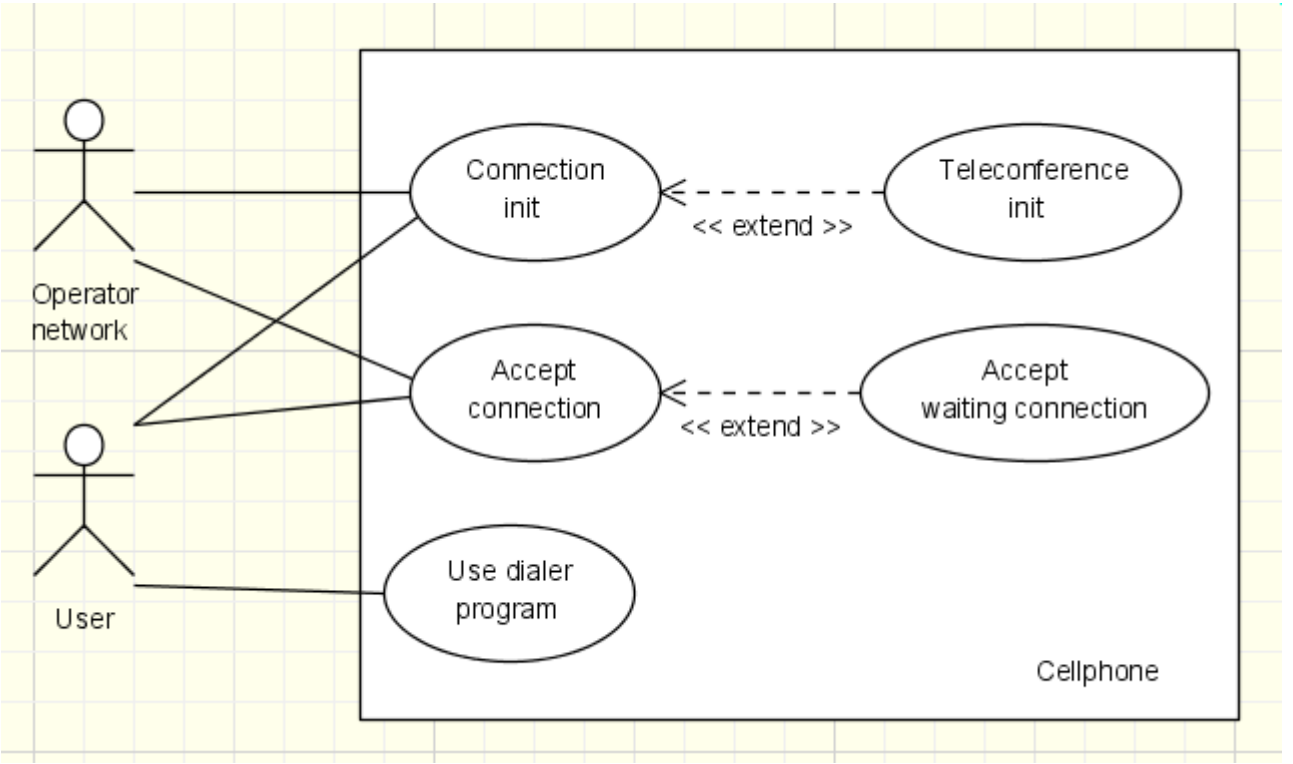
Dokumentowanie przypadków użycia

- Diagram przypadków użycia sam w sobie jest bardzo ogólnikowy
- Aby precyzyjnie określić pożądane zachowanie systemu, każdy przypadek użycia powinien posiadać dodatkowa informację, tzw. scenariusz
- Scenariusz jest sekwencją akcji, określająca zachowanie
- Dla złożonych przypadków można zdefiniować główny oraz alternatywne scenariusze
- Scenariusz może zostać zapisany w języku naturalnym, pseudo-kodzie, tabeli itp.

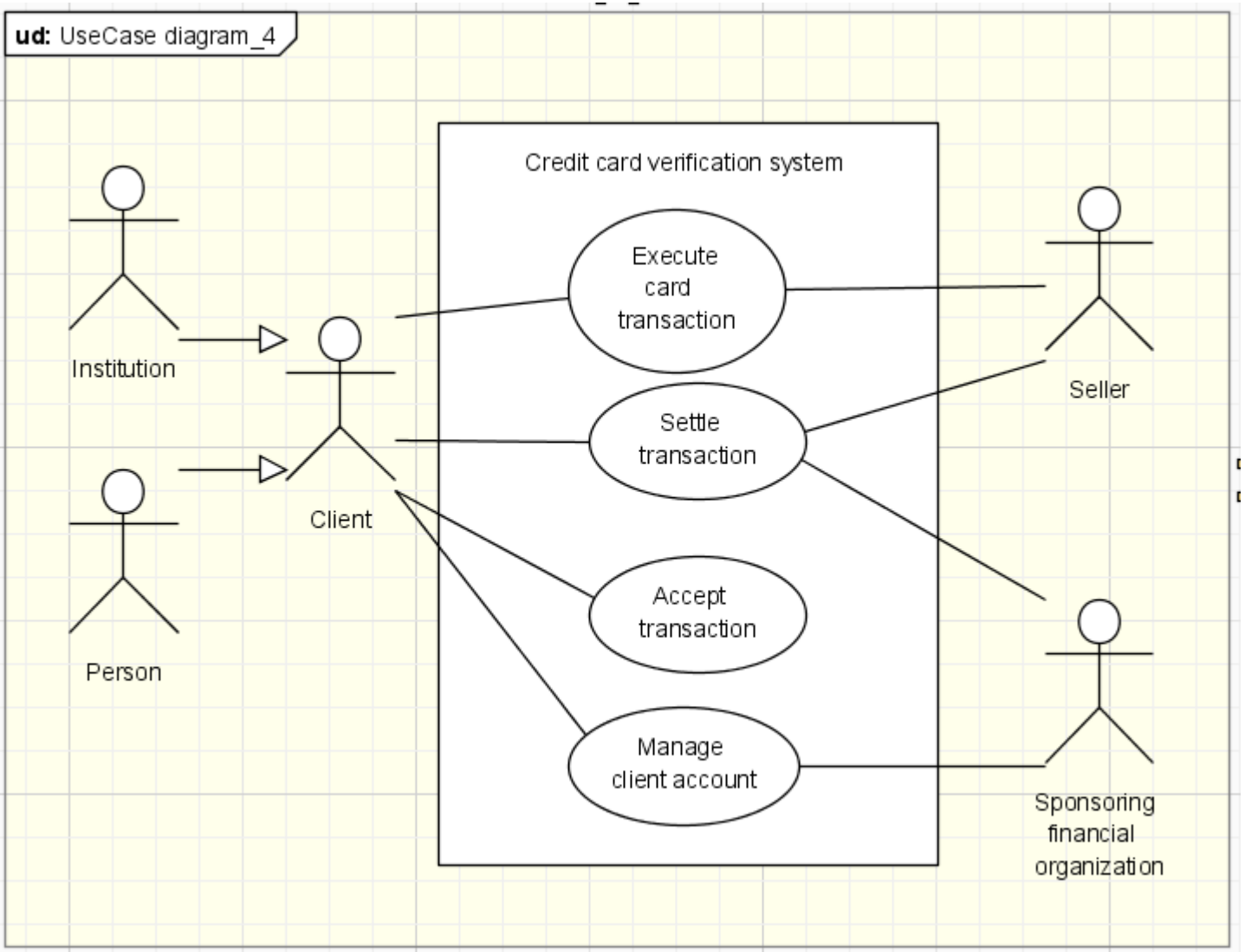
Przykładowe diagramy



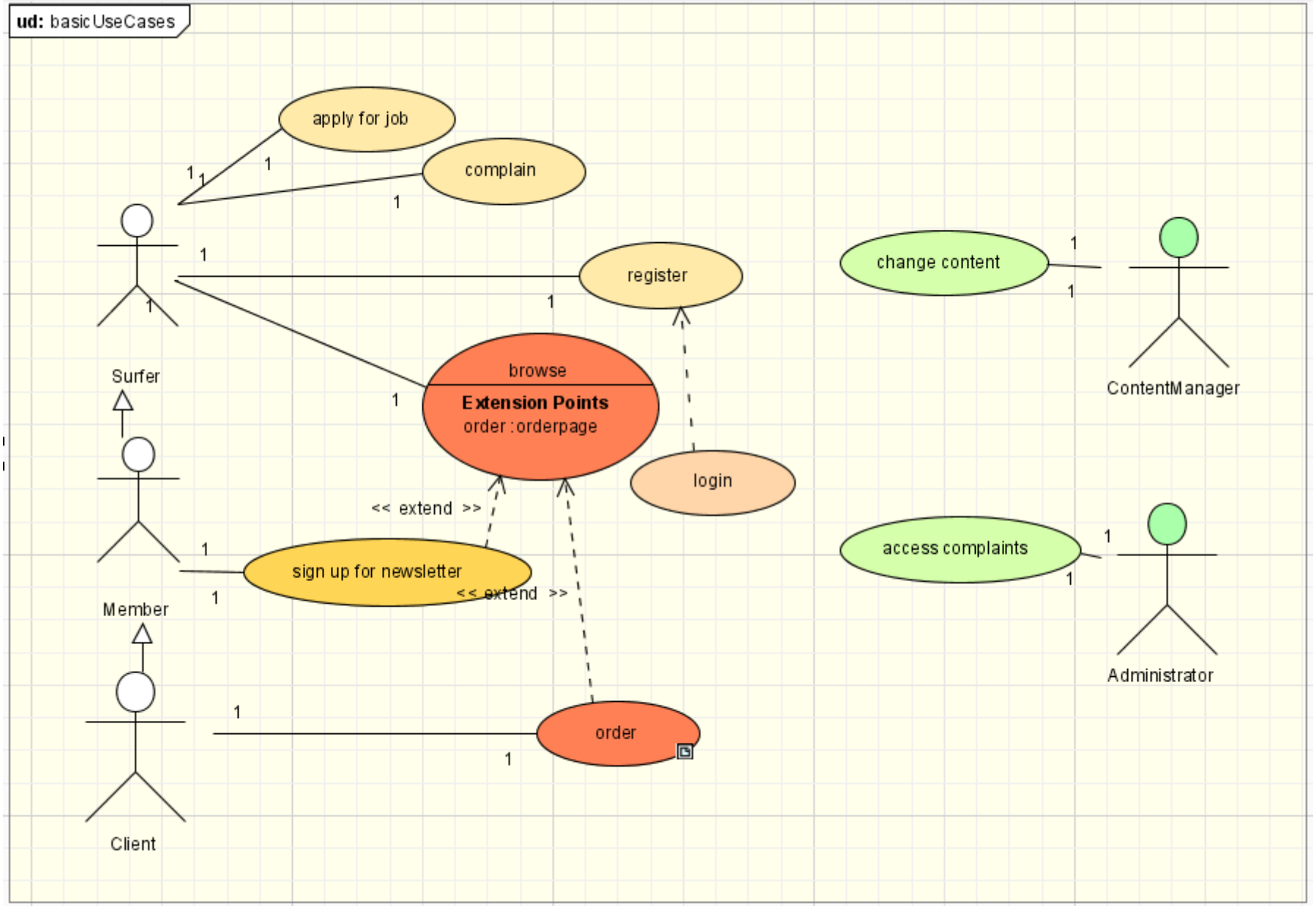
Przykładowe diagramy



Przykładowe diagramy



Przykładowe diagramy

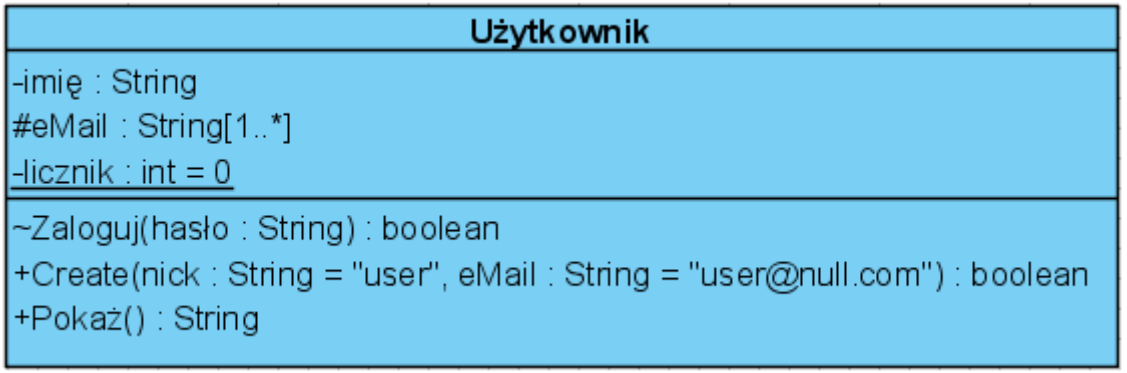


Diagramy klas

- Zawiera informacje o statycznych związkach między elementami (klasami)
- Są ściśle powiązane z technikami programowania zorientowanego obiektowo
- Są jednymi z istotniejszych diagramów w UML

Symbol klasy

- Symbolem klasy jest prostokąt, zwykle podzielony poziomymi liniami na trzy sekcje:
  - nazwy
  - atrybutów
  - operacji
- W razie potrzeby może zostać uzupełniony dodatkowymi sekcjami (np. wyjątków)

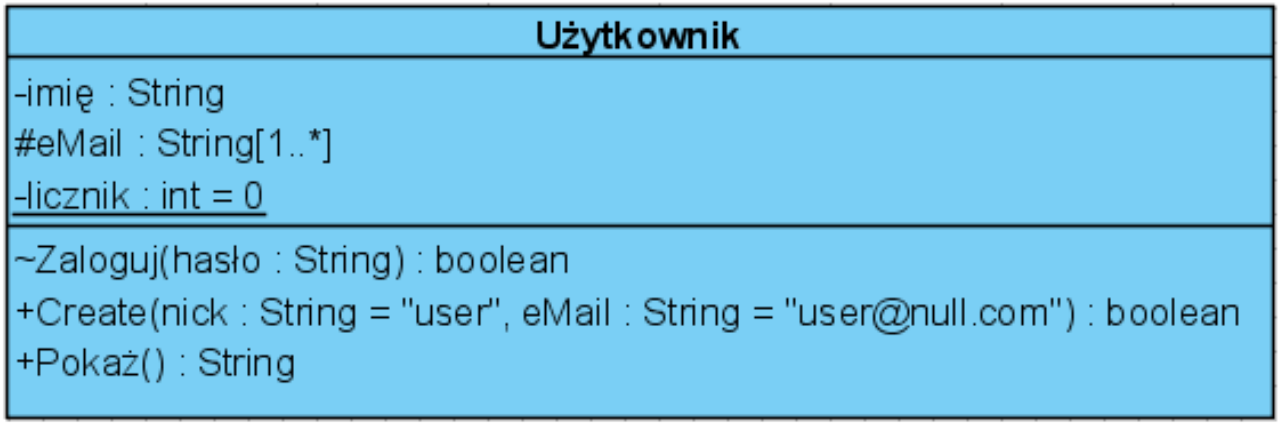


Symbol klasy

- Przy złożonych klasach wyświetlenie wszystkich atrybutów i operacji może zabrać zbyt dużo miejsca
- Możliwe rozwiązania to:
  - Wyświetlenie tylko nazwy klasy, bez sekcji atrybutów i operacji
  - Wyświetlenie tylko nazwy klasy, z pustymi sekcjami atrybutów i operacji
  - Wyświetlenie tylko części atrybutów lub operacji, zaznaczając kontynuację poprzez wielokropek
  - Ukrycie niektórych atrybutów lub operacji

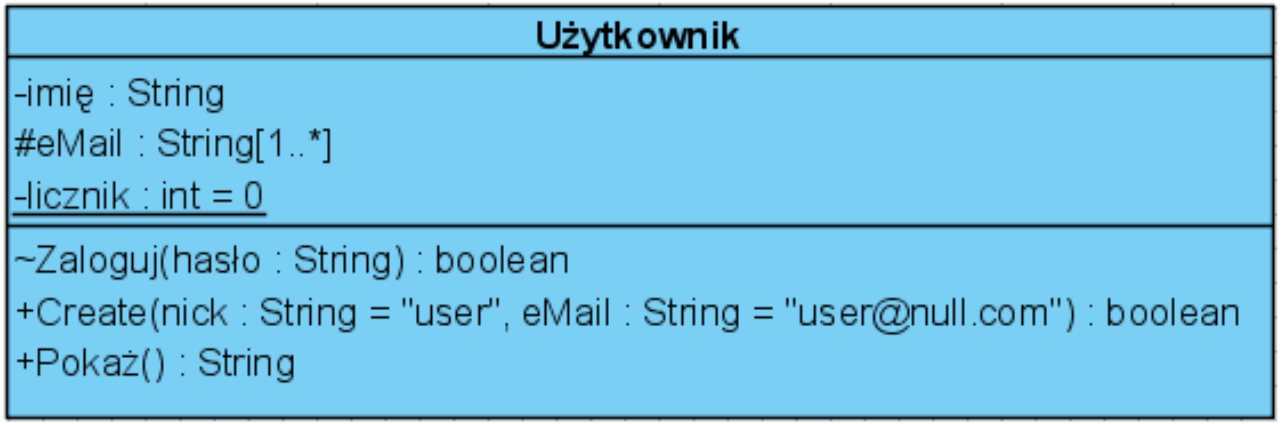
Kontrola dostępu

- Można określić modyfikatory dostępu dla składowych
- Są one ściśle powiązane z koncepcjami programowania zorientowanego obiektowo
- Możliwe rodzaje dostępu:
  - \+ publiczny
  - \- prywatny
  - # chroniony
  - ~ pakietu



Składniki statyczne

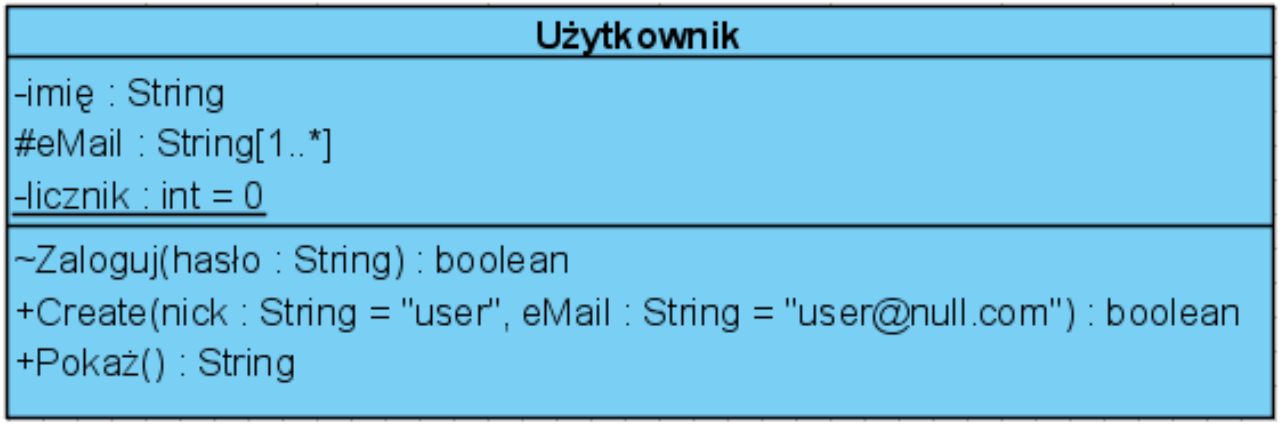
- Składniki można zadeklarować jako statyczne – działające na rzecz klasy, nie obiektu
- Koncepcja identyczna jak w językach zorientowanych obiektowo
- Reprezentacją graficzną jest podkreślenie



Specyfikacja składników

- Atrybuty mogą mieć określone:
  - Typ. Typ jest umieszczany po nazwie, oddzielony dwukropkiem
  - Liczebność
  - Wartość początkową
- Operacje mogą mieć określone:
  - Typ zwracany. Typ jest umieszczany po nazwie, oddzielony dwukropkiem
  - Argumenty. Każdy argument może być określony tak jak atrybut, z dodatkowym oznaczeniem kierunku przekazywania wartości (domyślnie “in”)

Specyfikacja składników



Związki

- Wszystkie 4 typy związków są używane
- Głównym typem jest asocjacja
- Może mieć następujące cechy (pogrubiono nowe w stosunku do diagramów przypadków użycia):
  - **nazwa**
  - **role**
  - kierunek nawigacji
  - liczebność
  - **agregacja**

Nazwa

- Można nazwać asocjację aby doprecyzować jej znaczenie
- Nazwa może zawierać kierunek

Role

- Inny sposób doprecyzowania asocjacji
- Rola klasy jest określona przez tekst umieszczony w pobliżu tej klasy
- Można określić jednocześnie nazwę i role

Kierunek nawigacji

- Domyślnie asocjacja jest dwukierunkowa
- Aby była jednokierunkowa, dodaje się strzałkę
- Oznacza to że **komunikacja jest** jednokierunkowa (inaczej niż diagramy przypadków użycia)

Liczebność

- Znaczenie identyczne jak w diagramach przypadków użycia

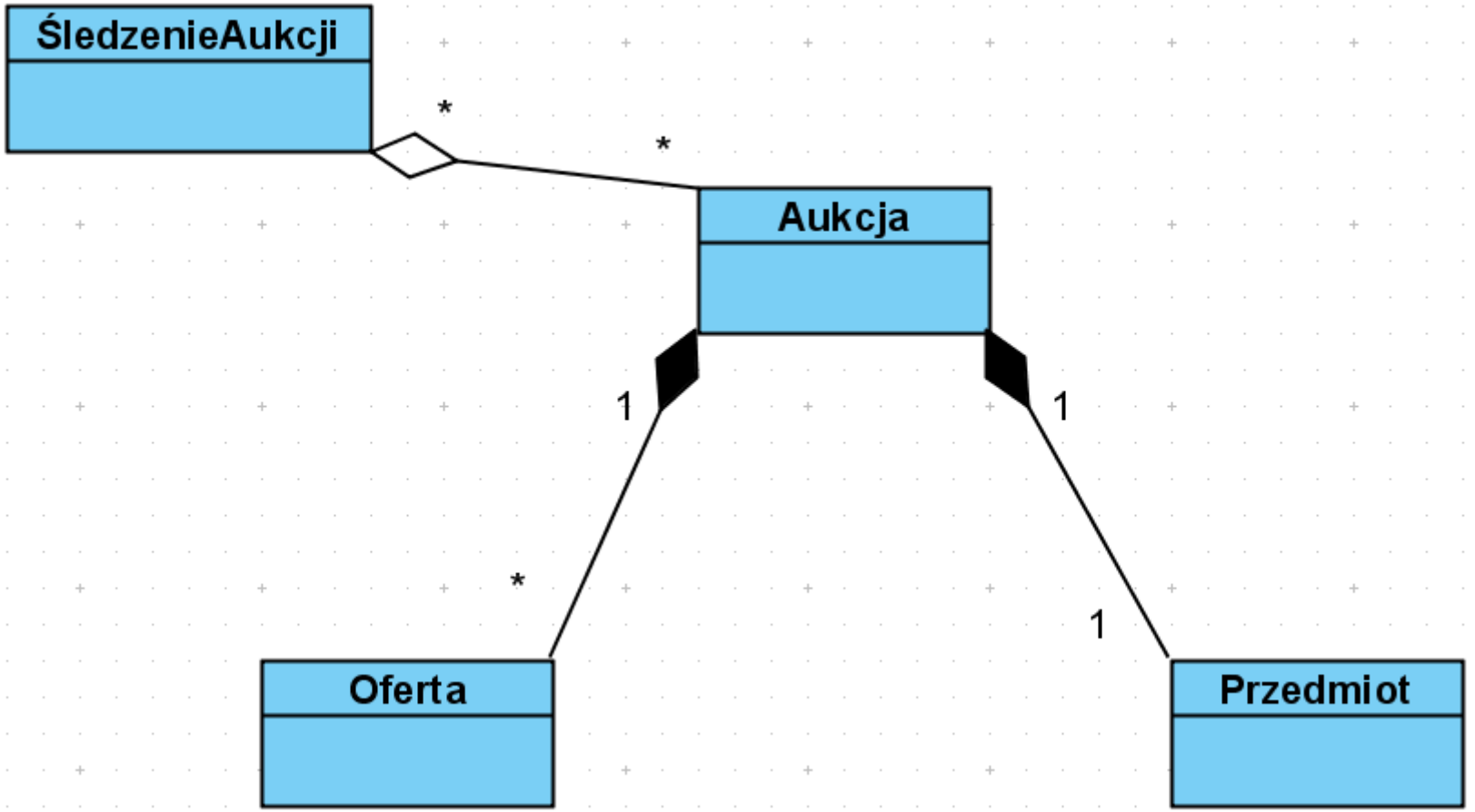
Agregacja

- Określa związek między całością i częścią
- Są dwa typy:
  - całkowita (kompozycja, silna agregacja)
  - częściowa (agregacja, słaba agregacja)
- Jest obrazowana przez romb umieszczony przy symbolu określającym całość
- Silna agregacja jest zobrazowana przez pełen romb, słaba – przez pusty

Silna i słaba agregacja

- W przypadku silnej agregacji części składowe nie mogą istnieć jeśli symbol określający całość jest usunięty.
  - analogia do zawierania obiektu przez inny obiekt.
- W przypadku słabej agregacji jest to możliwe. Jeden obiekt może być też zawierany przez wiele innych.
  - analogia do zawierania wskaźnika (bądź referencji) do innego obiektu.

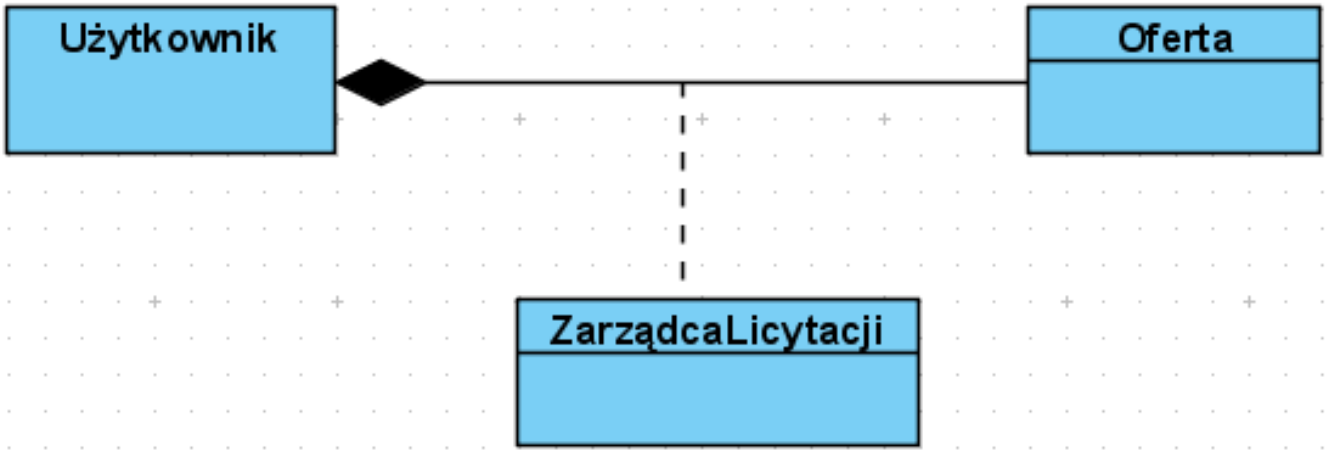
Silna i słaba agregacja



Klasa asocjacji

- Kolejny sposób dokładnego określenia asocjacji
- Jest zobrazowana przez klasę umieszczoną w pobliżu asocjacji i połączoną z nią przerywaną linią

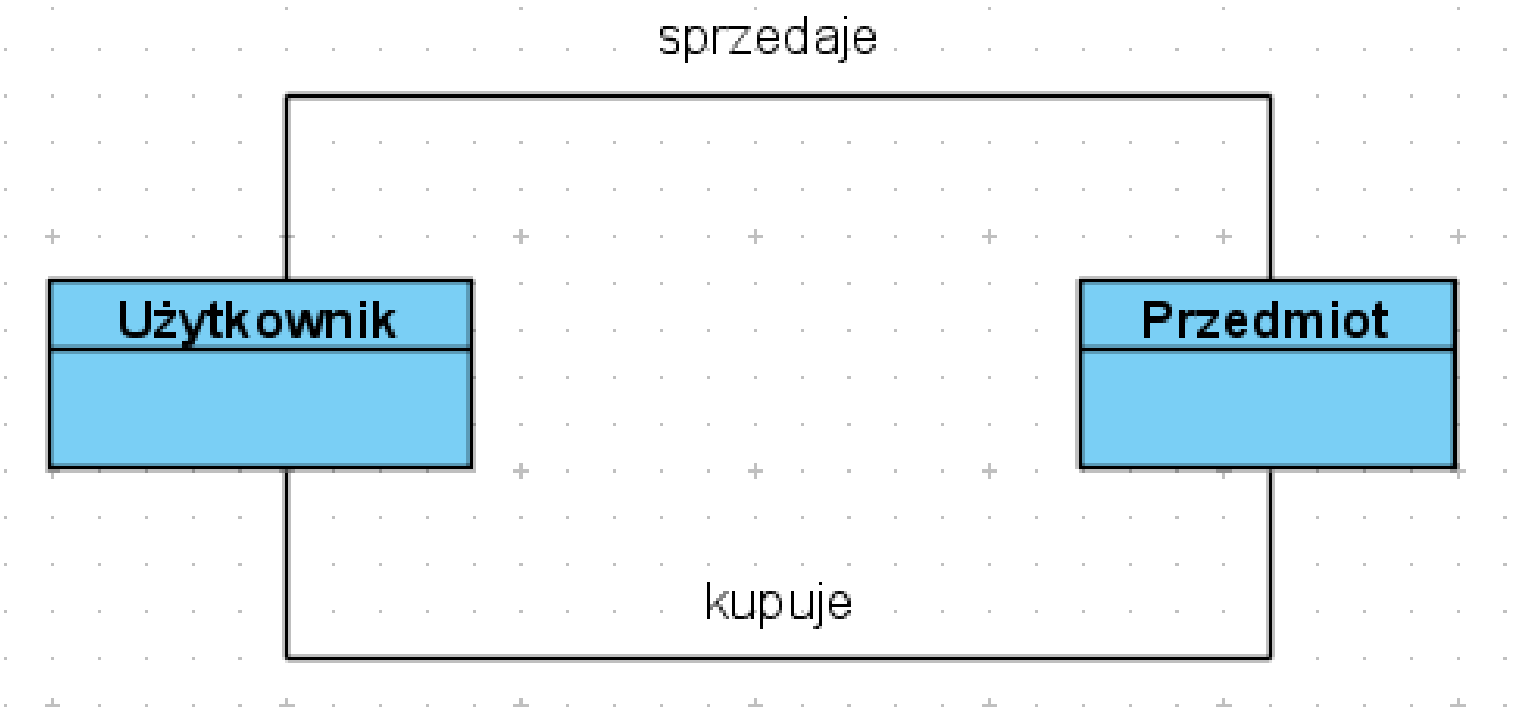
Klasa asocjacji



Asocjacja wielokrotna

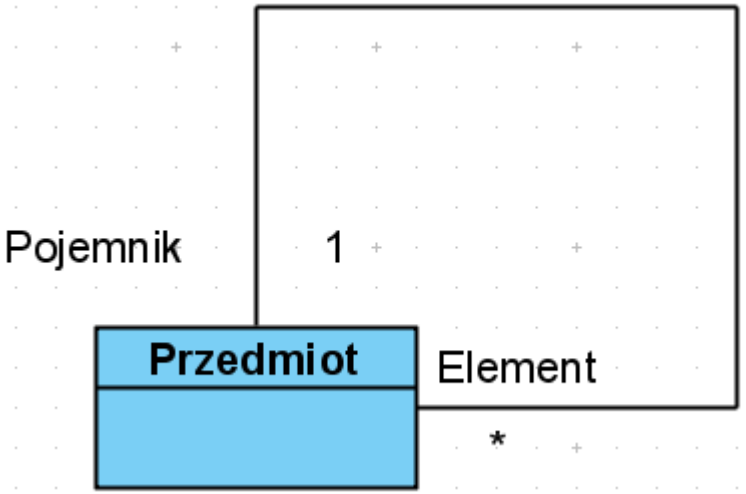
- Dwie klasy mogą być w odmienny sposób związane ze sobą w różnych kontekstach
- W efekcie może być więcej niż jedna asocjacja między klasami
- W takim wypadku każda powinna być nazwana

Asocjacja wielokrotna



Asocjacja zwrotna

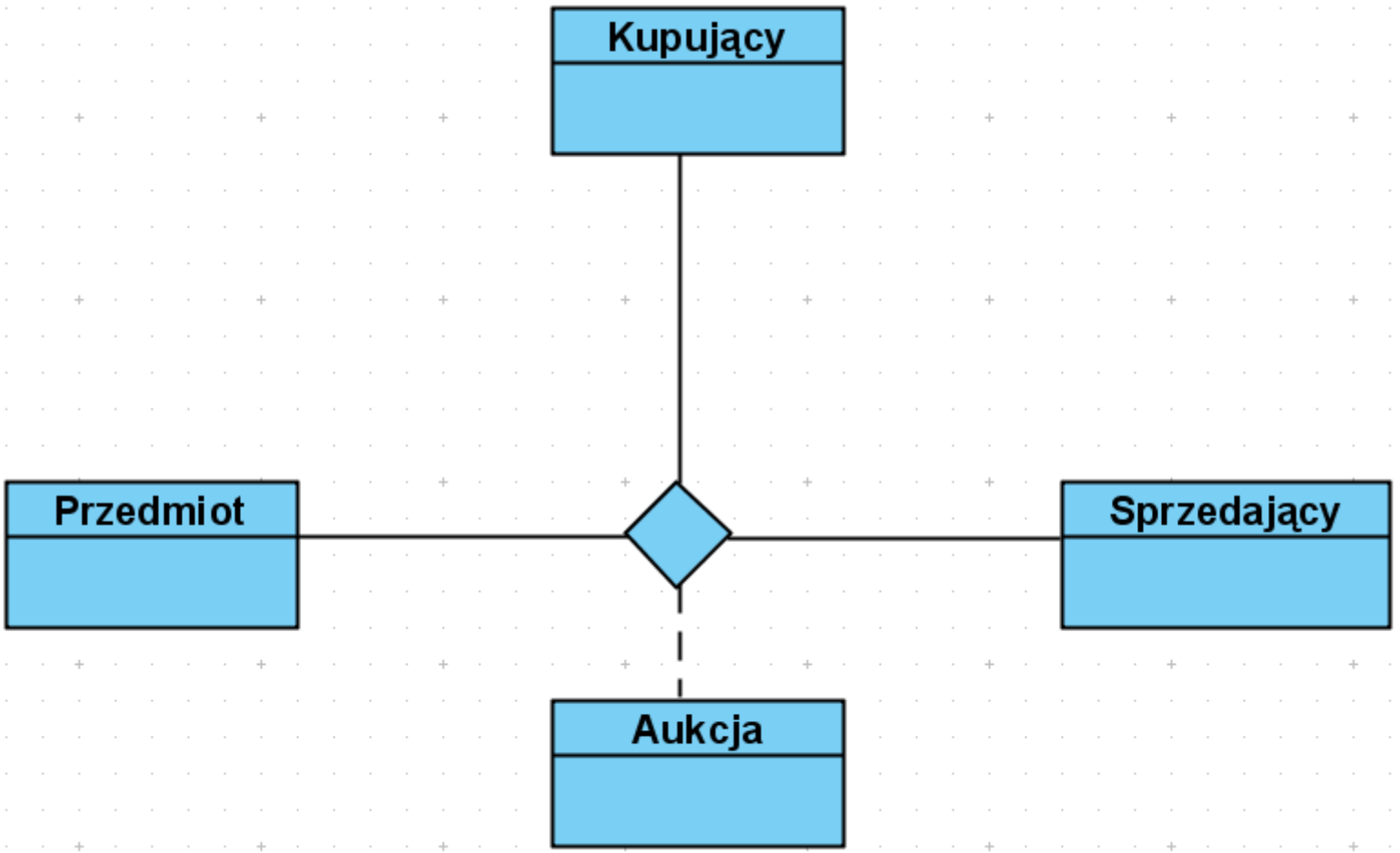
- Asocjacja może być zwrotna



Asocjacja N-arna

- Asocjacja może określać związek między więcej niż dwiema klasami
- Taka asocjacja może zawierać klasę asocjacji
- Nie należy jej mylić z asocjacją wielokrotną

Asocjacja N-arna

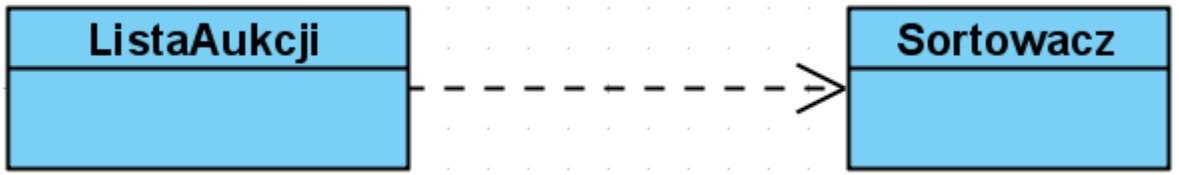


Podsumowanie asocjacji

- Asocjacja – związek między dwiema klasami (student – nauczyciel, sprzedający – kupujący)
- Słaba agregacja – obiekty jednej klasy należą do obiektów drugiej, ale fragmenty mogą istnieć bez całości (zamówienie – produkty, biblioteka – książki)
- Silna agregacja (kompozycja) – j. w., ale część nie może istnieć bez całości (wielokąt – jego wierzchołki, zamówienie – adres dostarczenia)

Zależność

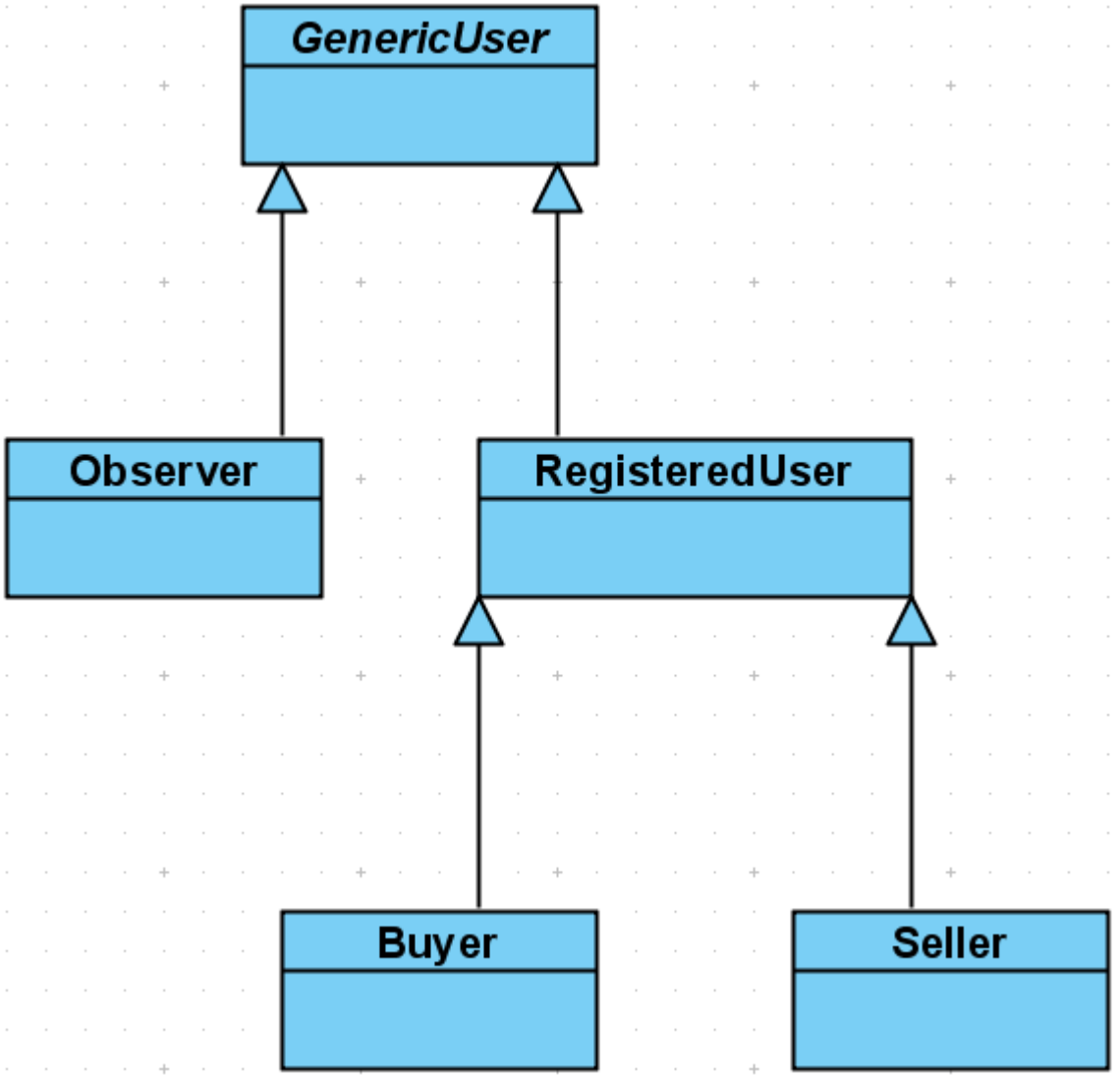
- Oznacza że jedna klasa (klient) w jakiś sposób używa innej klasy (dostawca)
- Jest obrazowana linią przerywaną zakończoną strzałką wskazującą na dostawcę



Uogólnienie

- Ściśle powiązane z koncepcją dziedziczenia w programowaniu zorientowanym obiektowo.
- Można stosować m. in klasy abstrakcyjne
  - nie mają one instancji (obiektów)
  - są wyróżnione nazwą pisaną kursywą

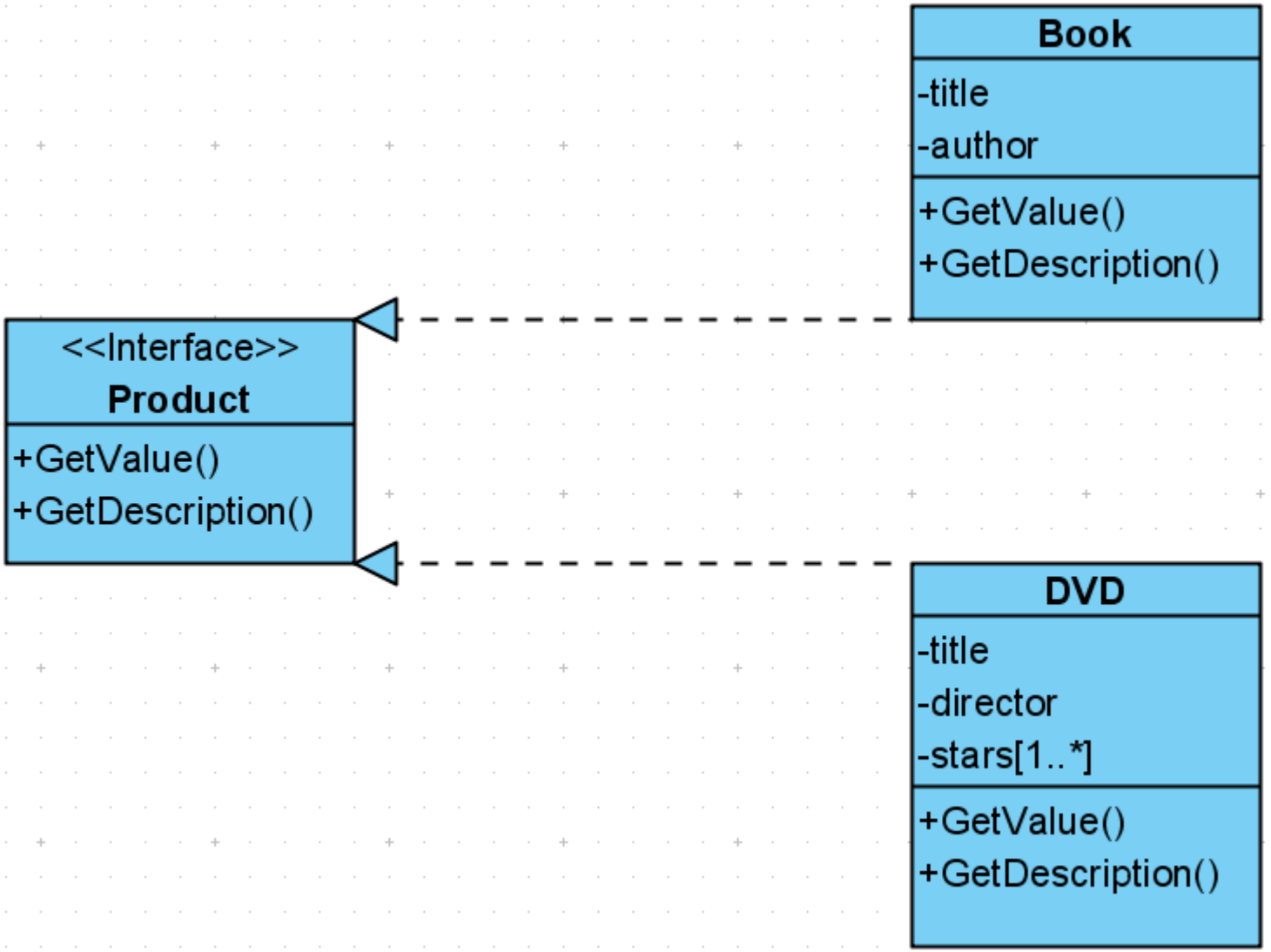
Uogólnienie



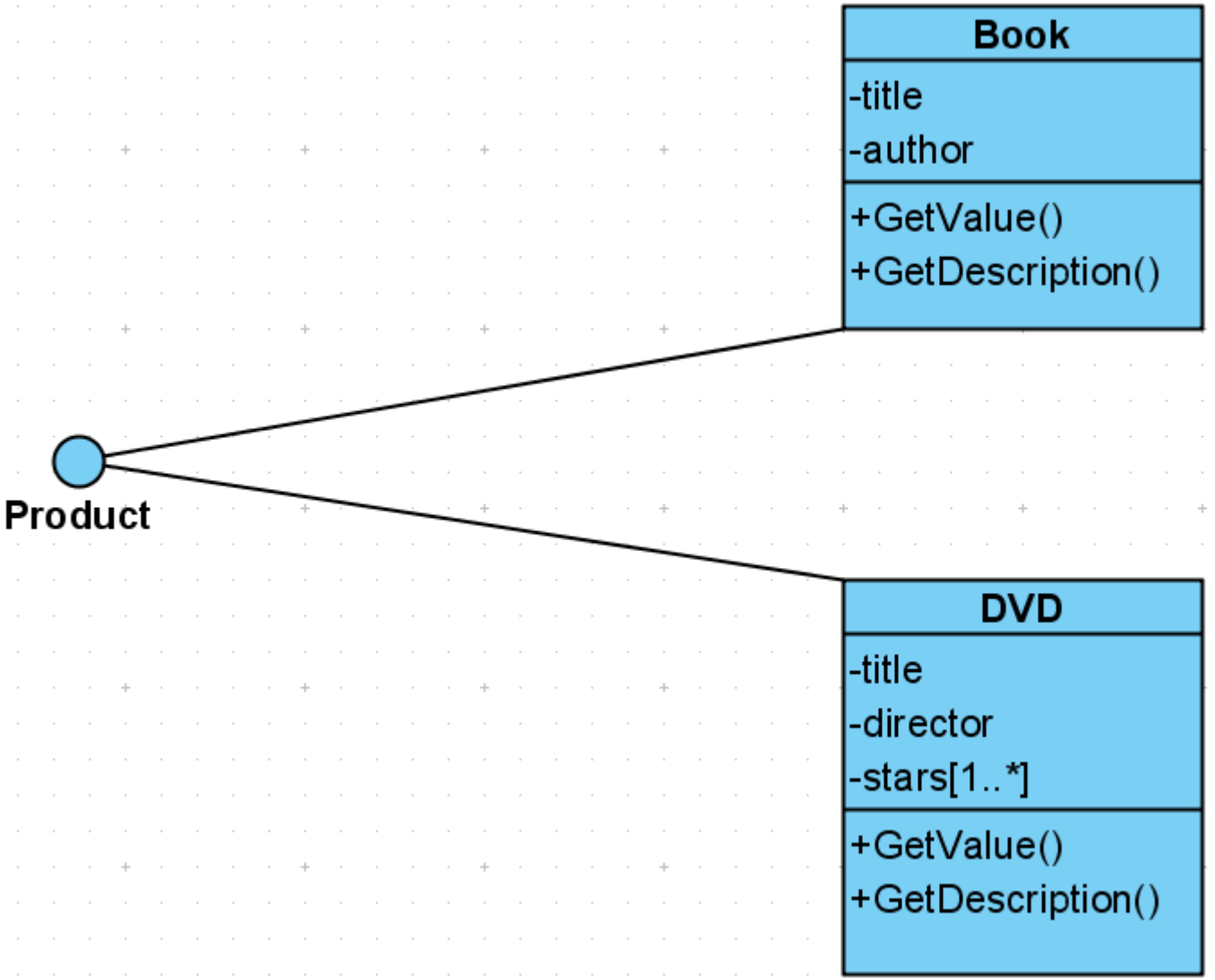
Realizacja

- Związek między interfejsem i jego implementacją
- Obrazowana linią przerywaną z pustą strzałką wskazującą od klasy do interfejsu
- Interfejs może być zobrazowany jako prostokąt z operacjami (podobnie do klasy) bądź jako kółko

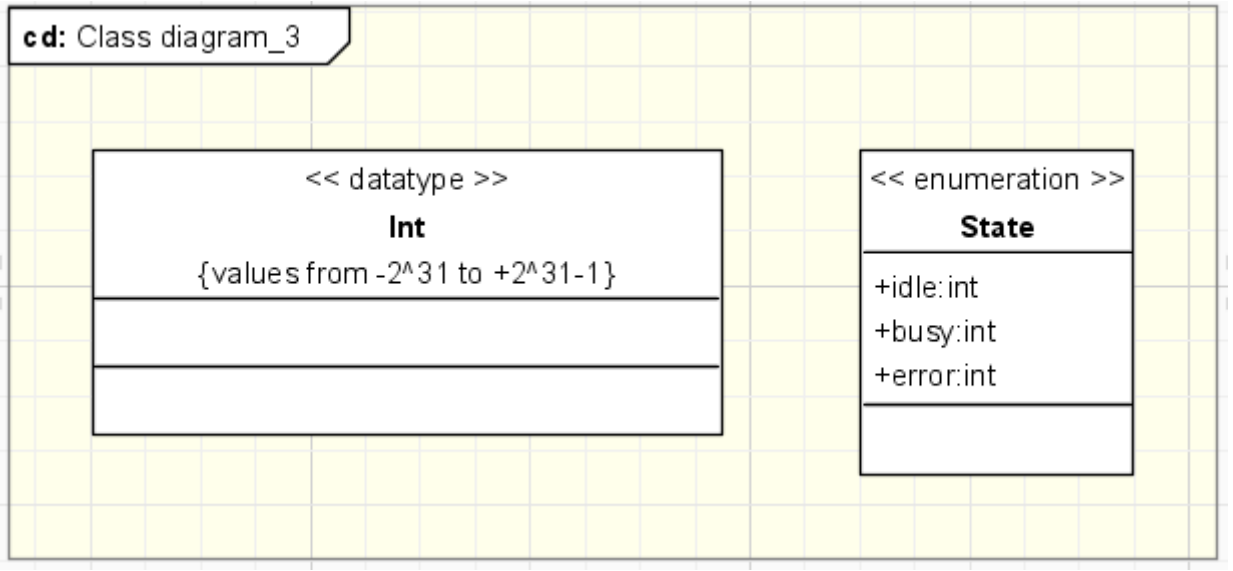
Realizacja


Book
-title
-author
+GetValue()
+GetDescription()
<<Interface>>
Product
+GetValue()
+GetDescription()
DVD
-title
-director
-stars[1..*]
+GetValue()
+GetDescription()

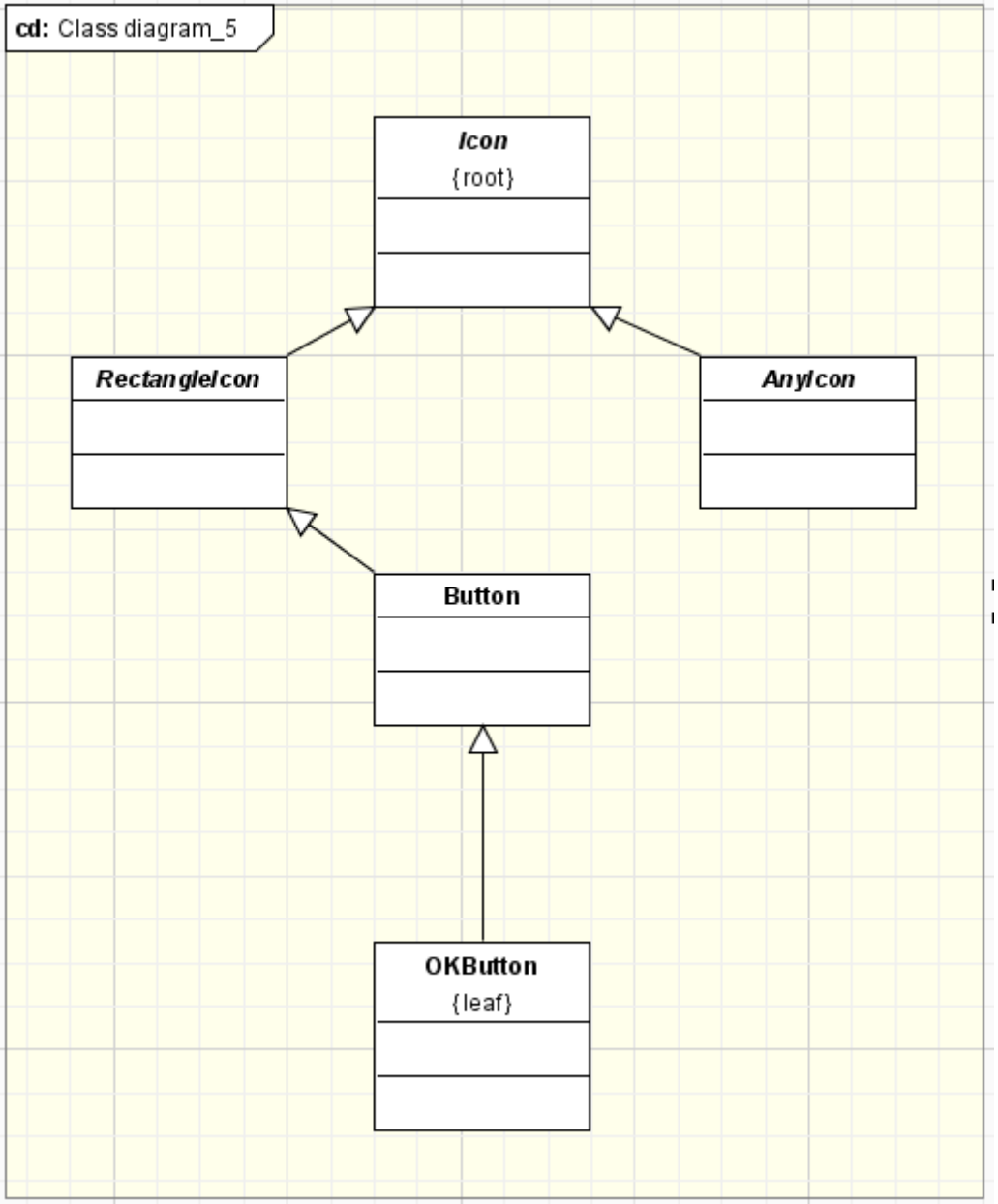
Realizacja


Book
-title
-author
+GetValue()
+GetDescription()
Product
DVD
-title
-director
-stars[1..*]
+GetValue()
+GetDescription()

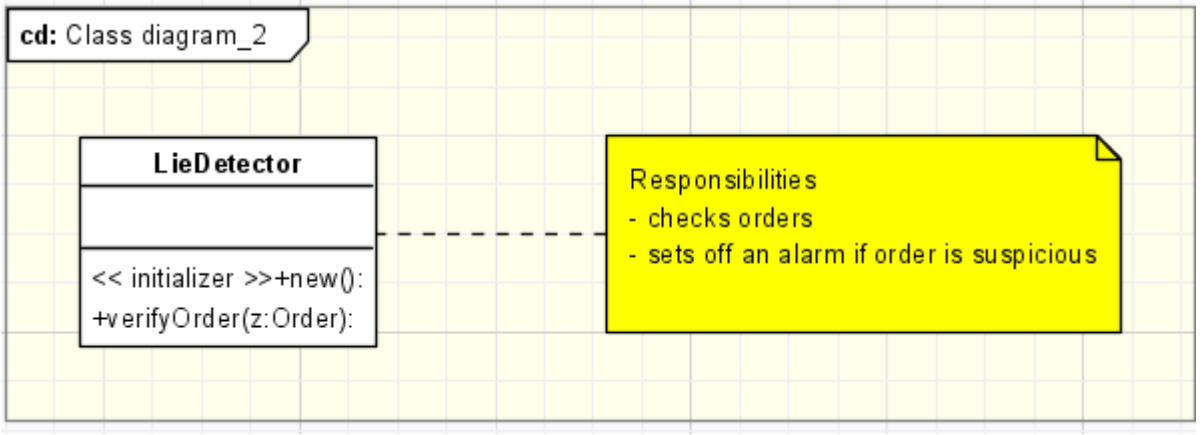
Przykładowe diagramy



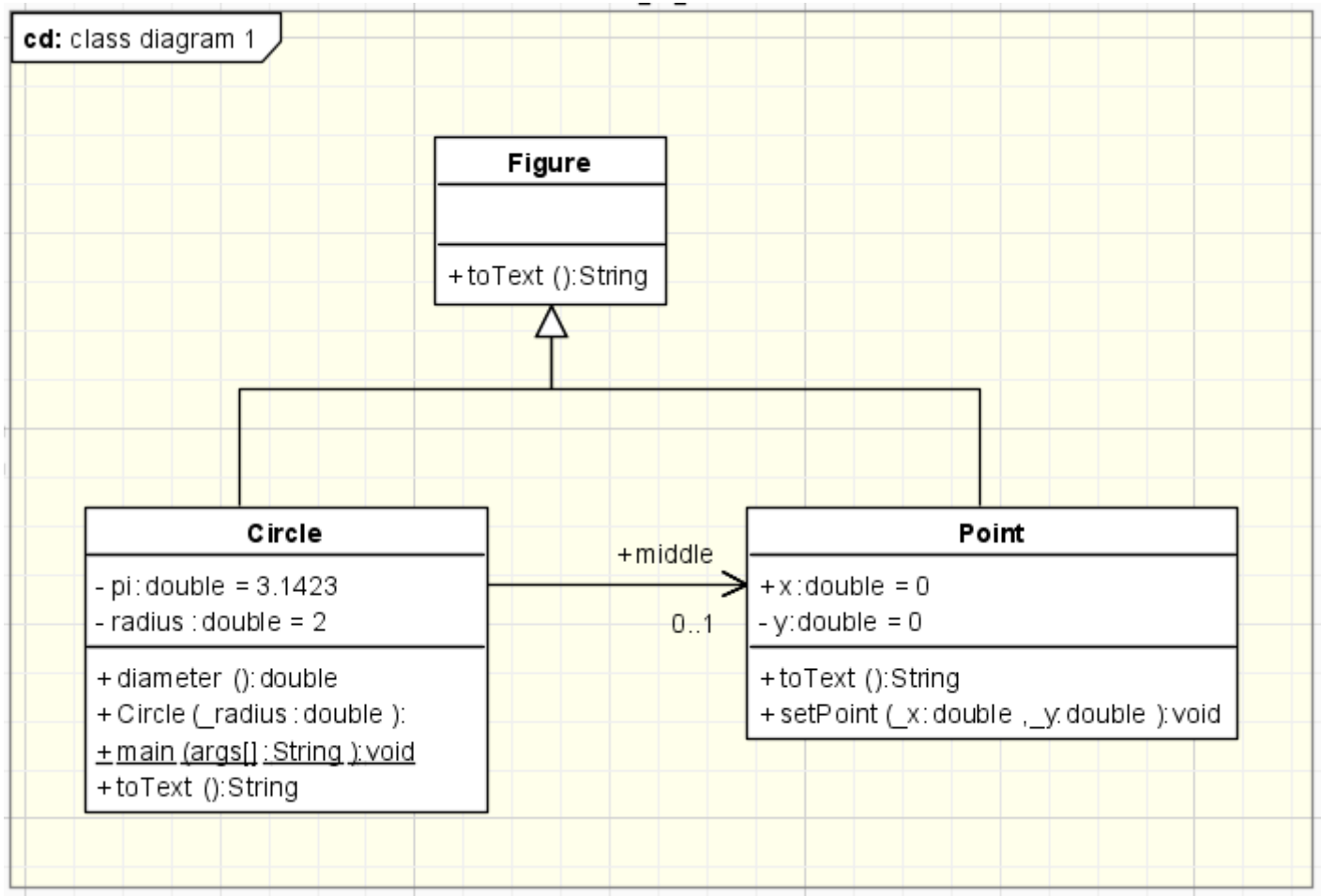
Przykładowe diagramy



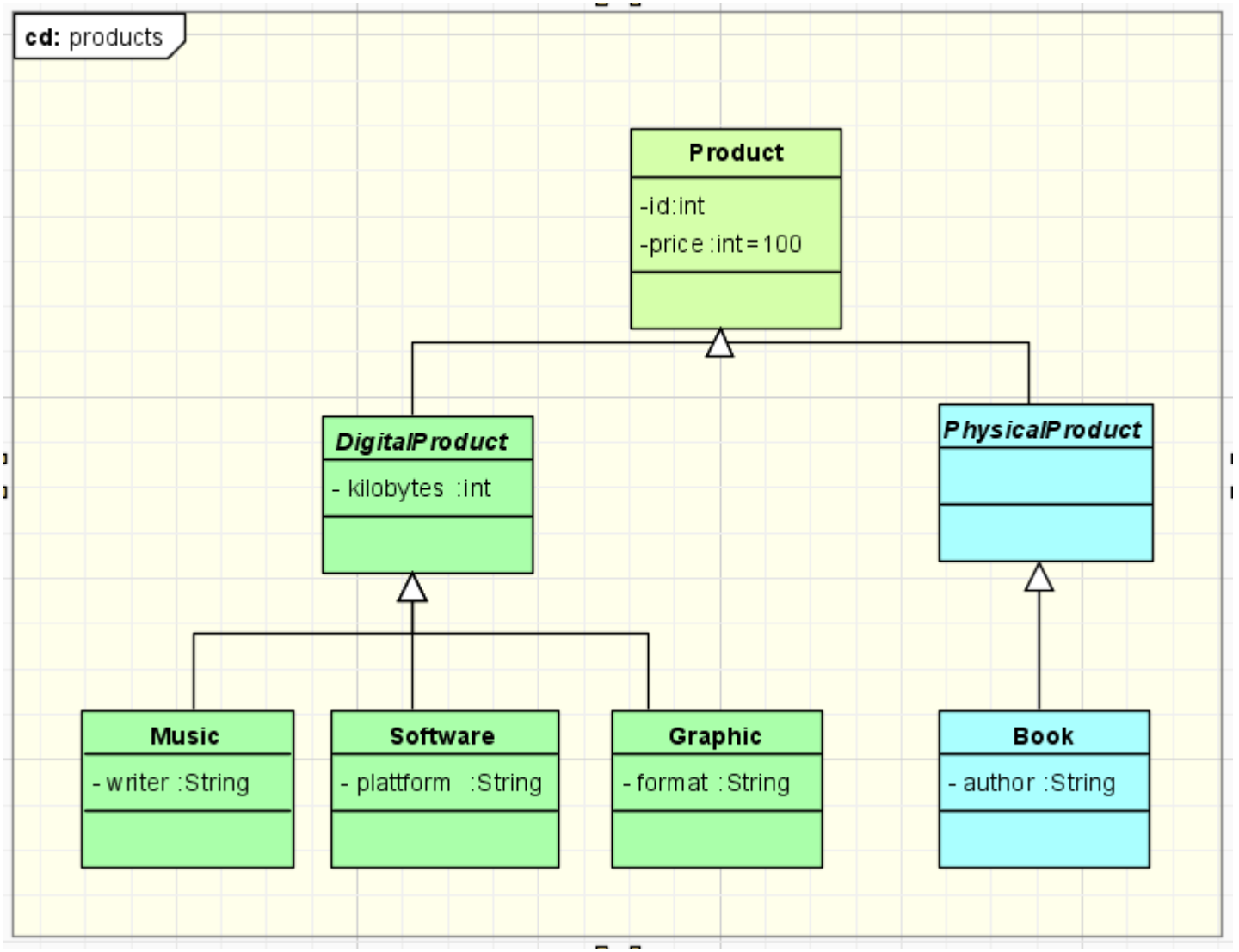
Przykładowe diagramy



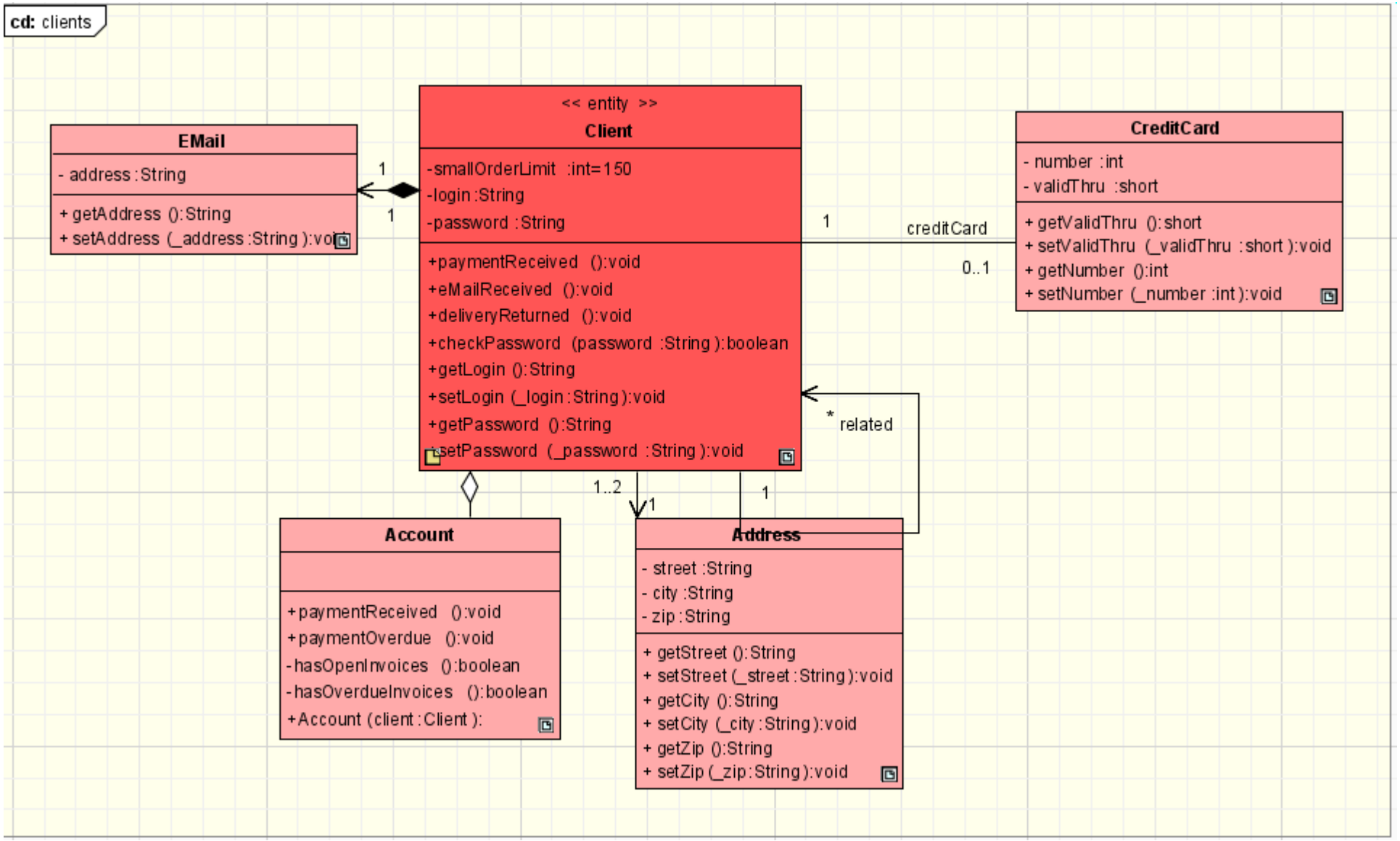
Przykładowe diagramy



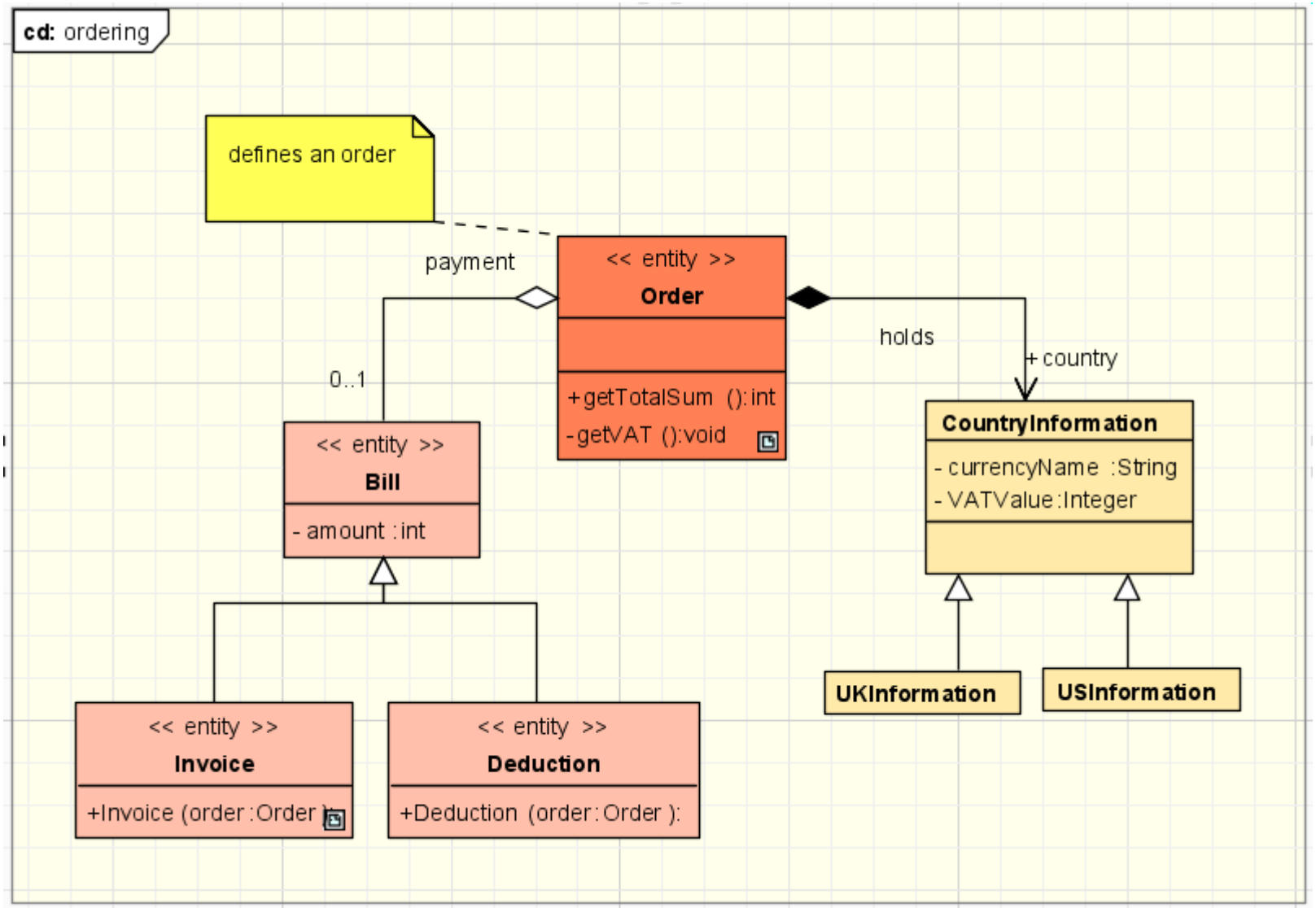
Przykładowe diagramy



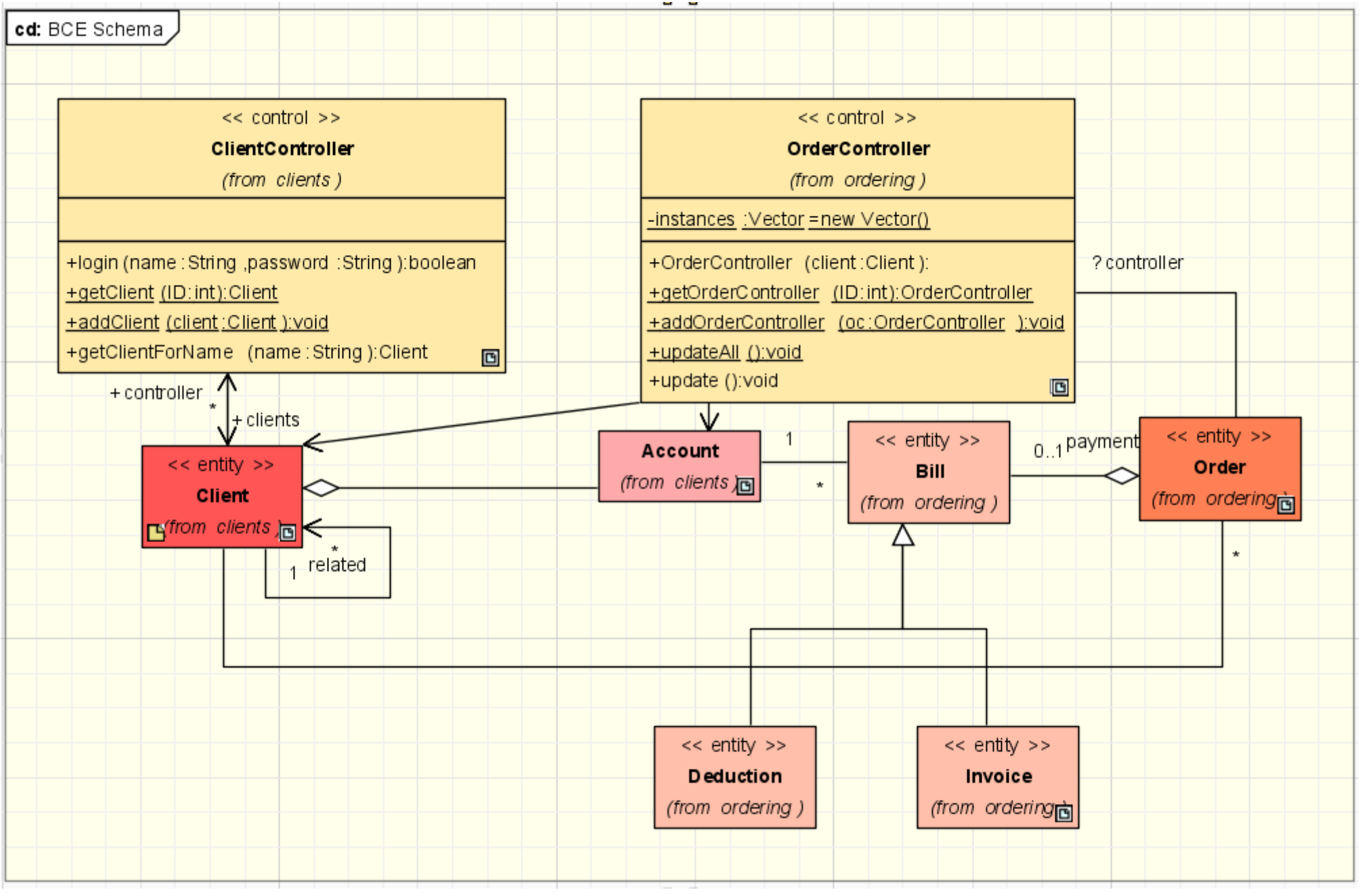
Przykładowe diagramy



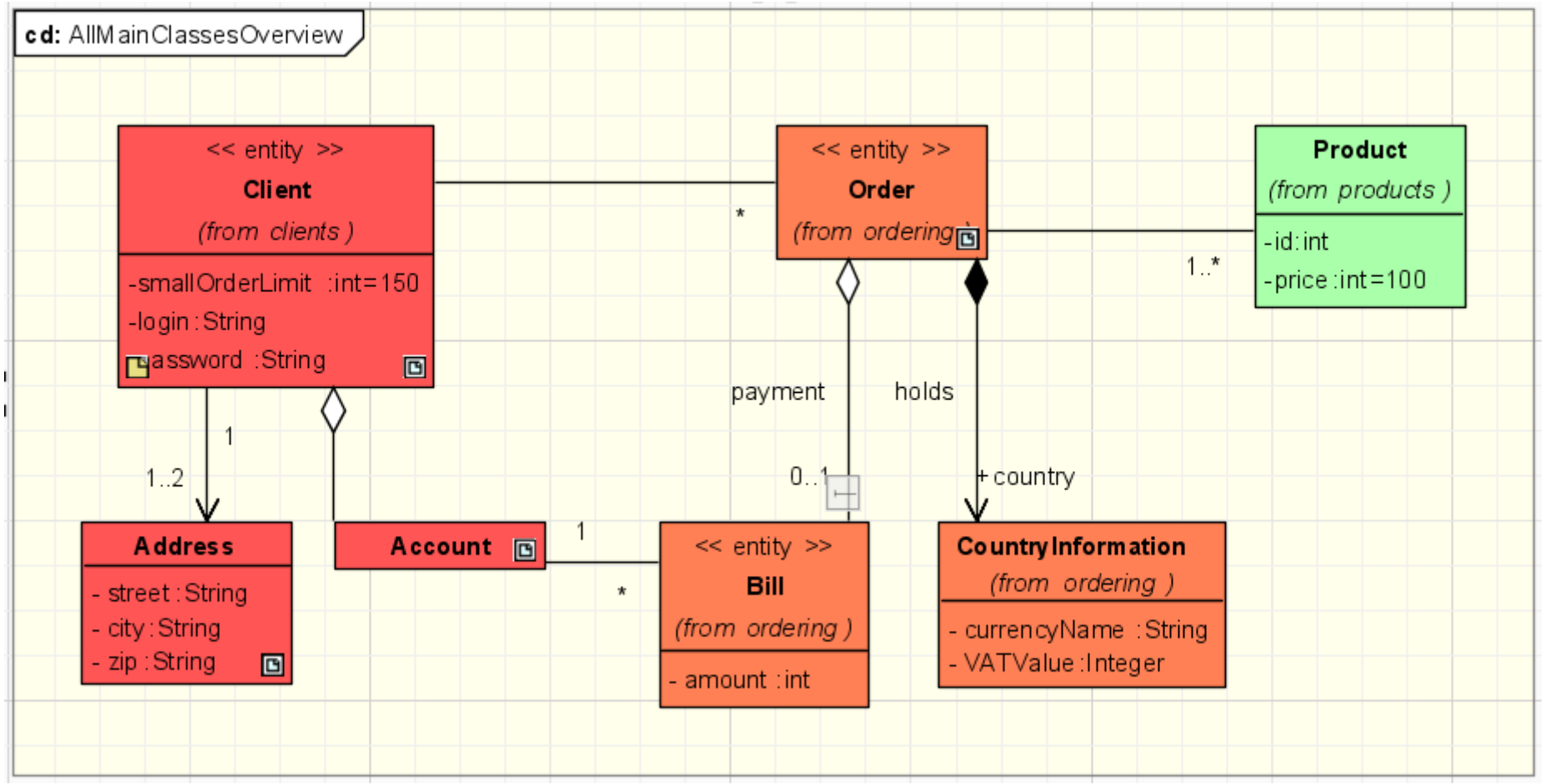
Przykładowe diagramy



Przykładowe diagramy



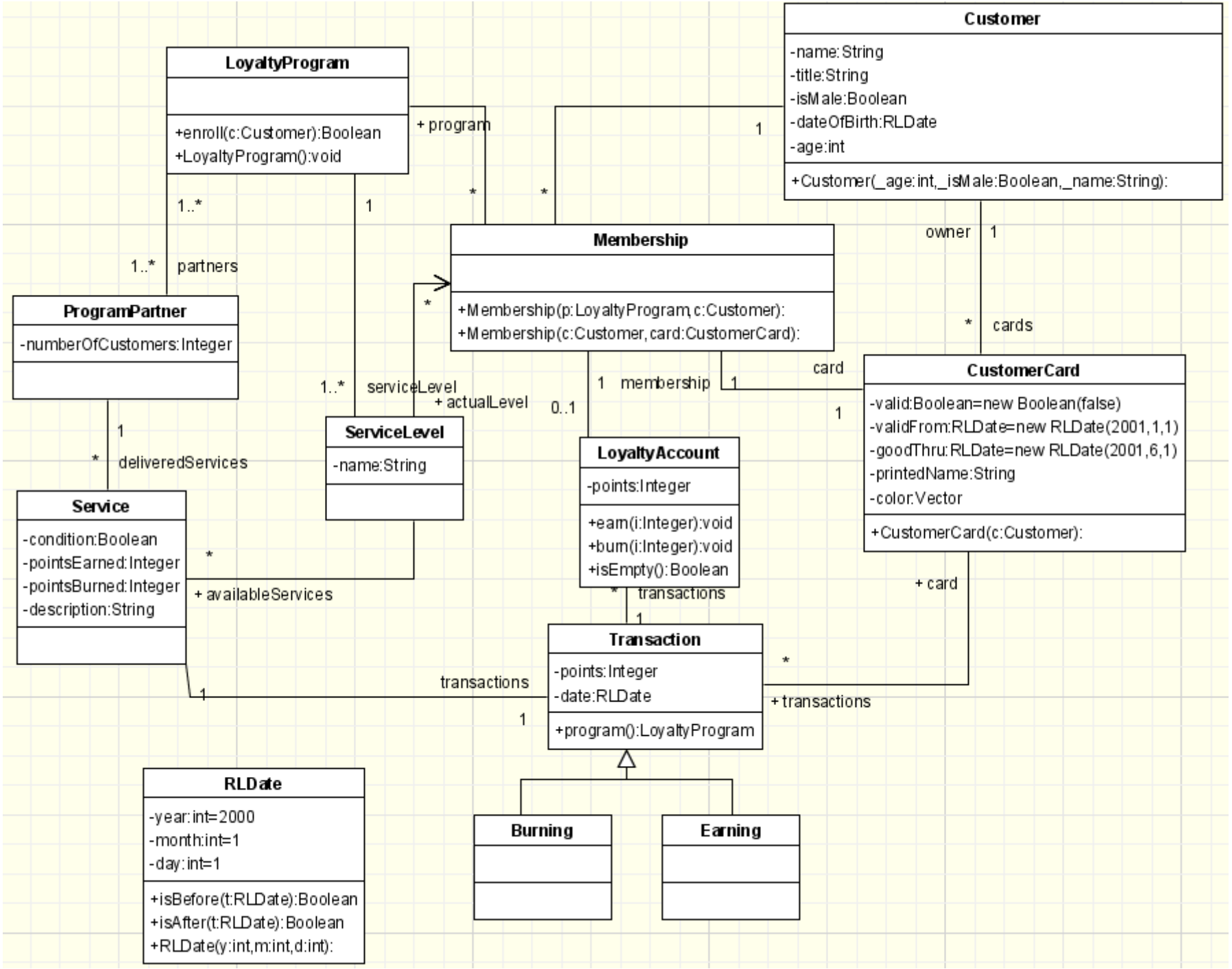
Przykładowe diagramy



Generacja kodu

```
public class Client {
    private products.int smallOrderLimit = 150;
    private String login;
    private String password;
    public clients.CreditCard creditCard;
    public clients.EMail eMail;
    public java.util.Collection address = new java.util.TreeSet();
    public java.util.Collection client = new java.util.TreeSet();
    public ordering.void paymentReceived() {
        return null;
    }
    public ordering.void eMailReceived() {
        return null;
    }
    public ordering.void deliveryReturned() {
        return null;
    }
    public boolean checkPassword(String password) {
        return false;
    }
      /*...*/
    public java.util.Collection order = new java.util.ArrayList();
    clients.Account account;
    public clients.ClientController controller;
 }
```

Przykładowe diagramy



Diagramy czynności (activity)

- Opisują dynamikę systemu (klas- statykę)
- Prezentują przepływ danych i sterowania
- Mogą być stosowane do modelowania:
  - Procesów biznesowych
  - Scenariuszy przypadków użycia
  - Algorytmów
  - Operacji

Elementy składowe

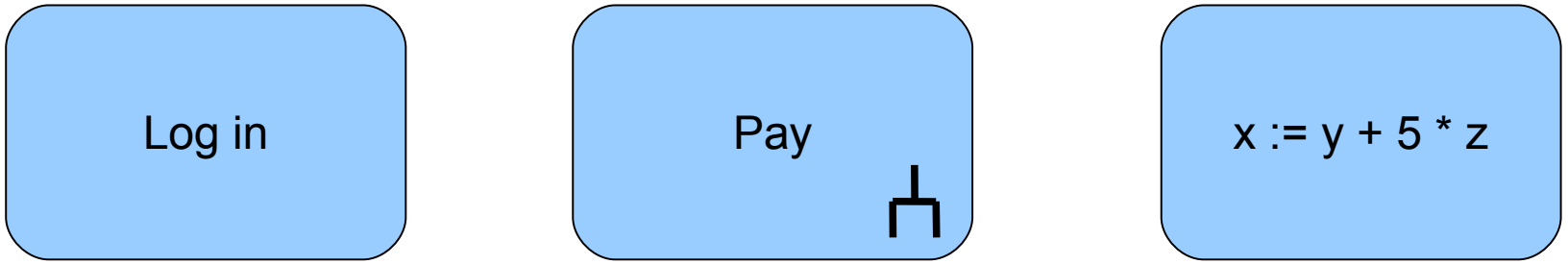
- Czynność
- Akcja
- Przepływ sterowania
- Początek
- Koniec
- Zakończenie przepływu

Czynność

- Reprezentuje złożony proces, algorytm
- Aby poprawić czytelność, nie wszystkie elementy muszą być przedstawione
- Może zostać (na osobnym diagramie) doprecyzowana, poprzez kolejny diagram czynności, opisujący jej “wnętrze” - powstaje struktura hierarchiczna
- Dekompozycja może przebiegać do poziomu akcji – najmniejszej, niepodzielnej jednostki

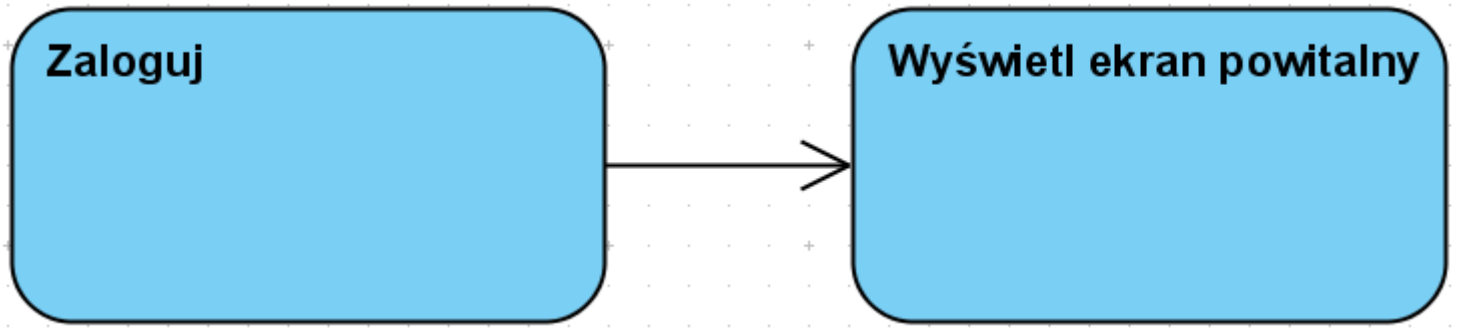
Czynność i akcja

- Symbolem jest prostokąt z zaokrąglonymi rogami
- Czasem czynności posiadające powiązane z nimi poddiagramy posiadają specjalny znacznik
- Akcje mają podobny symbol (choć może się różnić wielkością)



Przepływ sterowania

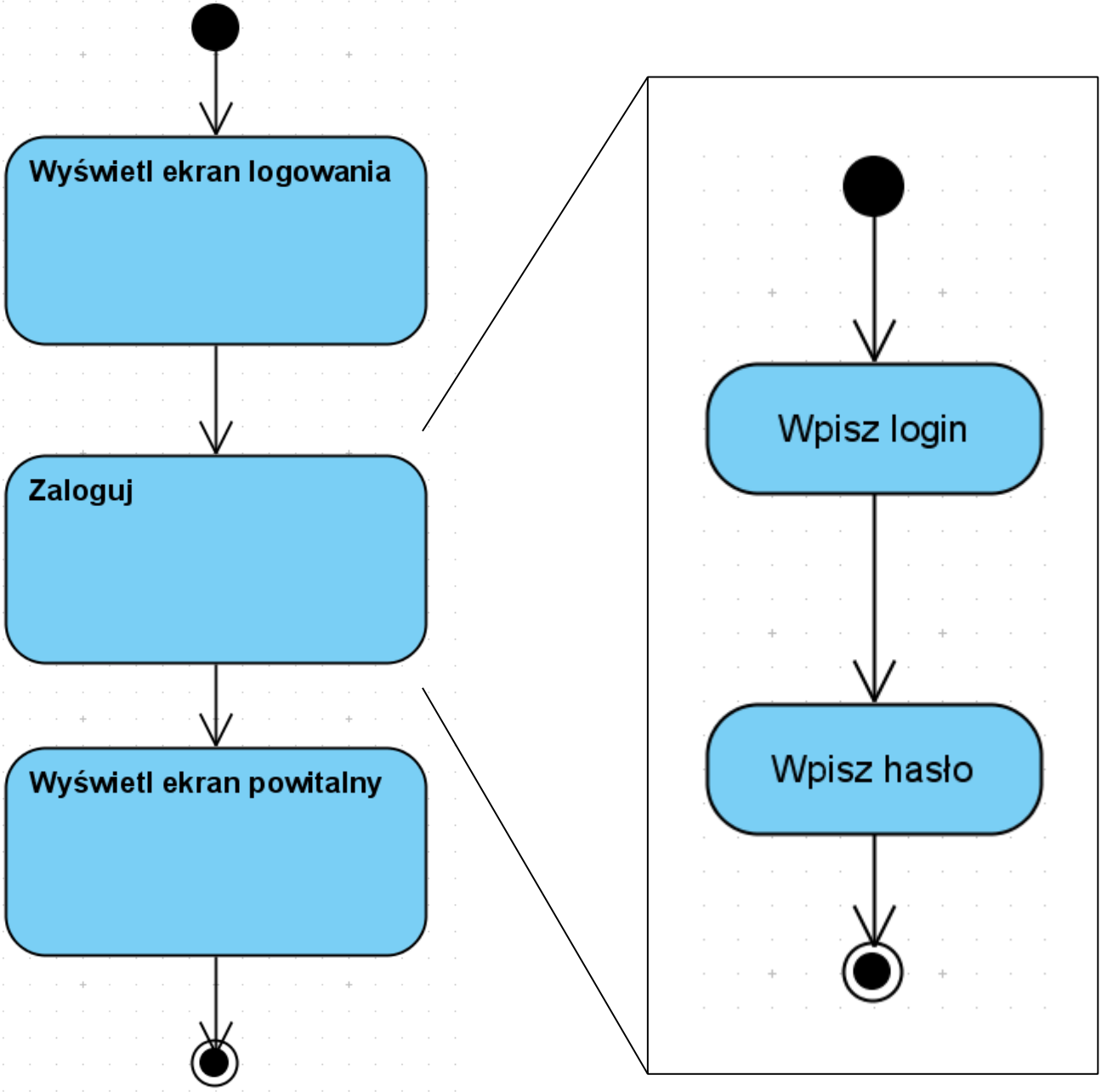
- Jest to związek między dwiema czynnościami/akcjami mówiący, że po zakończeniu jednej sterowanie będzie przekazane do drugiej
- Symbolem jest strzałka



Początek, koniec, zakończenie przepływu

- Początek określa początek przepływu sterowania. Zwykle jeden na diagram. Symbolem jest pełne kółko
- Koniec oznacza zakończenie wszystkich przepływów w diagramie. Może być więcej niż jeden. Symbolem jest małe pełne kółko wewnątrz większego pustego
- Zakończenie przepływu oznacza koniec jednego przepływu. Może być więcej niż jedno. Symbolem jest kółko z krzyżykiem (X)

Prosty diagram



Decyzja i złączenie

- Alternatywne ścieżki przepływu sterowania mogą zostać opisane przy pomocy węzłów decyzji i złączenia
- Węzeł decyzji ma jeden przepływ wejściowy i wiele wyjściowych. Tylko jedno wyjście może być wybrane w danej chwili
- Węzeł złączenia ma wiele wejść i jedno wyjście
- Symbolem obu węzłów jest romb

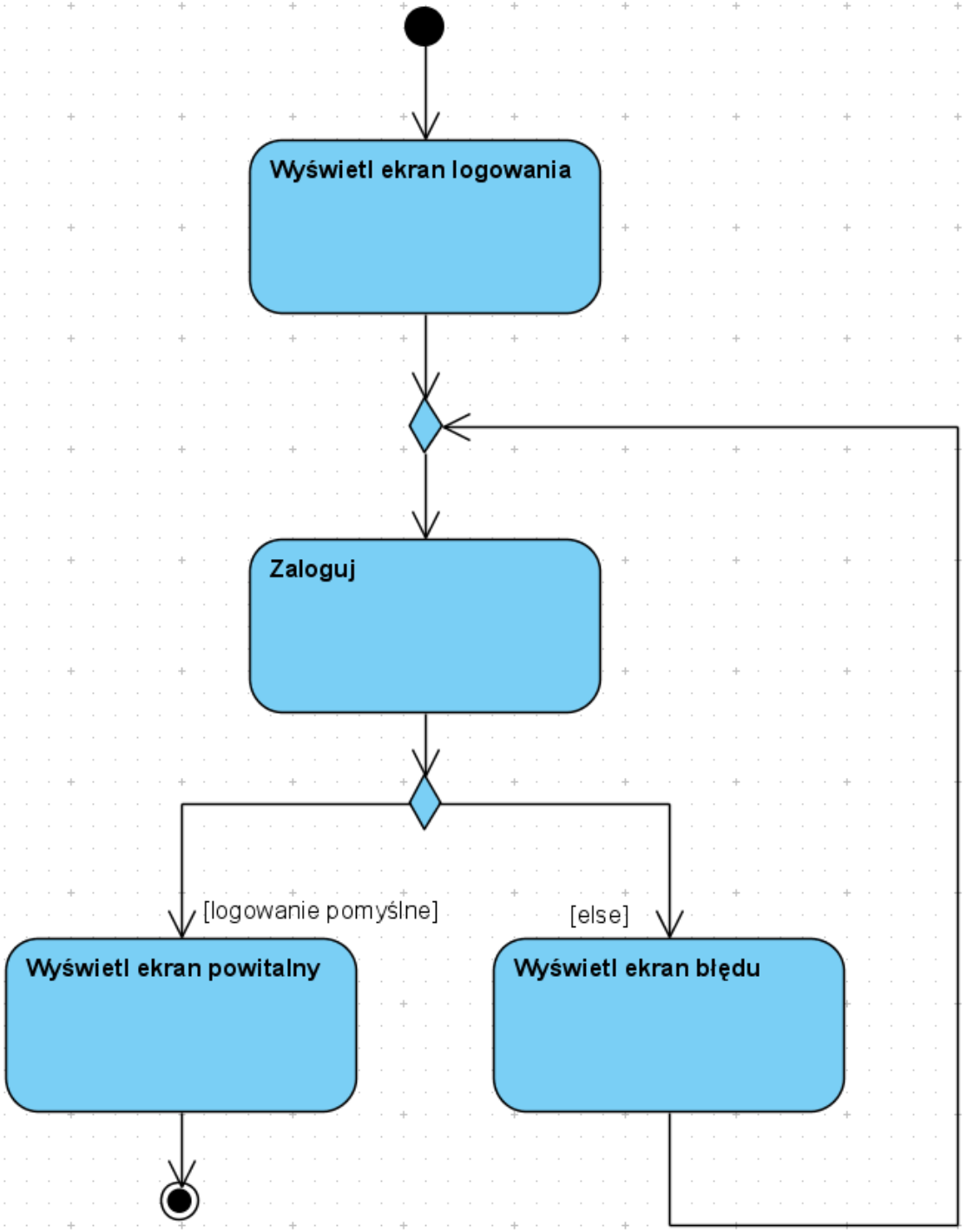
Decyzja

- Wybór wyjścia następuje na podstawie warunku
- Warunek jest umieszczany przy wyjściu, w nawiasach kwadratowych
- Wszystkie warunki muszą się wzajemnie wykluczać
- Jeden z warunków może zostać zastąpiony słowem kluczowym *else*

Złączenie

- Każdy przepływ pojawiający się na wejściu zostanie natychmiast przekazany na wyjście – brak synchronizacji

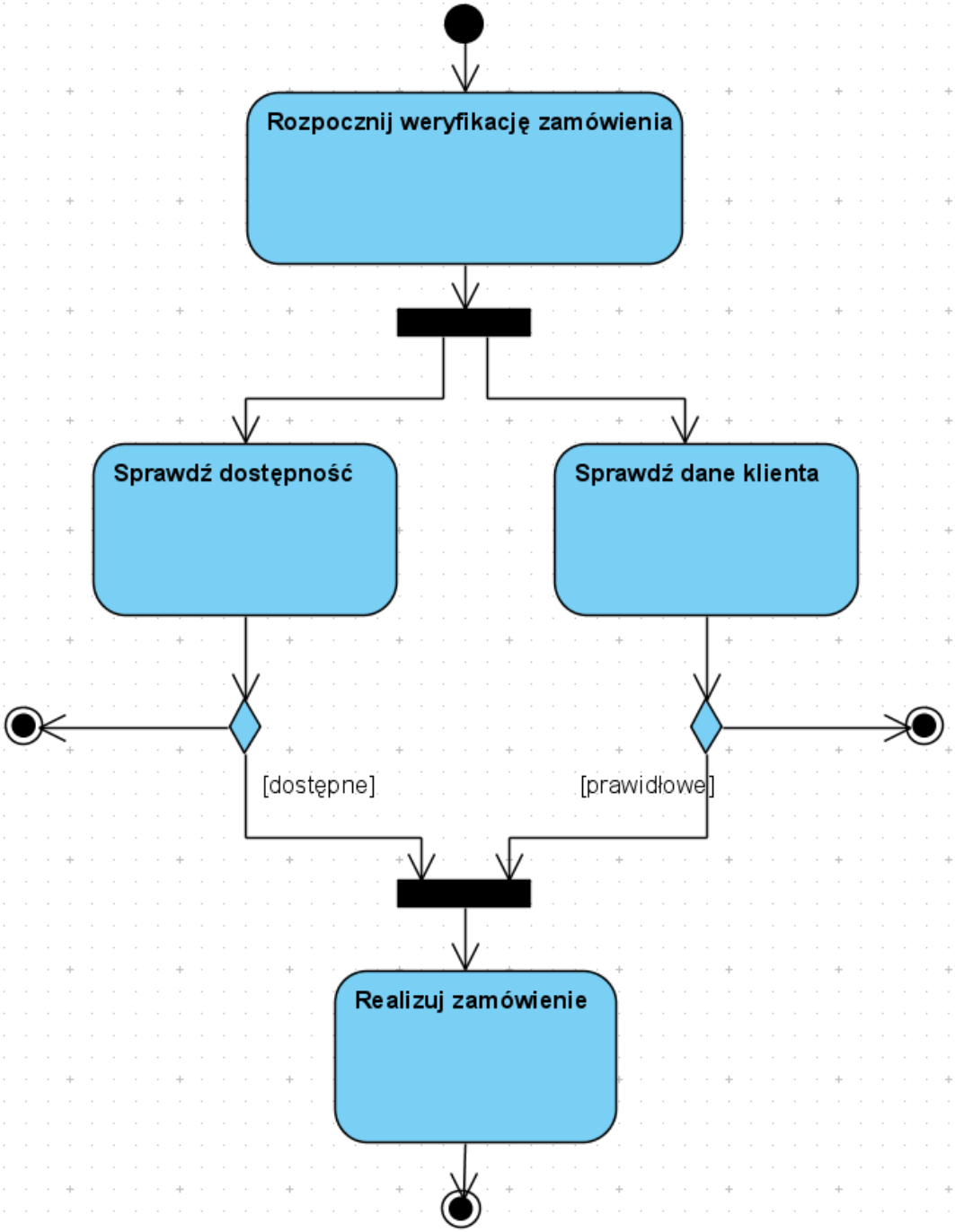
Przykładowy diagram



Przepływy równoległe

- Przepływy mogą być wykonywane równolegle
- Umożliwiają to węzły rozwidlenia i scalenia
- Rozwidlenie ma jedno wejście i wiele wyjść – przepływ wchodzący jest rozdzielany
- Scalenie ma wiele wejść i jedno wyjście. Przepływ zostanie przekazany do wyjścia tylko jeśli przepływy dotarły do wszystkich wejść – następuje synchronizacja
- Można też użyć specyfikacji złączenia - przepływ dotrze na wyjście jeśli będzie spełniony określony warunek

Sample diagram



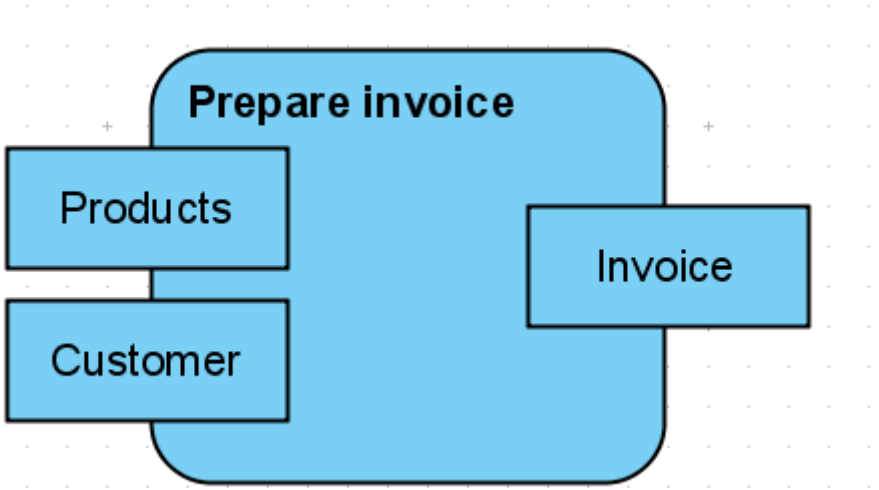
Przepływ obiektów

- Jako uzupełnienie można zaznaczyć przepływ obiektów
- Może być przydatny np. kiedy obiekt jest tworzony i przekazywany dalej do przetwarzania
- Obiekt musi być połączony z czynnością/akcją
- Symbolem jest prostokąt z nazwą
- Można też użyć tzw. przekaźników danych

Przepływ obiektów

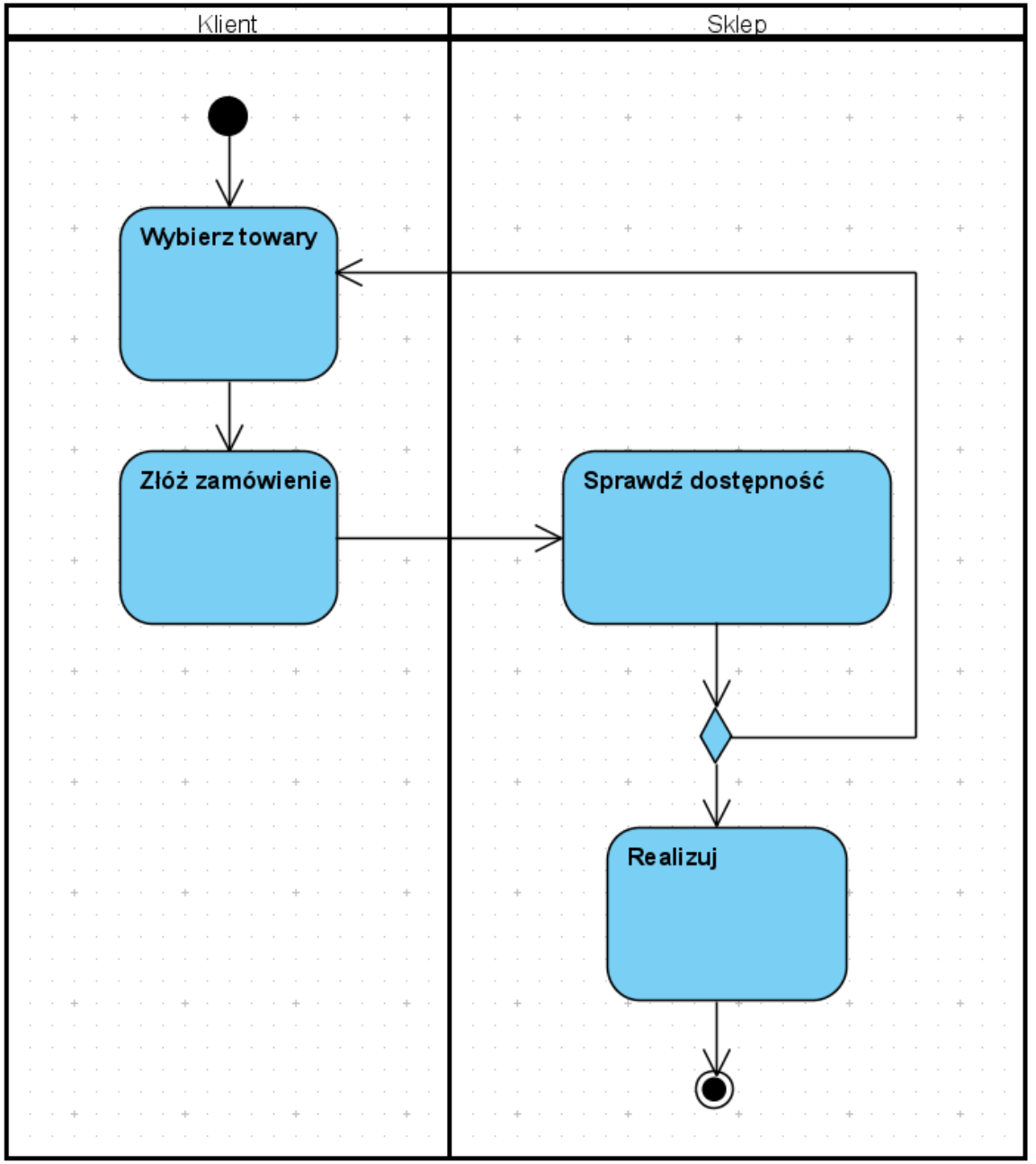
Parametr czynności

- Można określić, że obiekt jest parametrem czynności
- Jest to obrazowane prostokątem na krawędzi czynności



Partycje

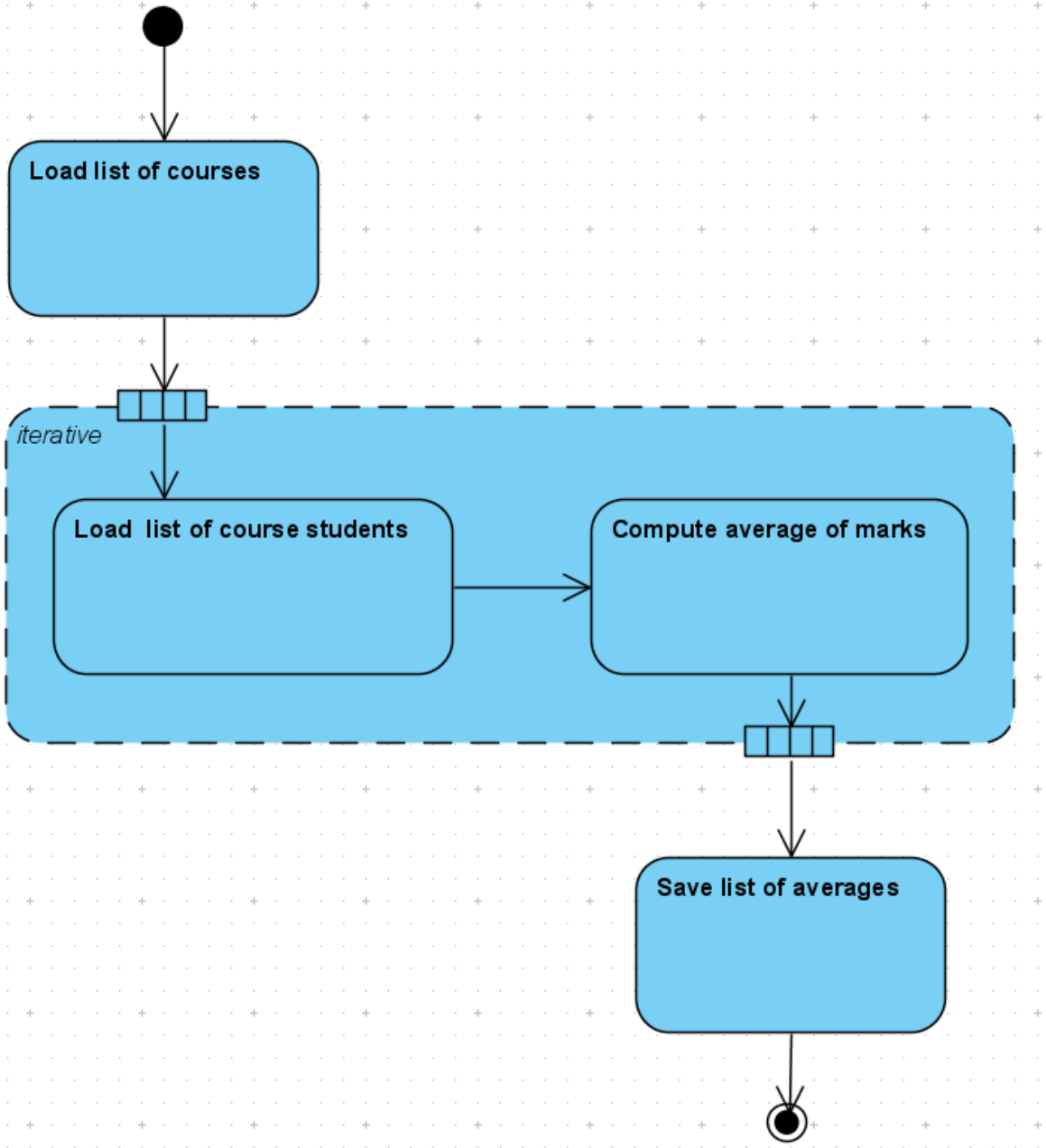
- Elementy diagramu mogą być pogrupowane w partycje



Region rozszerzenia

- Pozwala określić część diagramu wykonywaną wielokrotnie, w zależności od liczby elementów wejściowych
- Wejścia i wyjścia nazywane są węzłami rozszerzenia
- Wykonanie regionu określa łańcuch pisany kursywą, umieszczony wewnątrz regionu:
  - iterative
  - parallel
  - streaming

Przykładowy region rozszerzenia

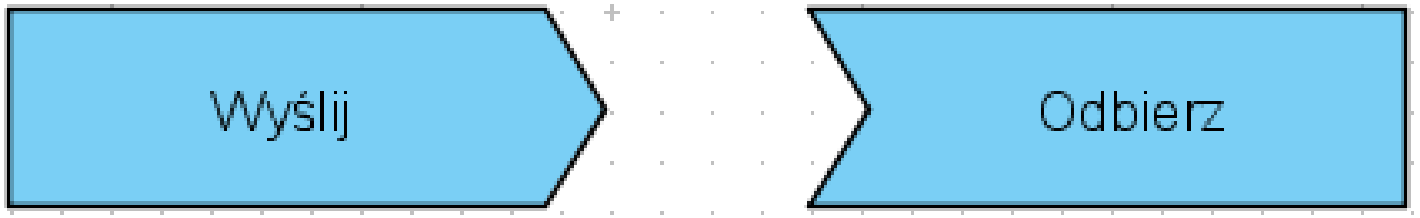


Obszar przerwania

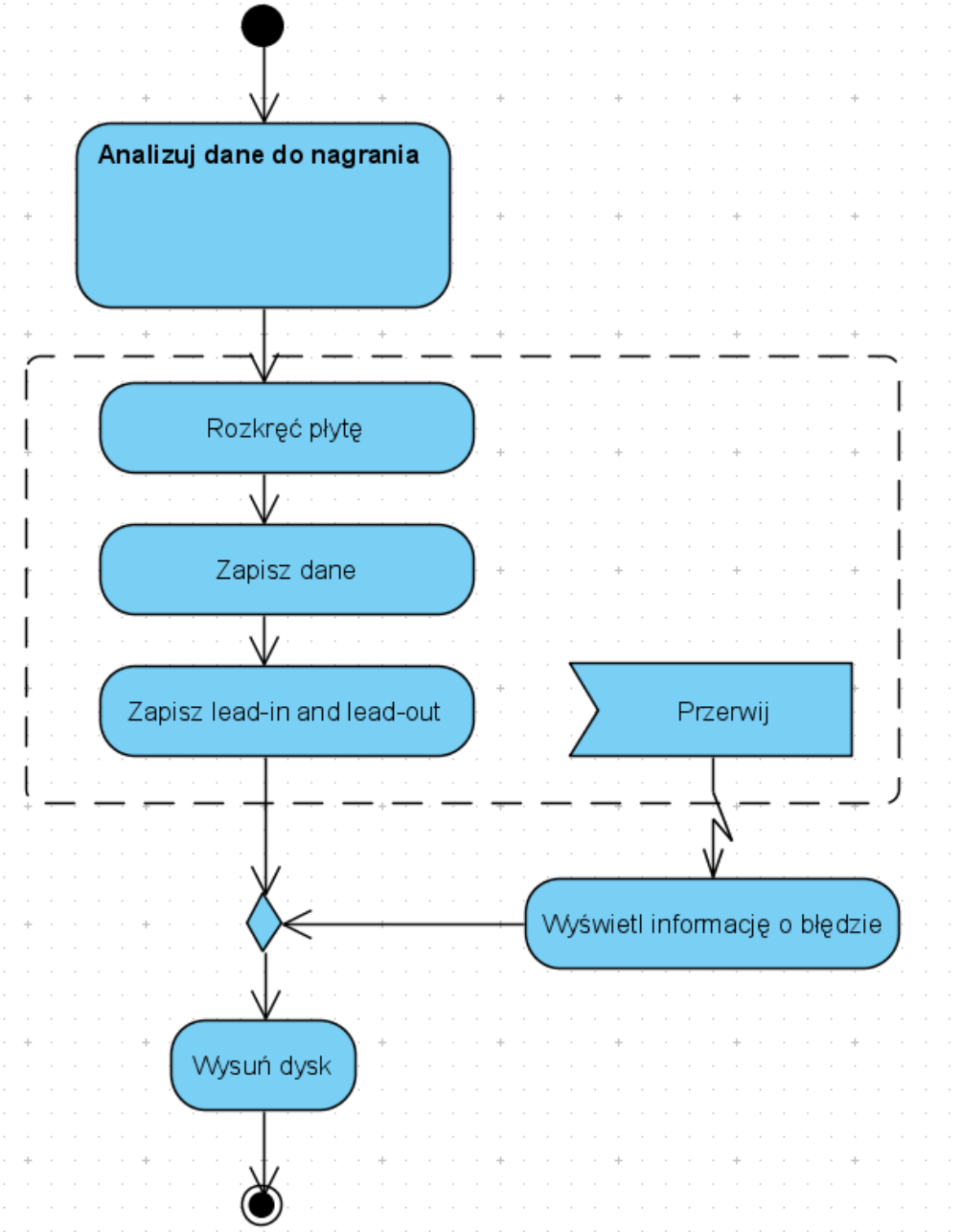
- Umożliwia wyznaczenie fragmentu diagramu którego wykonywanie może zostać natychmiast przerwane przy wystąpieniu zewnętrznego zdarzenia. Zostaje wtedy wykonany przepływ przerwania
- Wszystkie przepływy w regionie są przerywane, z wyjątkiem przepływu przerwania
- Przepływ przerwania zawsze zaczyna się w obszarze przerwania, kończy poza
- Użyteczne są symbole sygnałów

Sygnały

- Mogą służyć do opisu przetwarzania asynchronicznego
- Można:
  - Wysłać sygnał
  - Odebrać sygnał



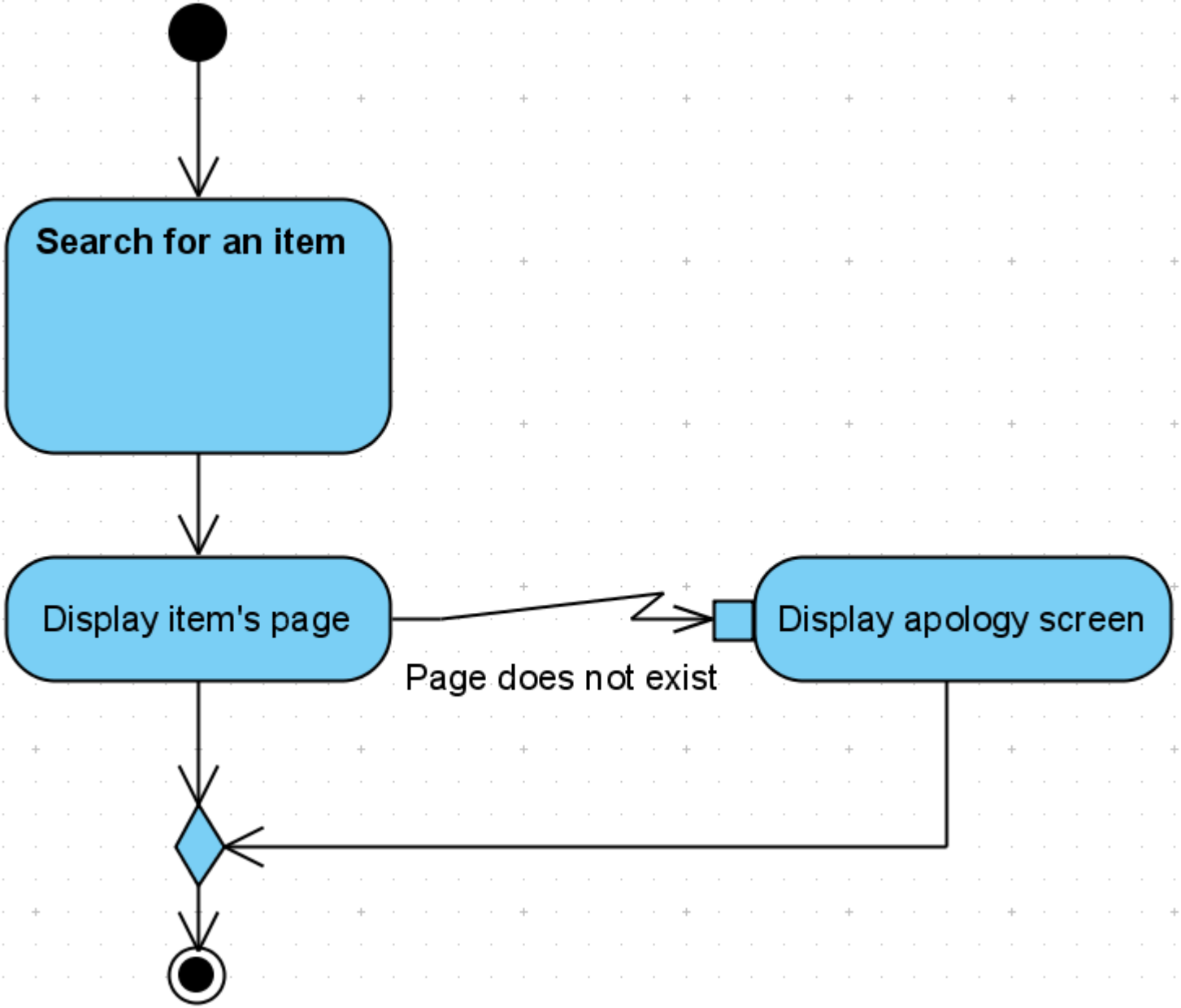
Obszar przerwania



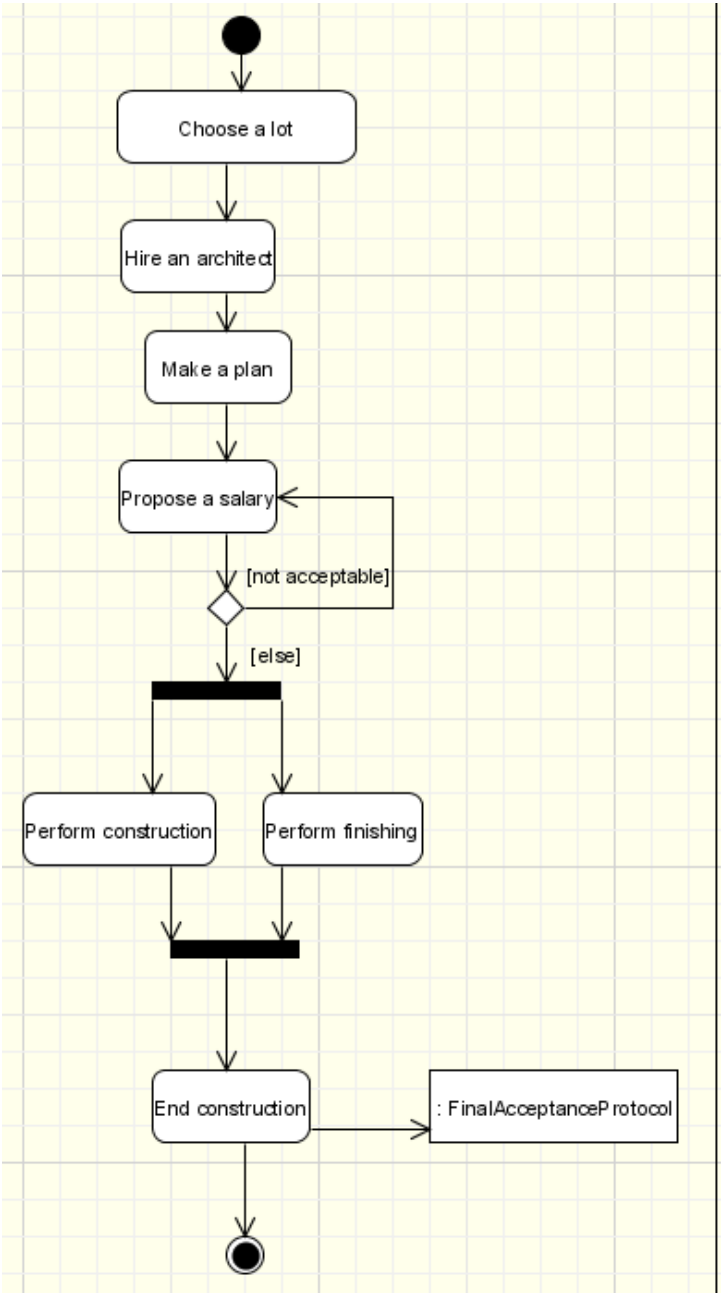
Obsługa wyjątków

- Pozwalają modelować czynność wykonywaną w przypadku wystąpienia błędu
- W przypadku błędu sterowanie jest przekazywane do procedury obsługi wyjątku
- Procedura musi być nazwana

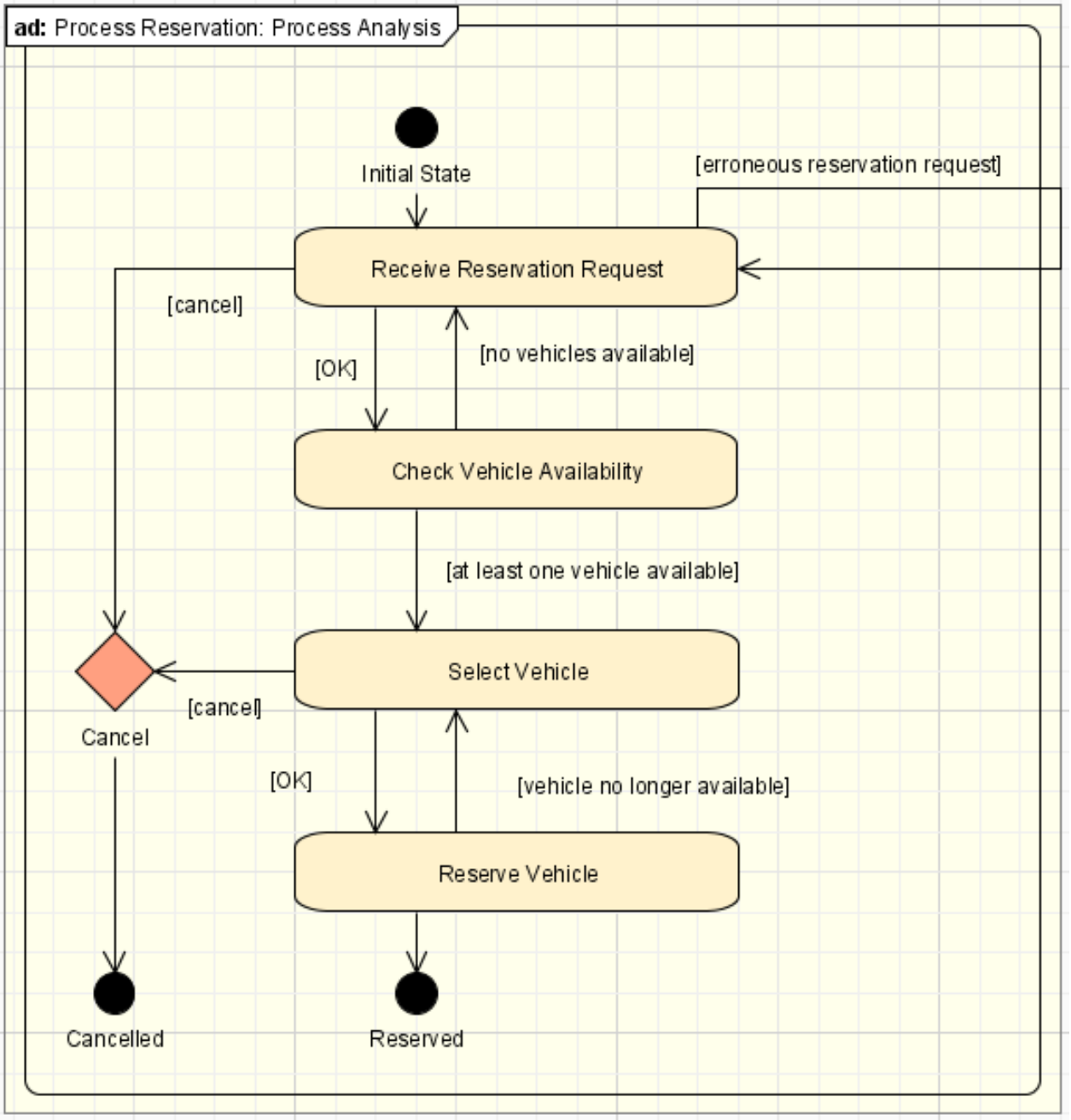
Obsługa wyjątków



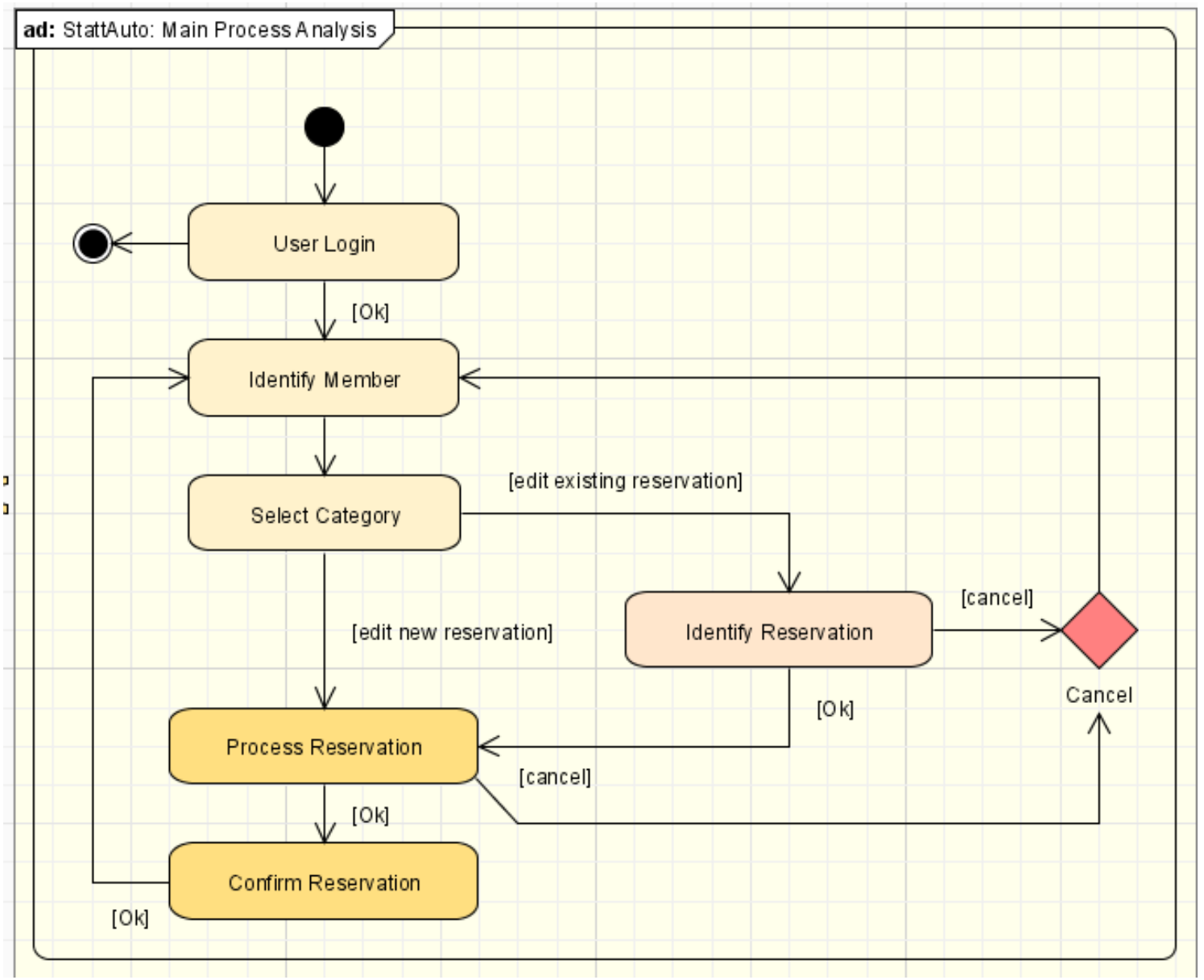
Przykładowe diagramy



Przykładowe diagramy



Przykładowe diagramy



Przykładowe diagramy

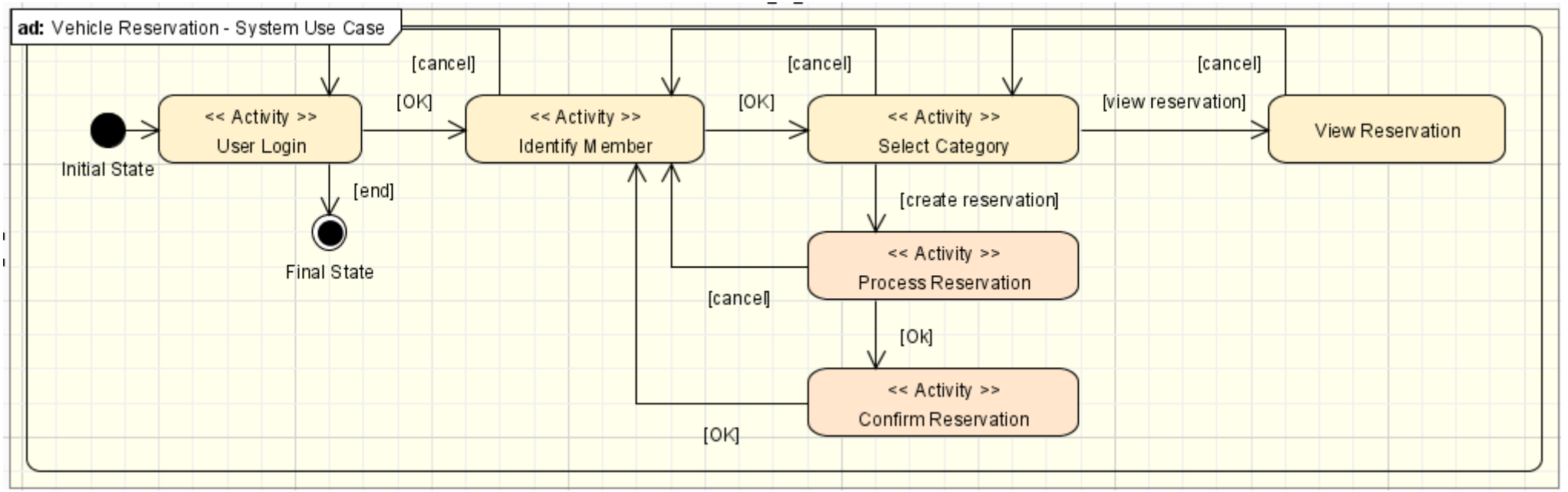


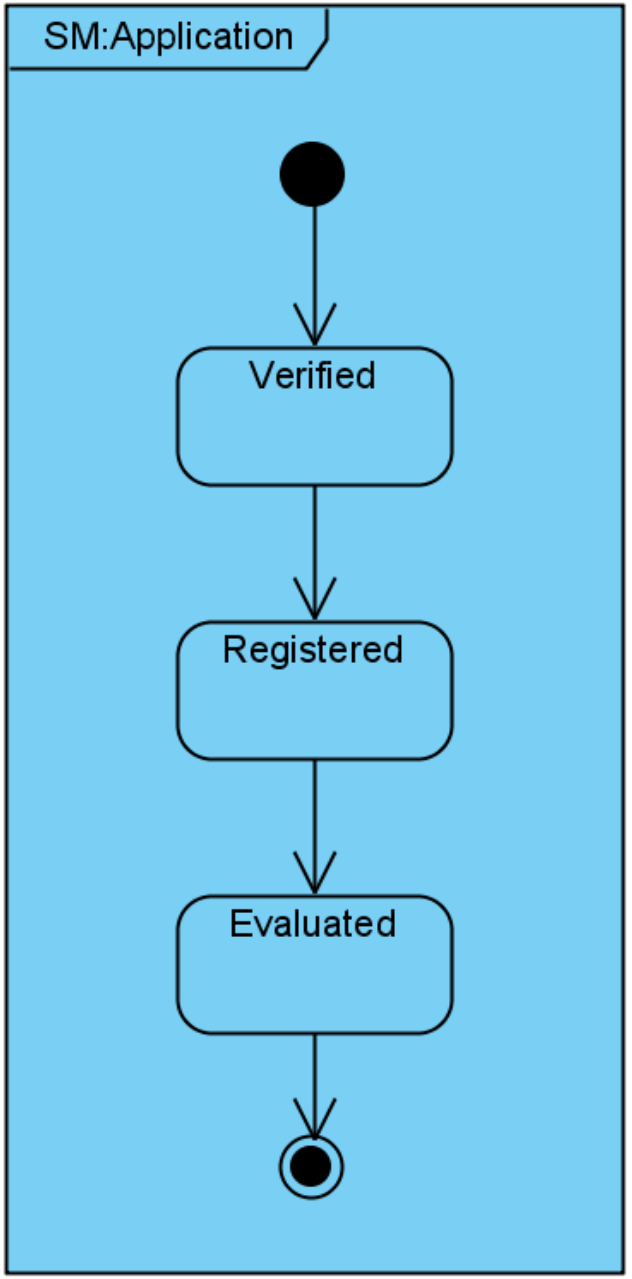
Diagram maszyny stanów (state machine)

- Opisuje przejścia między skończona liczbą stanów systemu
- Operuje pojęciem stanu obiektu
- Może być wykorzystany do generacji kodu
- Wykorzystuje elementy podobne jak w diagramie czynności
- Główne elementy to:
  - stan
  - przejście
  - stan początkowy i końcowy

Diagram maszyny stanów

- Stan opisuje stan obiektu, np. wykonywanie określonej czynności, oczekiwanie na zdarzenie
- Przejście oznacza, że obiekt będący w jednym stanie wykona pewne czynności i przejdzie do innego stanu w momencie spełnienia określonych warunków

Diagram maszyny stanów



Elementy stanu

- Stan może zostać podzielony na sekcje:
  - nazwa
  - czynności wewnętrzne
  - przejścia wewnętrzne
- Sekcje są rozdzielone poziomymi liniami

Nazwa stanu

- Powinna w jednoznaczny sposób definiować stan obiektu
- Jest to różnica w stosunku do diagramu czynności, gdzie nazwa była rozkazem wykonania jakiejś czynności

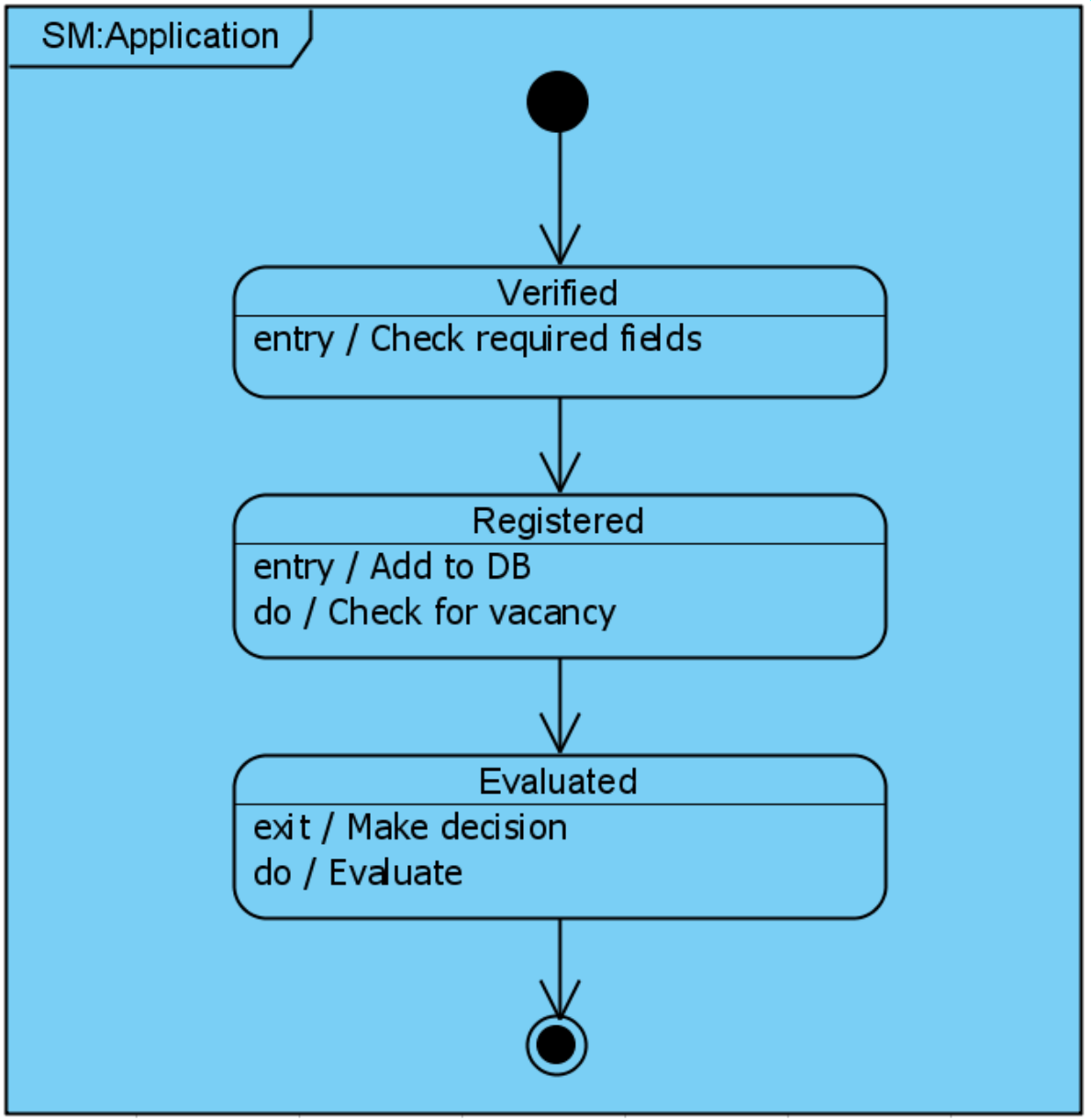
Czynności wewnętrzne

- Określają czynności wykonywane w powiązaniu ze stanem. Są trzy możliwości:
  - entry - czynność wykonywana gdy obiekt wchodzi w określony stan
  - do - czynność wykonywana w czasie gdy obiekt jest w określonym stanie
  - exit - czynność wykonywana gdy obiekt opuszcza określony stan
- W sekcjach entry i exit można zdefiniować tylko po jednej czynności, w sekcji do - wiele

Przejścia wewnętrzne

- Są to przejścia zaczynające się i kończące w tym samym stanie
- Nie powodują wywołania czynności określonych w sekcjach enter i exit

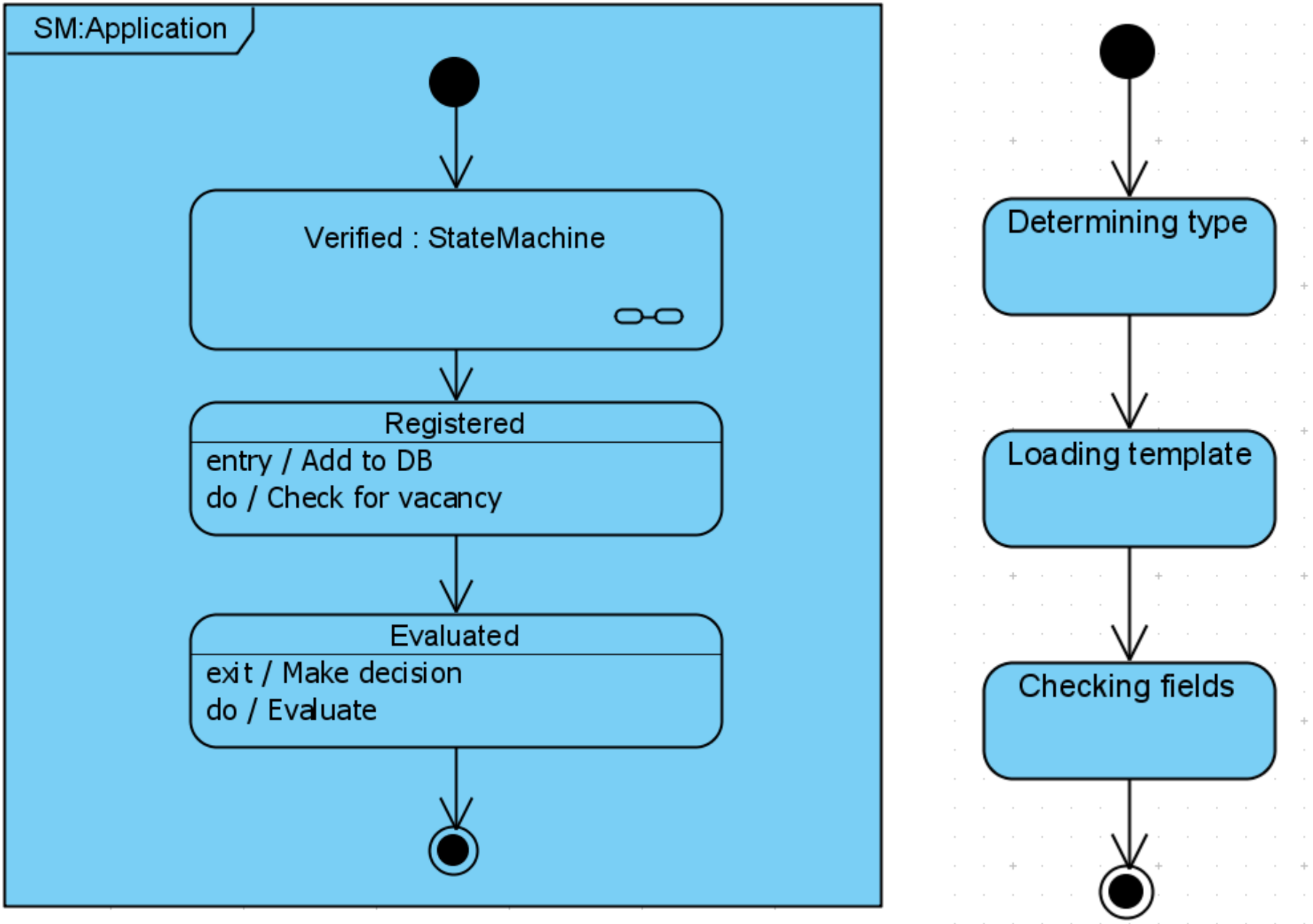
Przykładowy diagram



Stan złożony

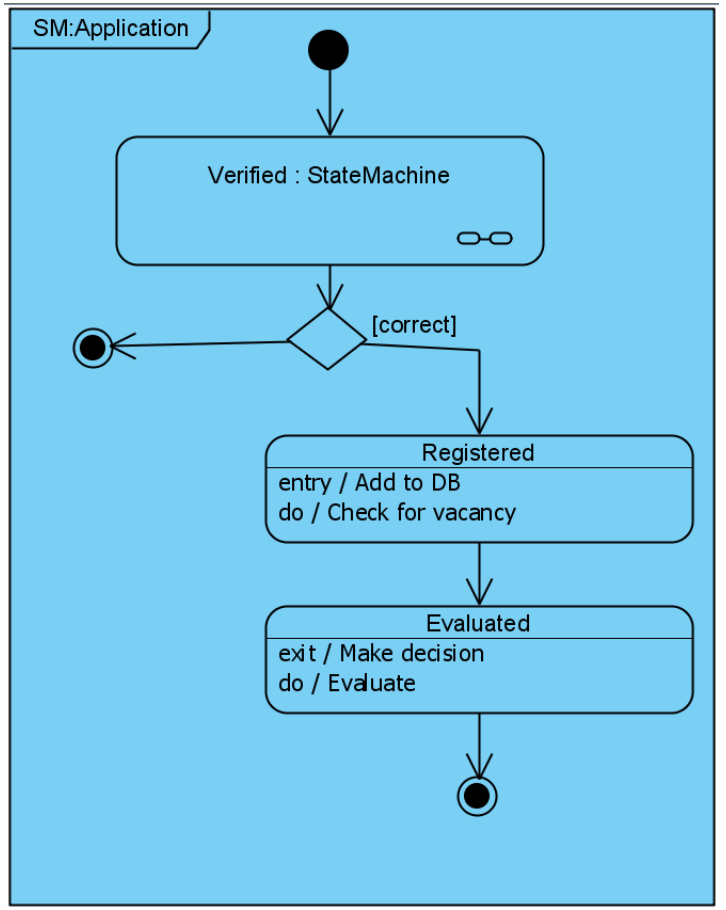
- Jest używany do przekazania większej ilości szczegółów o (złożonym) stanie
- Może zawierać podmaszynę, będącą normalnym diagramem maszyny stanów
- Stan zawierający podmaszynę oznaczony jest odpowiednim symbolem w prawym dolnym rogu

Stan złożony - podmaszyna



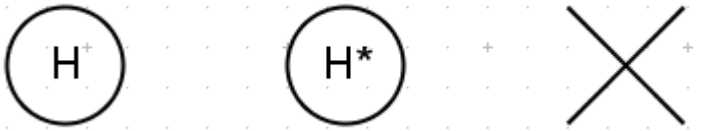
Węzły decyzji, złączenia, rozwidlenia i scalenia

- Można używać węzłów podobnych jak w diagramie czynności

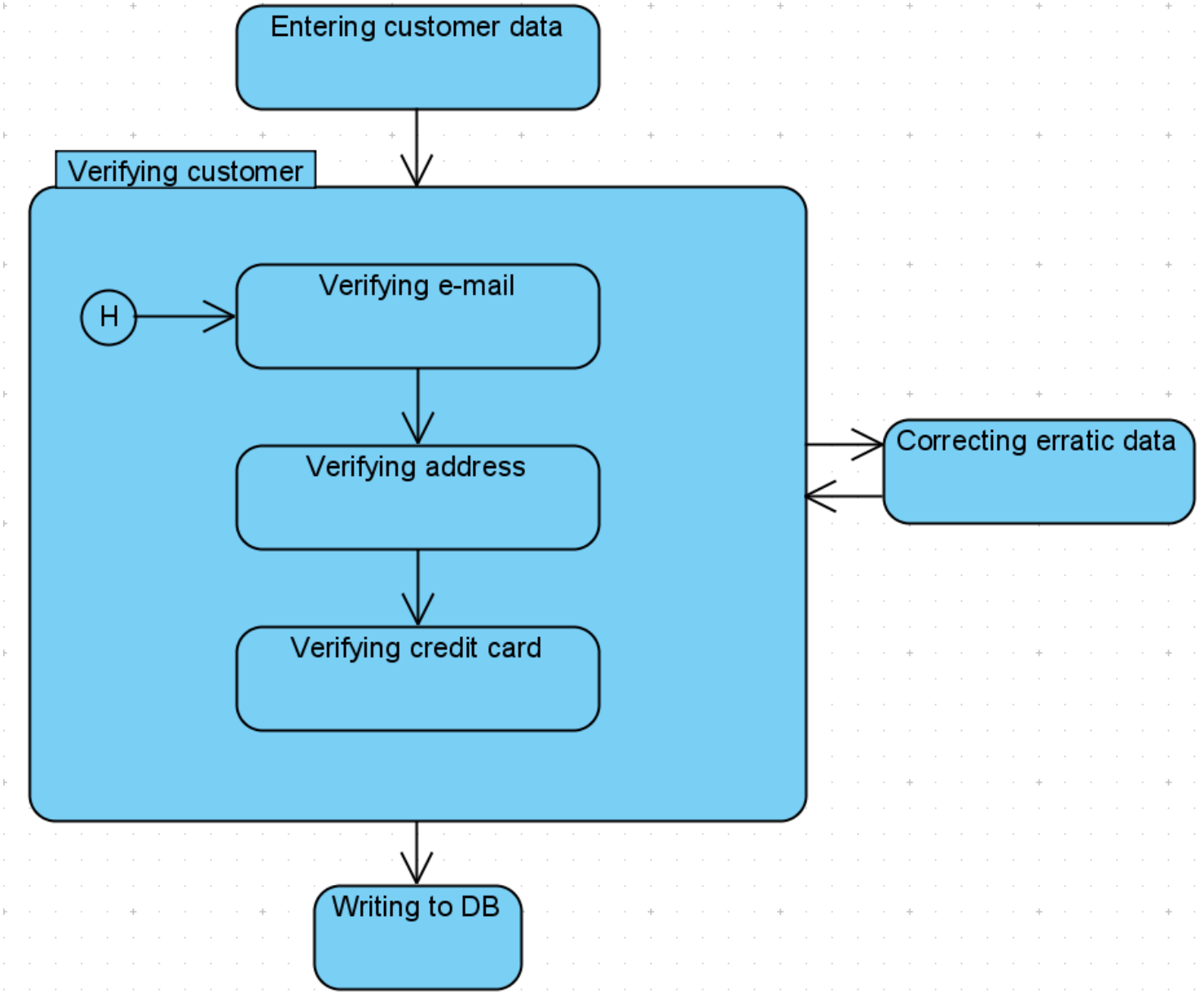


Wznowienie płytkie, głębokie, zakończenie

- Wznownienie (History) pozwala zapisać informacje o złożonym stanie w momencie opuszczenia tego stanu
  - płytkie wznownienie zapisuje wskaźnik do podstanu który był aktywny
  - głębokie zapisuje też informacje o wszystkich podstanach (czyli wskaźniki do aktywnych podstanów w podstanach itd.)
- Zakończenie oznacza koniec przetwarzania w maszynie z powodu zniszczenia przetwarzanego obiektu



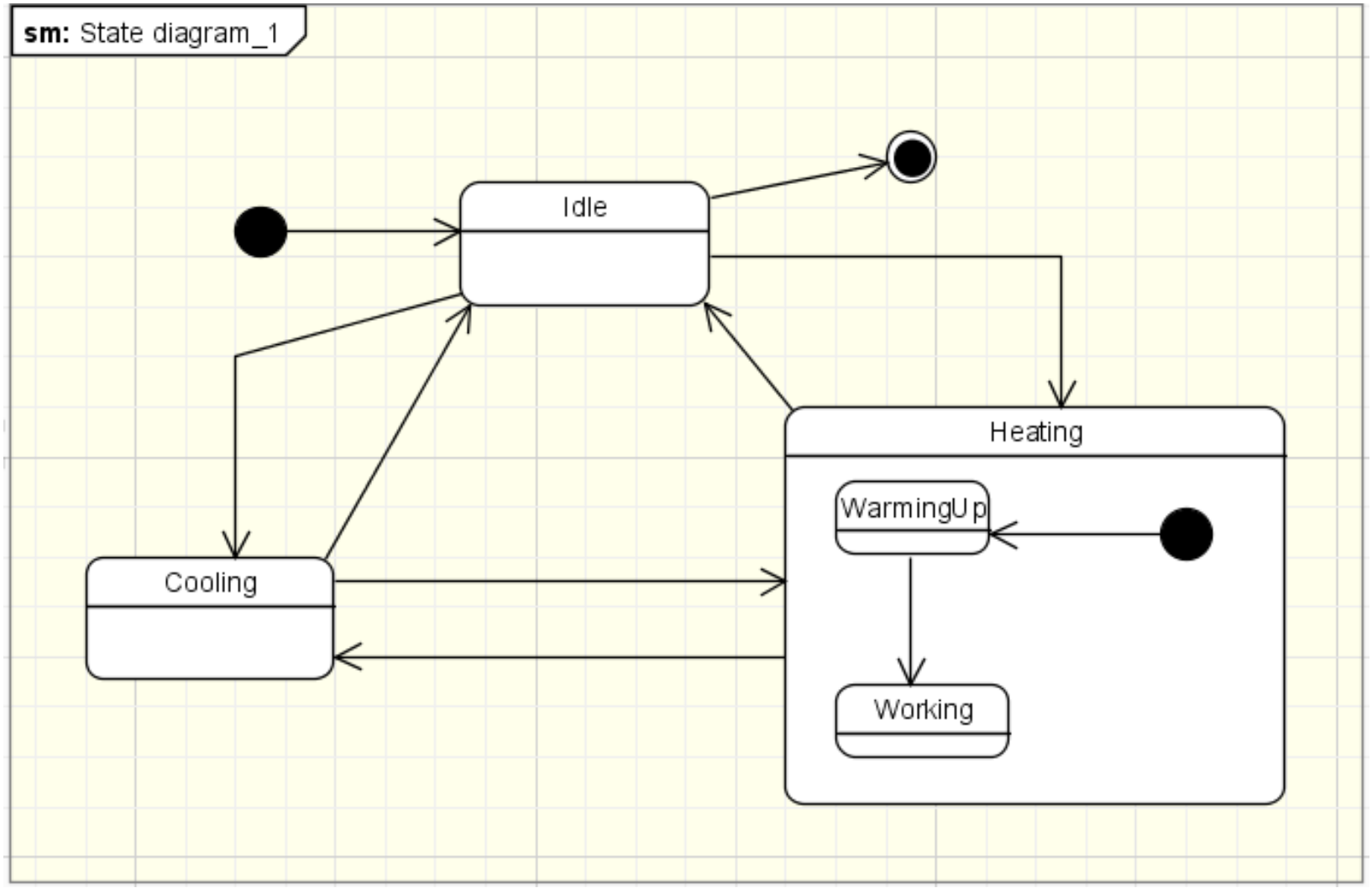
Przykład wznowienia



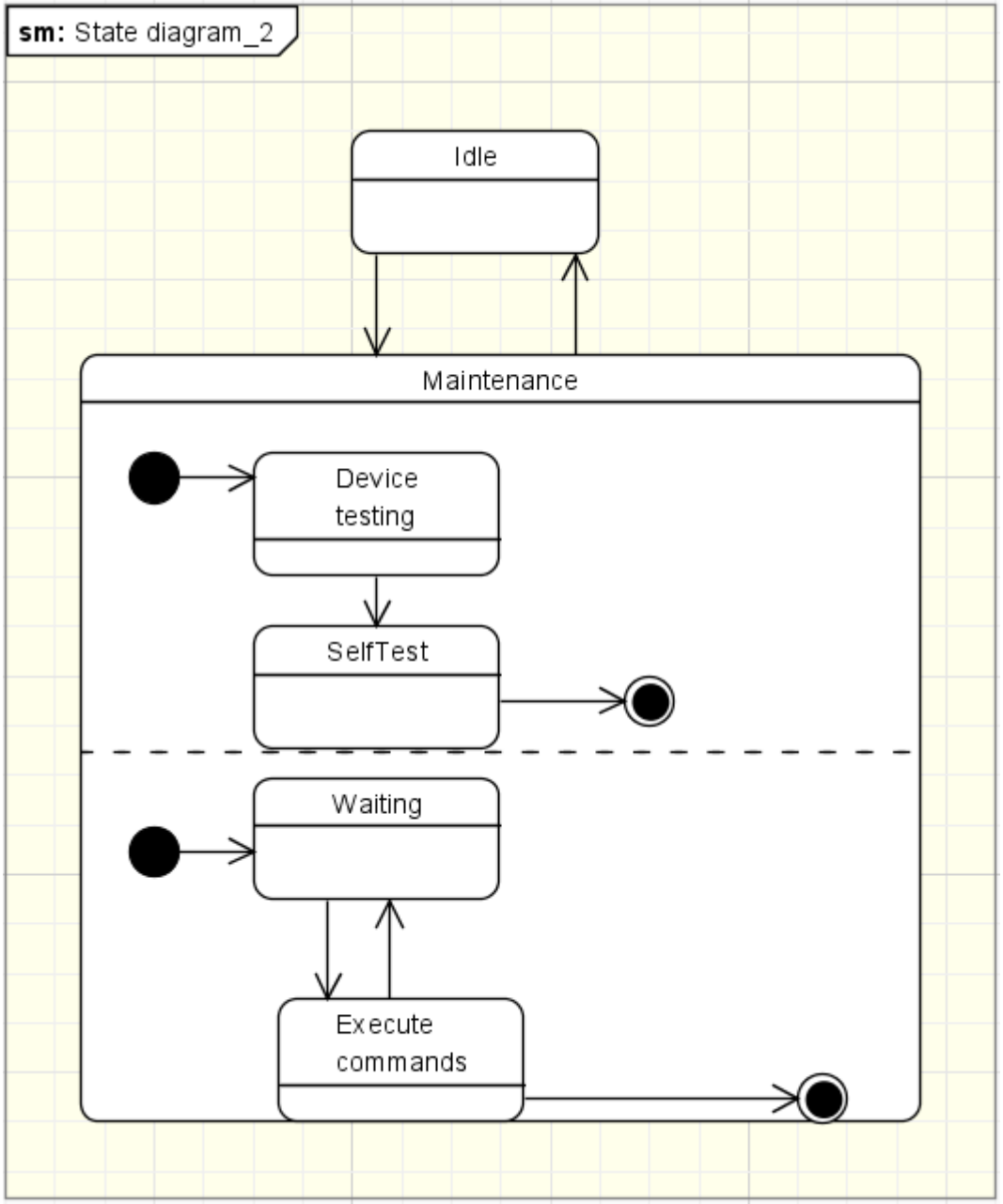
Zdarzenia

- Zdarzenia wywołują przejście z jednego stanu do innego. Możliwe zdarzenia to:
  - sygnał - asynchroniczne
  - wywołanie - podobne do wywołania funkcji
  - zdarzenie czasowe - występuje po określonym czasie; słowo kluczowe after
  - zdarzenie zmiany - występuje kiedy spełniony jest warunek; słowo kluczowe when

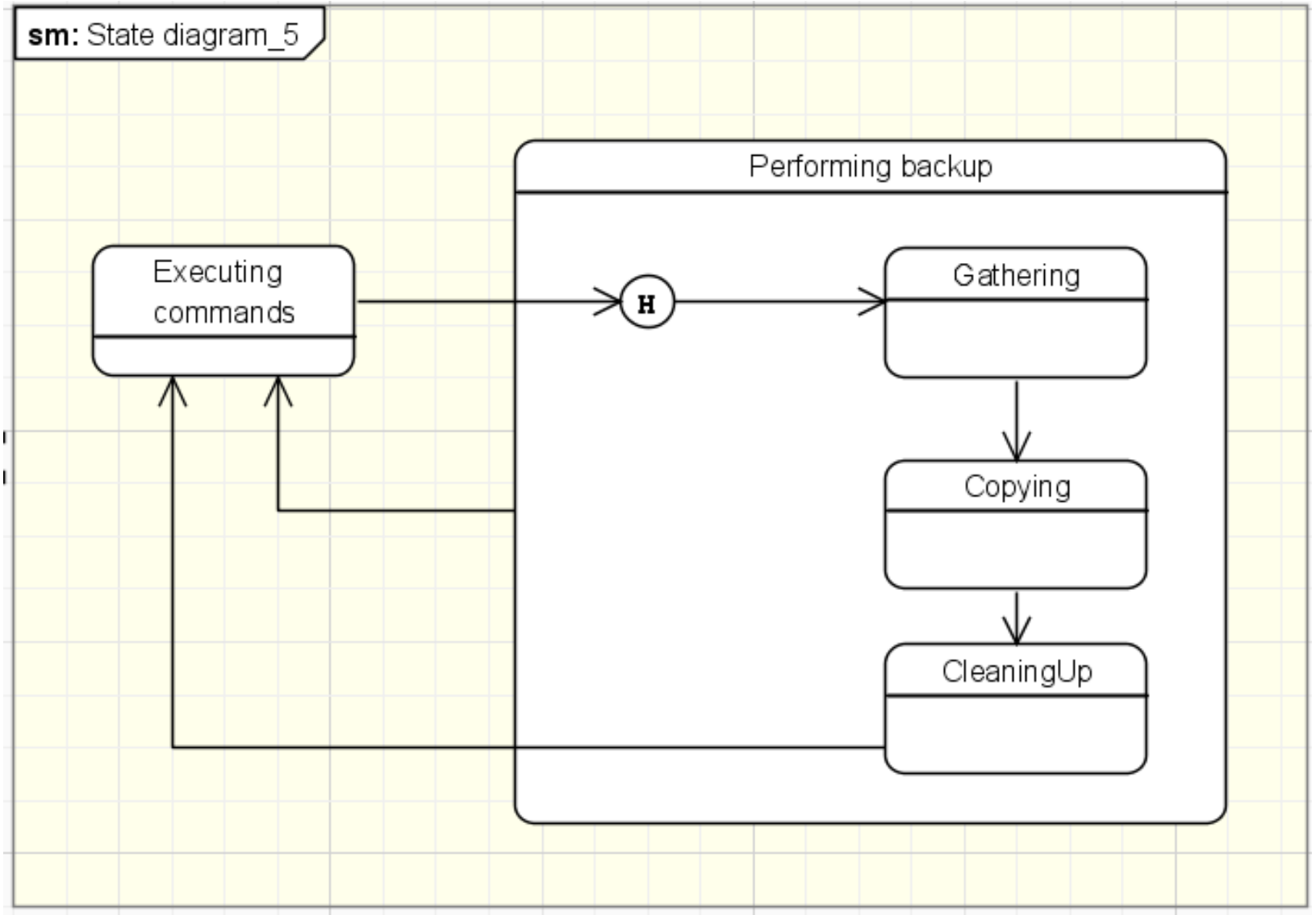
Przykładowy diagram



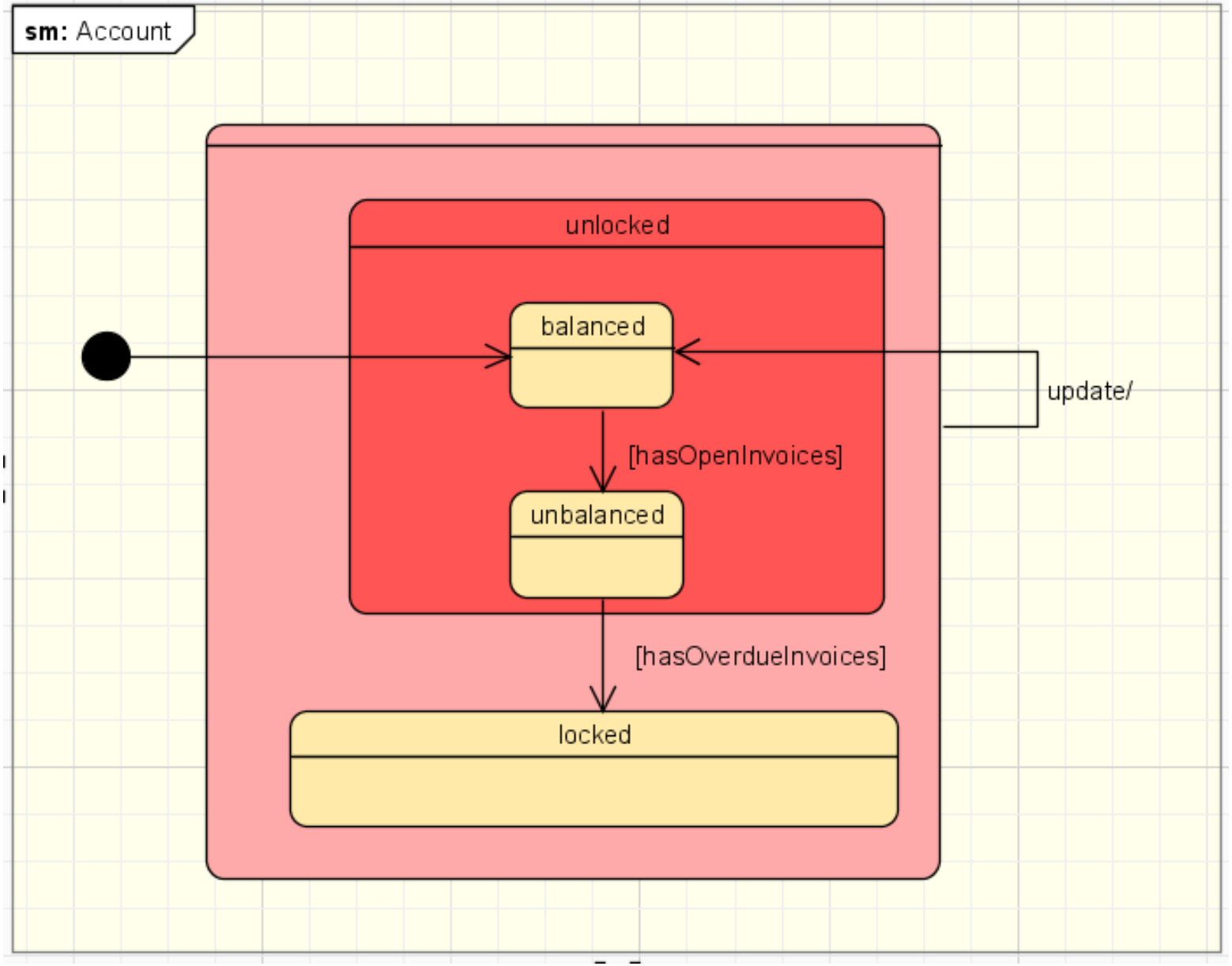
Przykładowy diagram



Przykładowy diagram



Przykładowy diagram



Przykładowy diagram

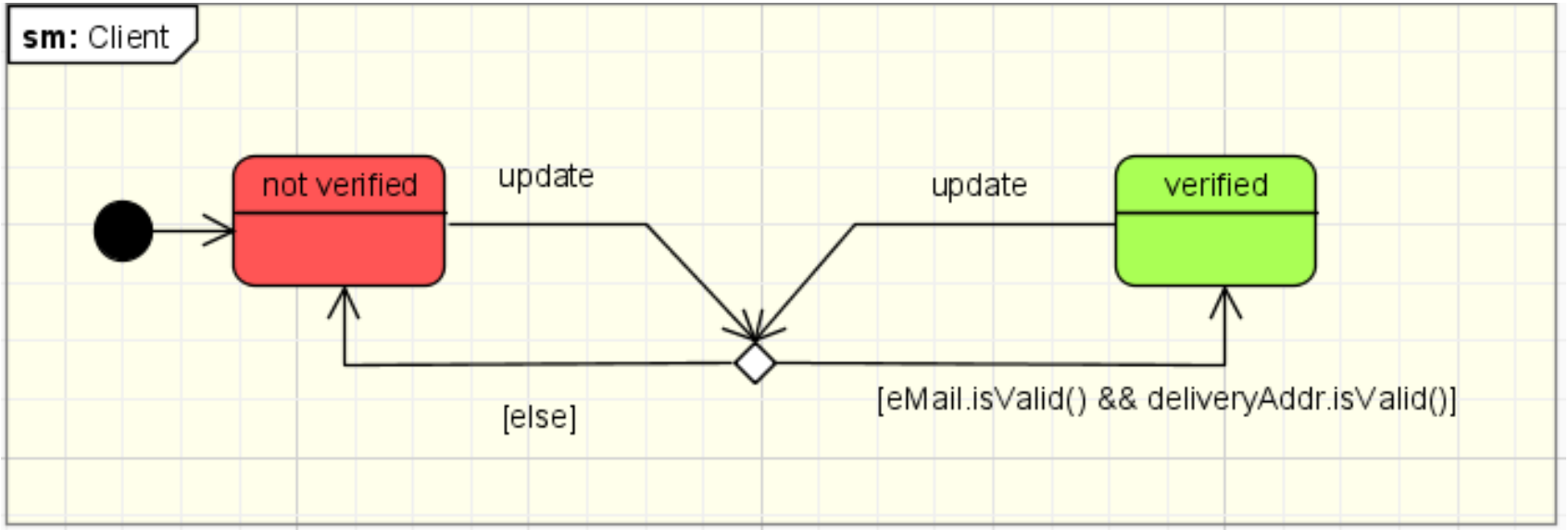


Diagram sekwencji (sequence)

- Opisuje interakcje między obiektami przy pomocy sekwencji wiadomości
- Dobrze nadaje się do dokumentowania przypadków użycia
- Diagram jest zorientowany w dwu wymiarach:
  - Oś pozioma związana jest z kolejnymi obiektami biorącymi udział w wymianie wiadomości
  - Oś pionowa związana jest z upływem czasu

Główne elementy

- Obiekt
- Linia życia
- Wiadomość
- Specyfikacja wykonania

Obiekt

- Obiekty klas są podstawowymi elementami w diagramie sekwencji
- Diagram może również zawierać instancję innych klasyfikatorów: przypadków użycia, aktorów, sygnałów itp.
- Obiekt jest przedstawiany jako prostokąt z nazwą (niekiedy podkreśloną)
- W prostych diagramach wszystkie obiekty umieszczone są przy górnej krawędzi diagramu

Linia życia

- Reprezentuje przedział czasu w którym obiekt istnieje
- Jest zobrazowana przez przerywana pionową linię, zaczynającą się na obiekcie i idącą w dół

Wiadomości

- Wiadomości reprezentują przepływ informacji między obiektami. Wiadomość jest poleceniem dla obiektu aby wykonać pewne operacje
- Kompletna składnia jest następująca:
  `predecessor/sequence_expression signature`
- Najważniejszym (i koniecznym) składnikiem opisu wiadomości jest sygnatura

Wiadomość - poprzednik

- Poprzednik to numer wiadomości która musi poprzedzać wiadomość, która ma być wykonana
- Jeśli jest więcej poprzedników, oddziela się ja przecinkami

Wiadomość - sekwencja

- Może zawierać:
  - identyfikator (liczbę, nazwę)
  - specyfikację rekurencji bądź iteracji
- Powyższe pola rozdziela się dwukropkiem
- Rekurencja jest opisana jako
  `[actualCondition]`
- Iteracja jako
  `*[iterationSpecification]`
- Przykłady: `1.2:[x>15]`

  `initial:*[i:=1..15]`

Wiadomość - sygnatura

- Może składać się z:
  - Nazwy (obowiązkowa)
  - Listy argumentów
  - Wartości zwracanej

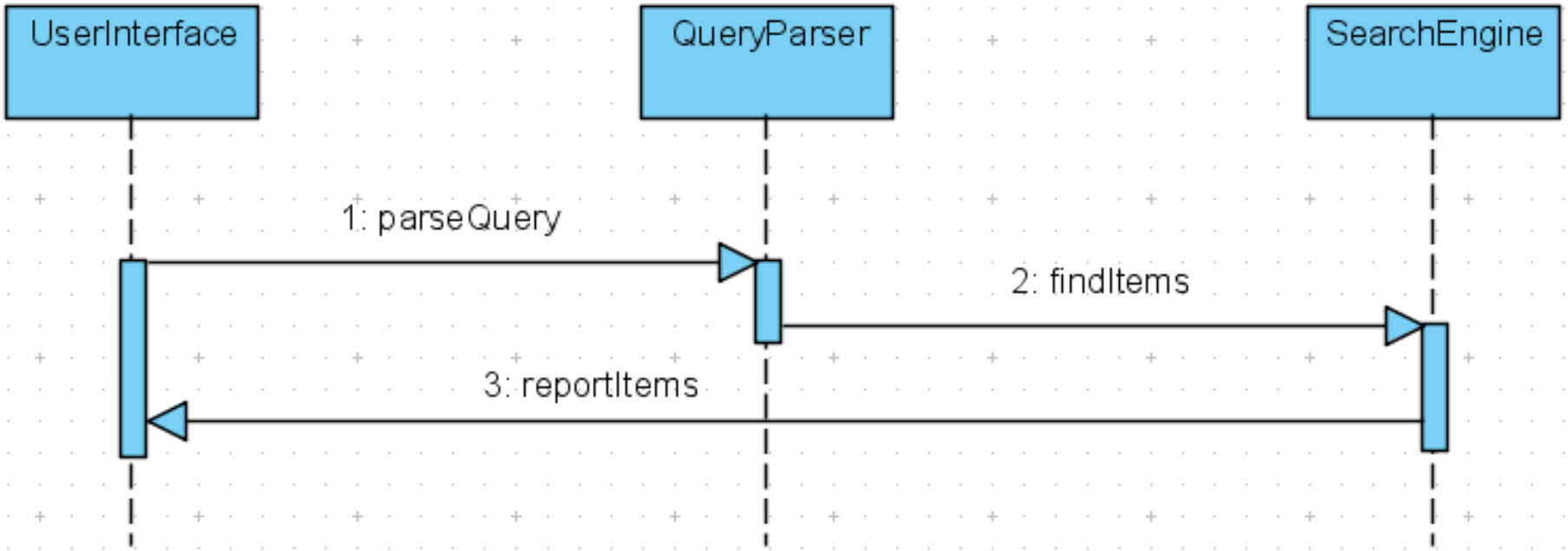
Wiadomość - sygnatura

- Nazwa wskazuje na operację która będzie wykonana przez odbiorcę wiadomości
- List argumentów i wartość zwracana mogą zostać określone analogicznie jak w specyfikacji klasy
- Wartość zwracana ma sens tylko jeśli w wyniku wykonania operacji do obiektu wywołującego przekazywana jest jakaś informacja
- Przykład:
  `findItem(name):itemsList`

Specyfikacja wykonania

- Obrazuje okres aktywności obiektu (obliczenia, przekazywanie wiadomości z/do)
- Jest przedstawiany jako prostokąt umieszczony na linii życia, jego wysokość określa okres aktywności
- Początek jest związany z aktywacją obiektu (zwykle wiadomością przekazaną od innego obiektu)

Przykładowy diagram



Rodzaje wiadomości

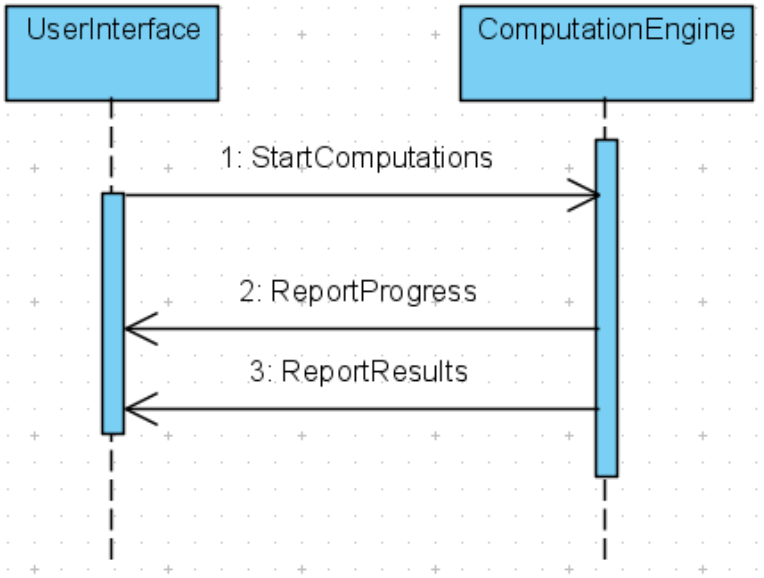
- Synchroniczna
- Asynchroniczna
- Zwrotna
- Zgubiona
- Znaleziona

Wiadomość synchroniczna

- Sterowanie jest przekazane do obiektu wywoływanego
- Przetwarzanie w obiekcie wywołującym jest wstrzymywane do momentu zakończenia wywołanej czynności
- Jest obrazowane pełną strzałką
- Jest odpowiednikiem wywołania funkcji

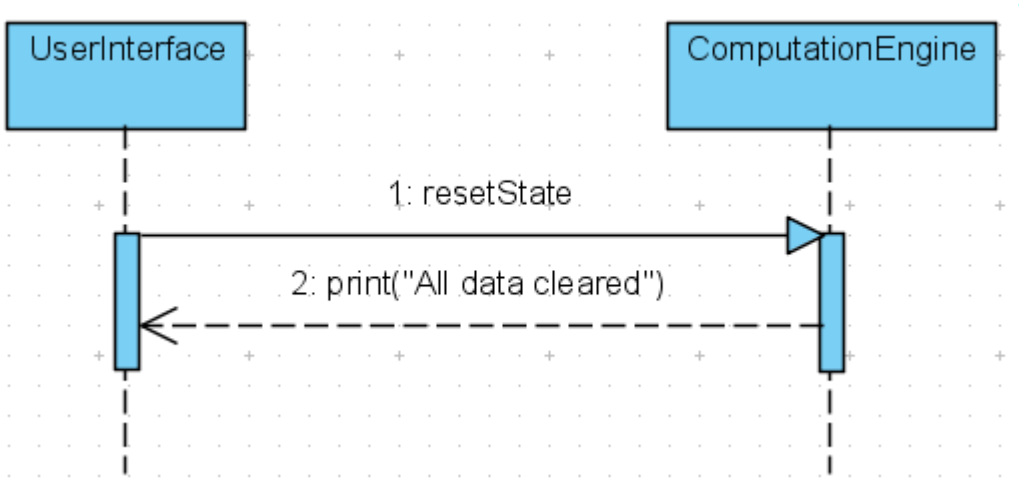
Wiadomość asynchroniczna

- Przetwarzanie w obiekcie wywołującym nie jest przerwane
- Jest obrazowane otwartą strzałką
- Możliwe jeśli wywołujący i wywoływany są w różnych wątkach (procesach itp.)



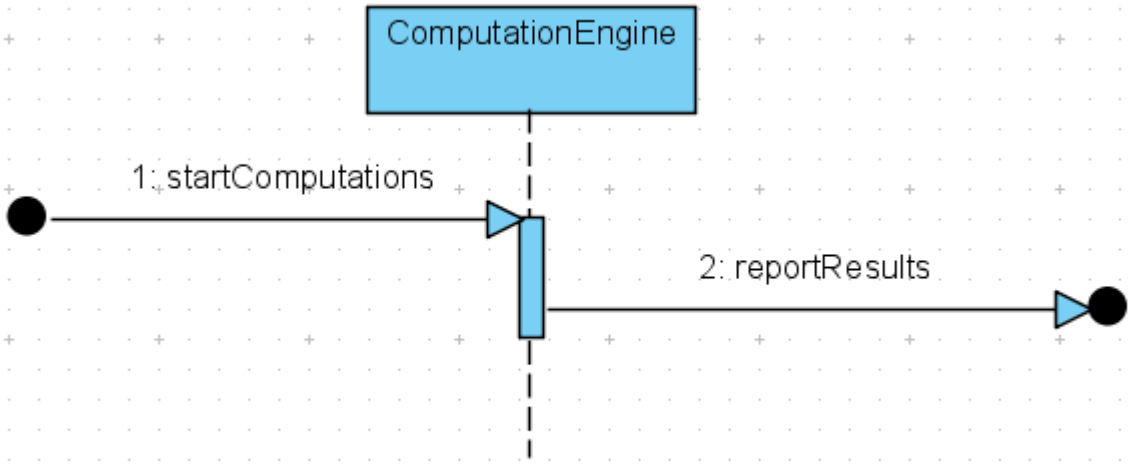
Wiadomość zwrotna

- Obrazuje oddanie sterowania do obiektu wywołującego
- Jest opcjonalna
- Jest przedstawiona jako strzałka z linią przerywaną



Wiadomości zgubione i znalezione

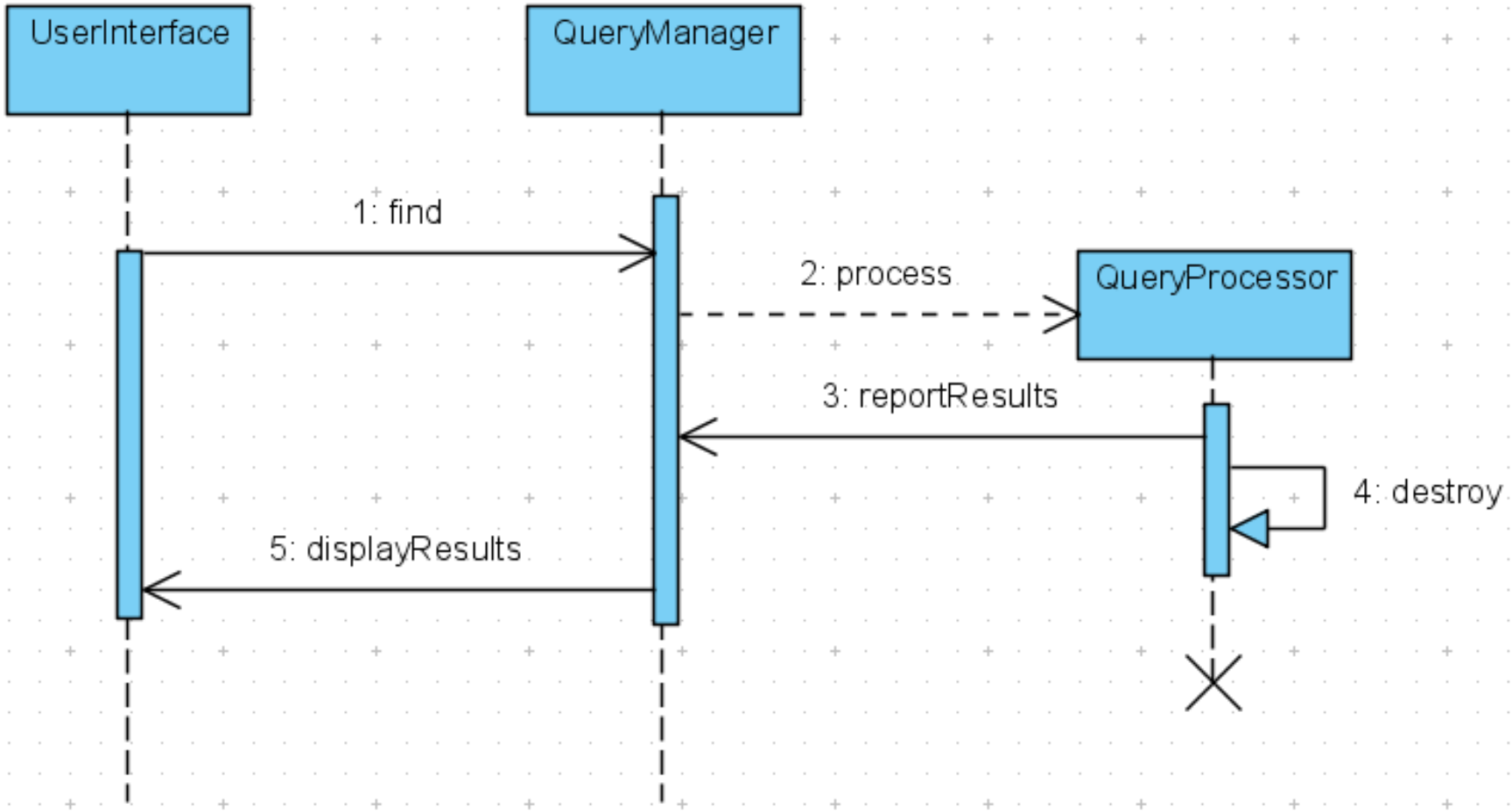
- Przydatne jeśli obiekt wywołujący (wiadomość znaleziona) lub wywoływany (zgubiona) nie są znane podczas tworzenia diagramu
- Sytuacja taka jest typowa w dużych systemach
- Graficznie jest to reprezentowane przez umieszczenie czarnej kropki zamiast obiektu



Tworzenie i niszczenie obiektów

- Tworzenie i niszczenie są powodowane przez odpowiednie wiadomości
- Wiadomości te mają stereotypy, odpowiedni “create” i “destroy”
- Na końcu wiadomości “create” umieszcza się tworzony obiekt (co powoduje że znajduje się on poniżej innych obiektów)
- Po otrzymaniu wiadomości “destroy” linia życia obiektu kończy się. Jest to dodatkowo wyróżnione umieszczeniem znaku X na końcu linii.

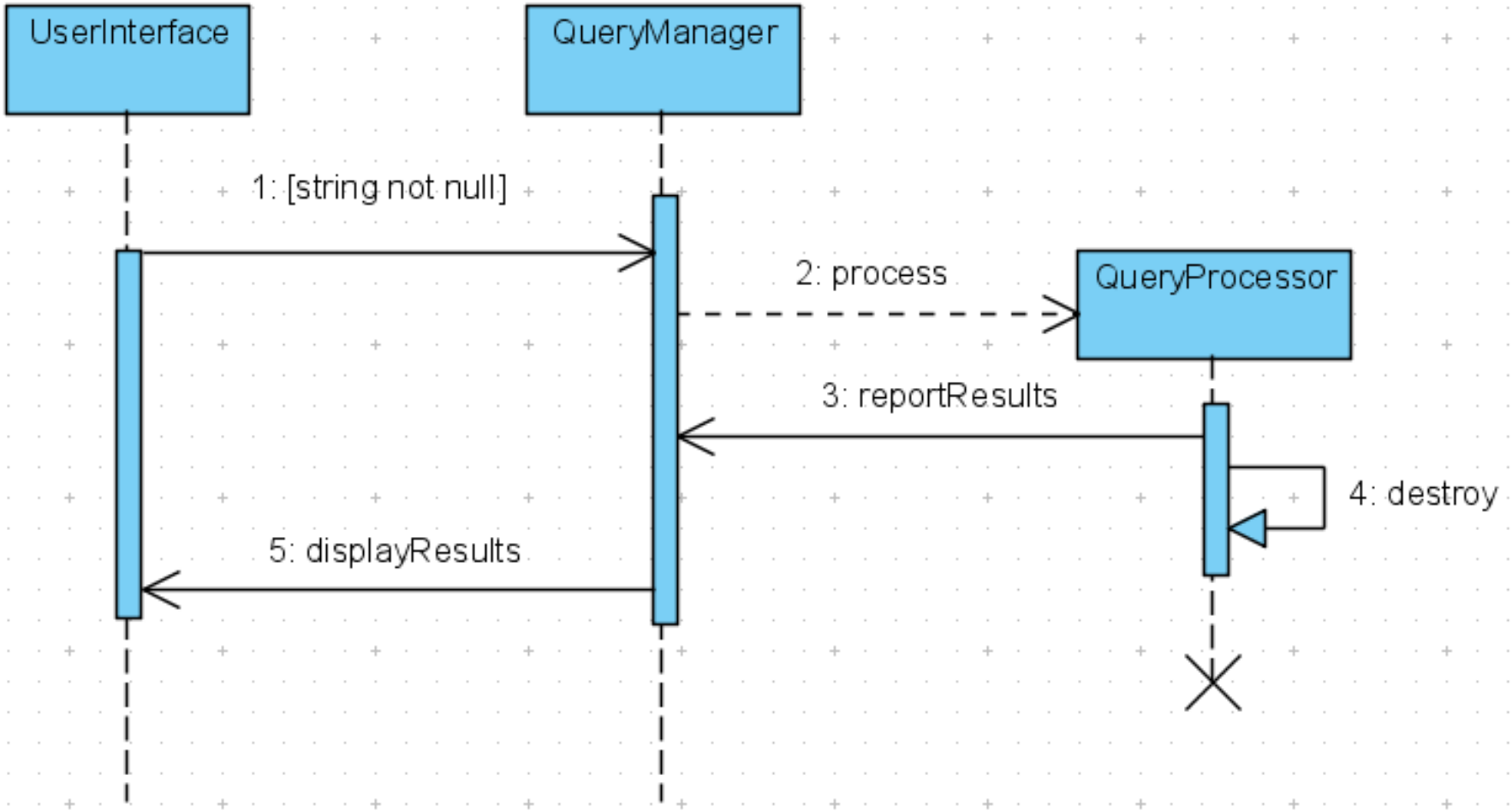
Przykładowe tworzenie i niszczenie



Wiadomości warunkowe

- Można określić warunek który warunkuje przekazanie wiadomości
- Są umieszczane w kwadratowych nawiasach przed specyfikacją wiadomości
- Można określić więcej niż jeden warunek
- Jeśli warunek nie jest spełniony, wiadomość nie zostanie przekazana

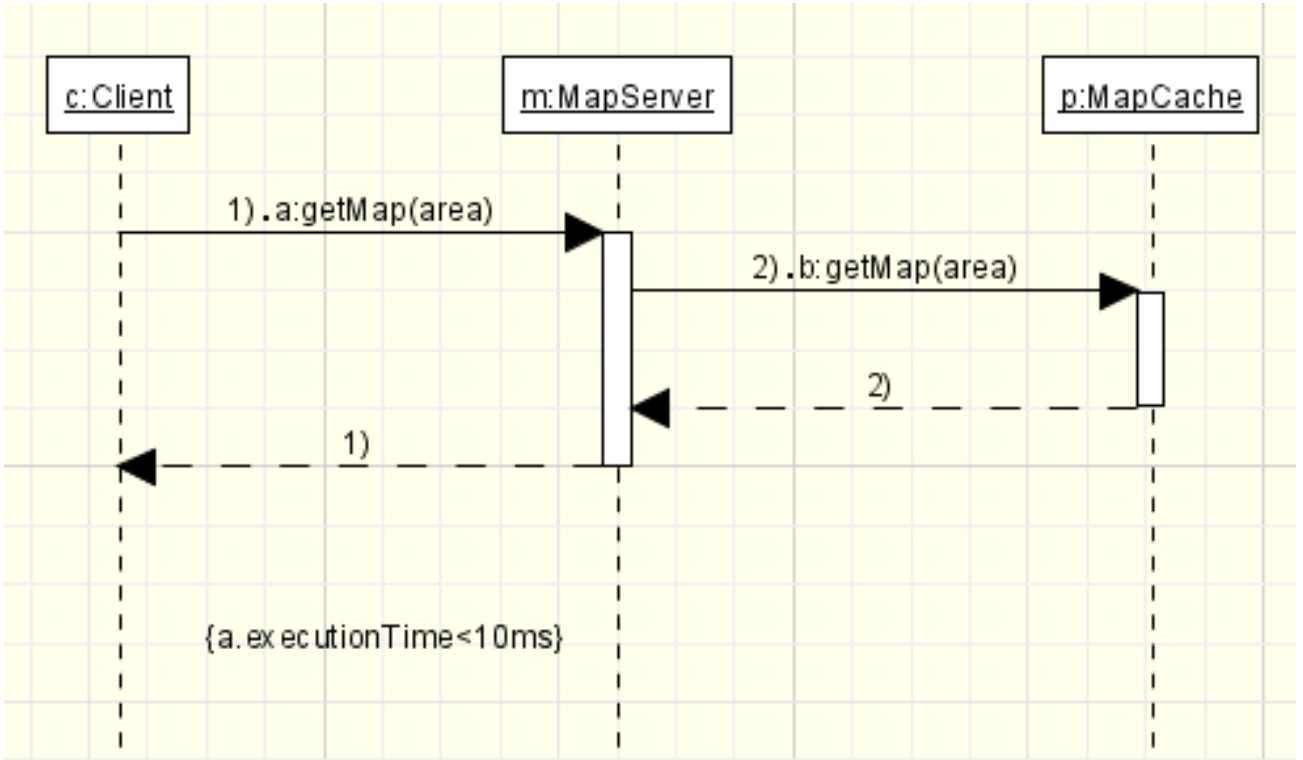
Przykładowa wiadomość warunkowa



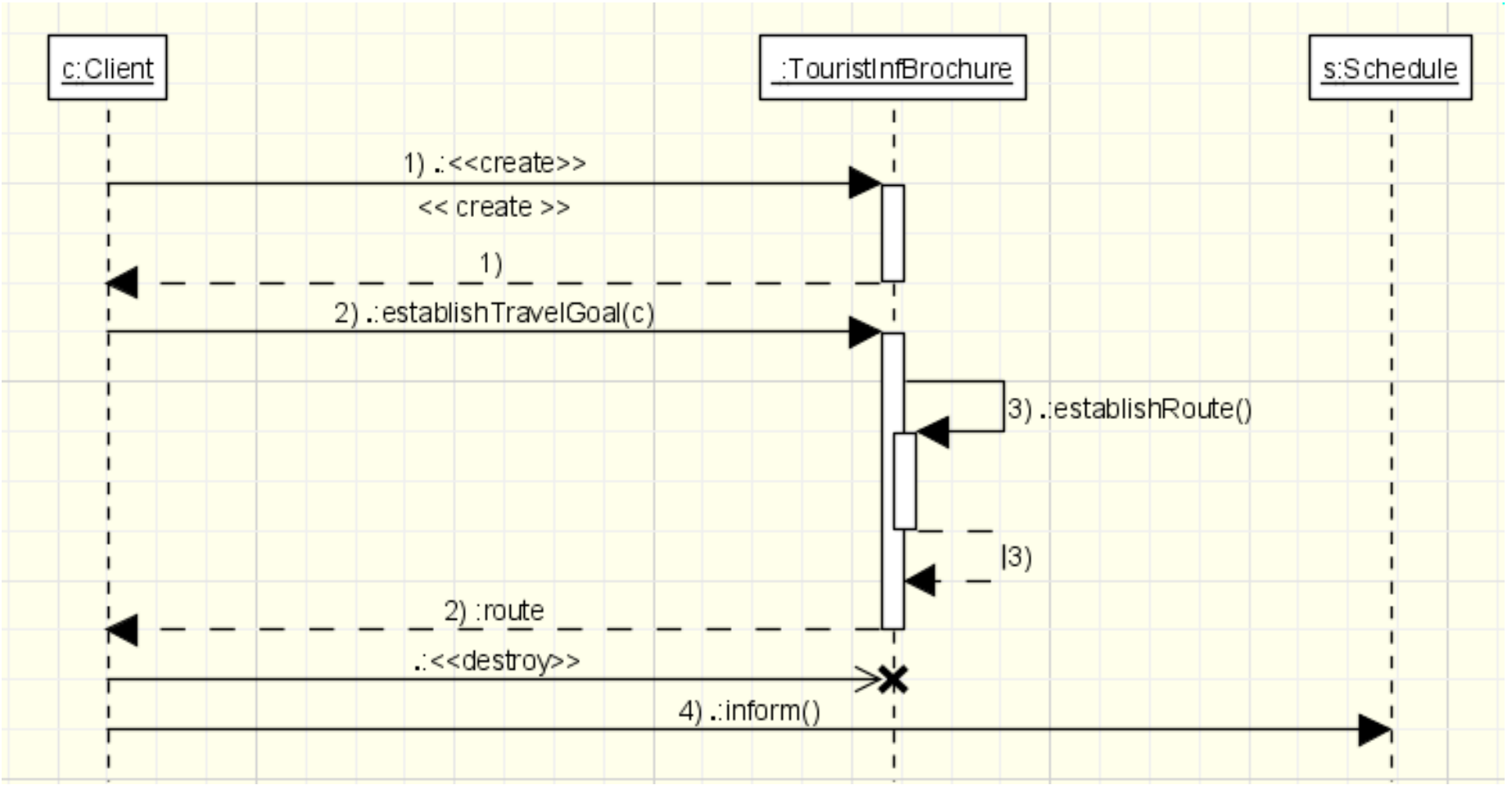
Rozgałęzienie

- Dwie lub więcej wiadomości może zostać przekazanych, w zależności od warunku
- Wiadomości alternatywne mogą zostać przekazane do tego samego obiektu, bądź różnych obiektów

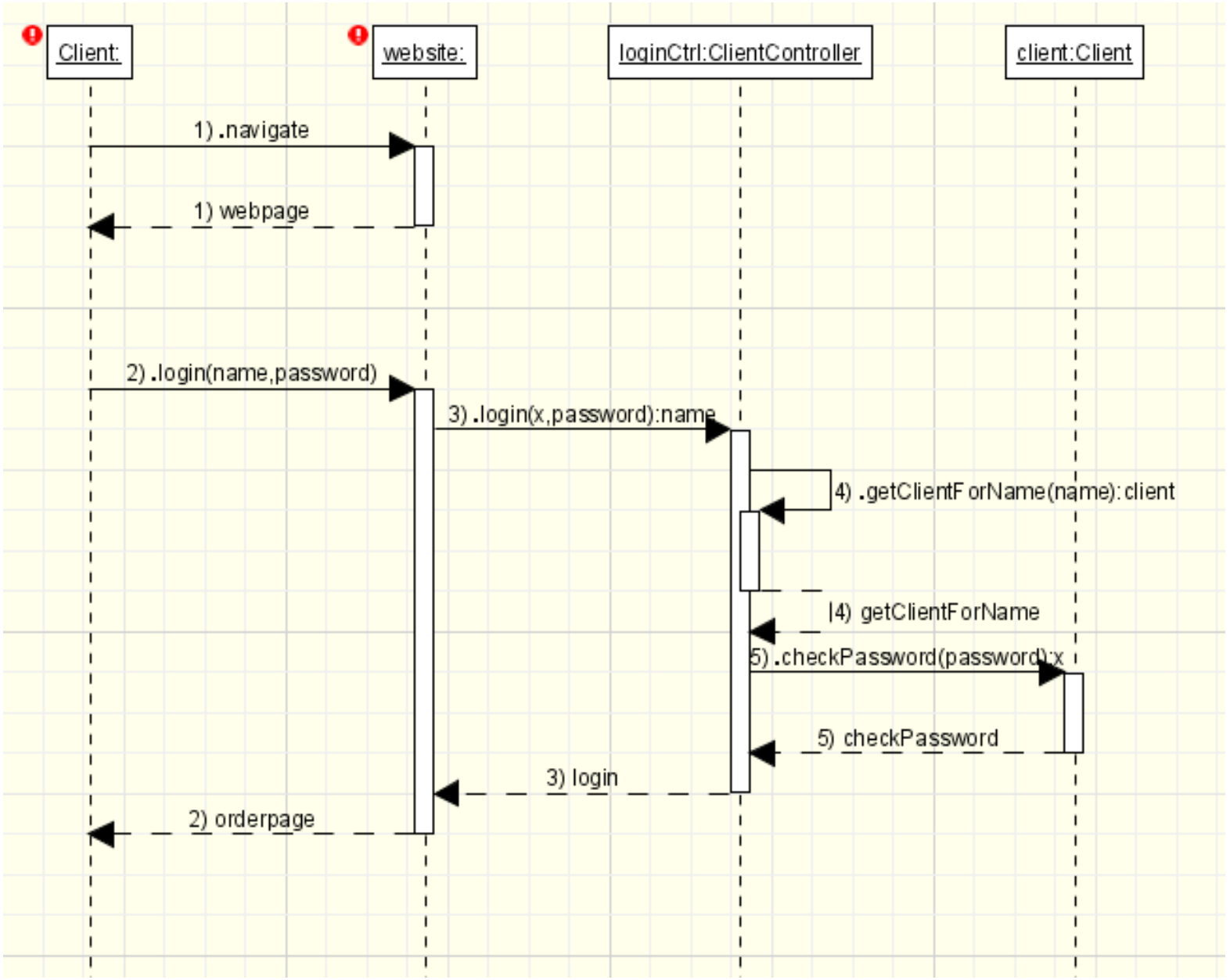
Przykładowy diagram



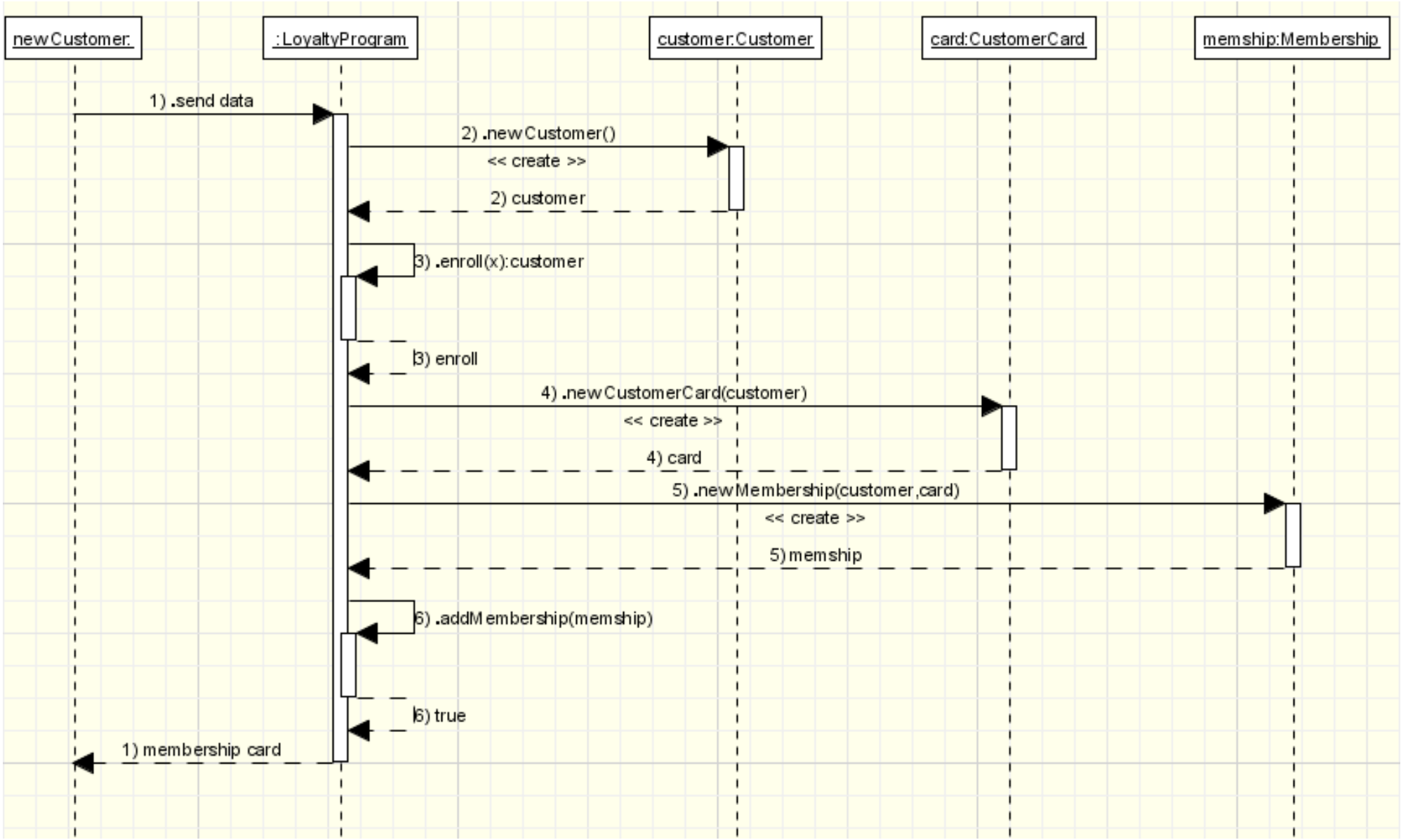
Przykładowy diagram



Przykładowy diagram



Przykładowy diagram



Fragmenty złożone

- Są to wybrane fragmenty diagramu sekwencji, do których odnosi się odpowiedni operator interakcji
- Są zobrazowane ramą otaczającą wybrany region. Rama ma nagłówek w lewym górnym rogu zawierający operator interakcji
- Niektóre operatory wymagają wyodrębnienia podfragmetów regionu. Są one wyodrębniane linią kropkowo-kreskową

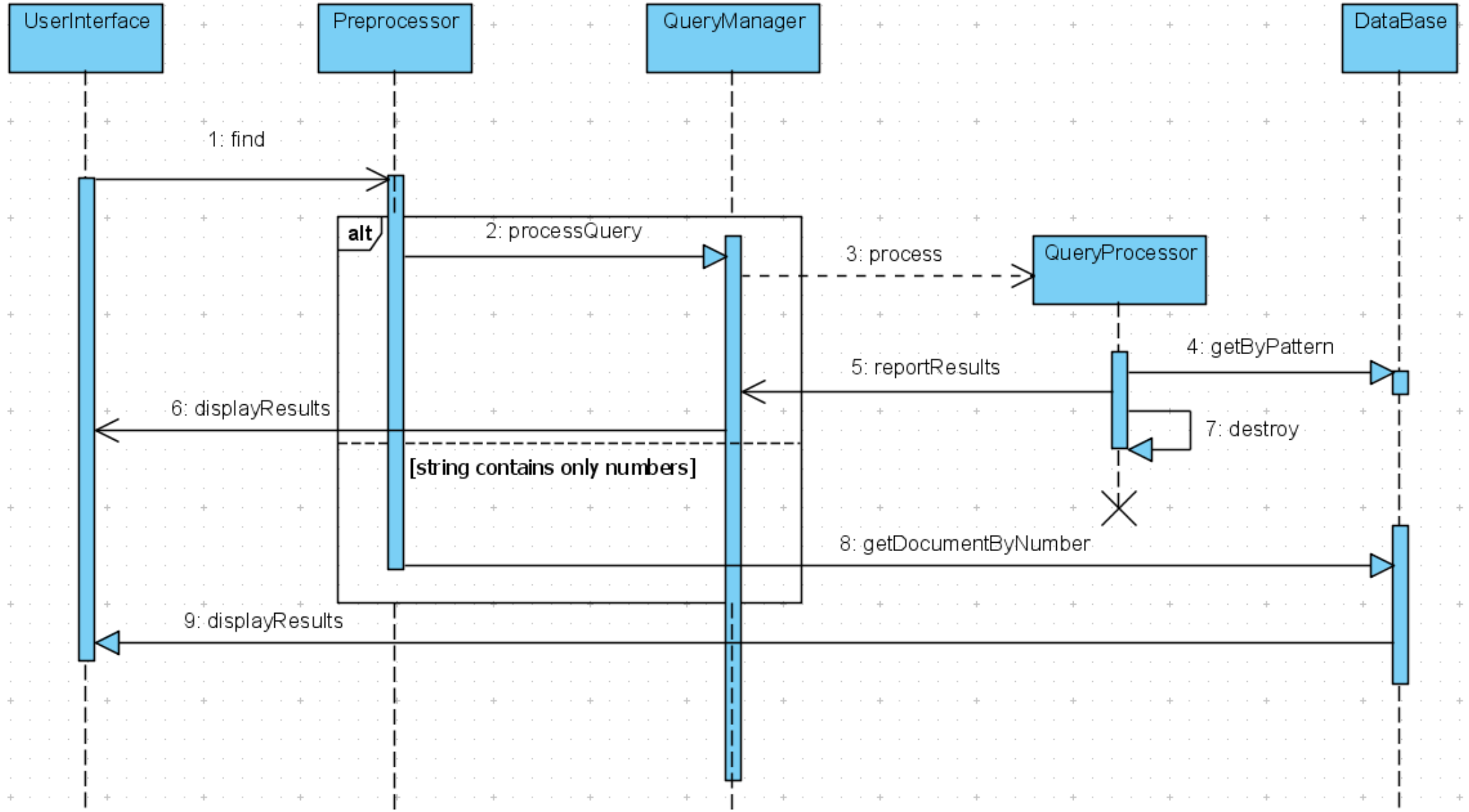
Wybrane operatory

- Alt
- Opt
- Break
- Loop
- Neg
- Par
- Critical
- Assert
- Consider
- Ignore

Operator alt - alternatywa

- Oznacza, że tylko jeden z podobszarów (operandów) obszaru objętego ramą może być wybrany
- Wybór ten zależy od od warunków umieszczonych w podfragmentach
- Podobszar bez warunku jest wyborem domyślnym
- Może być używany zamiast rozgałęzienia

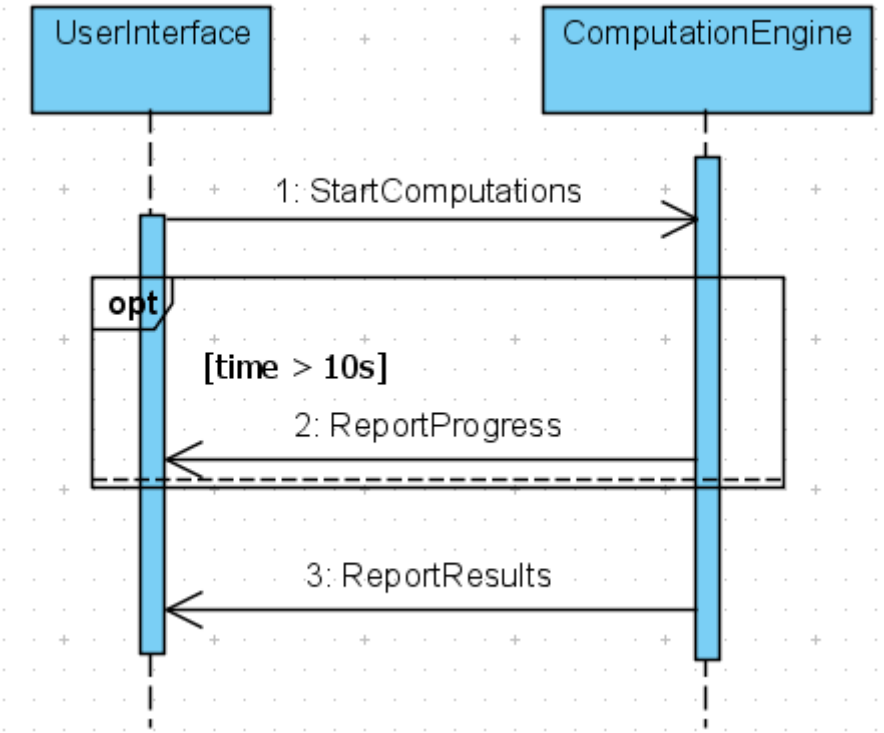
Przykład operatora alt



Operator opt – fragment opcjonalny

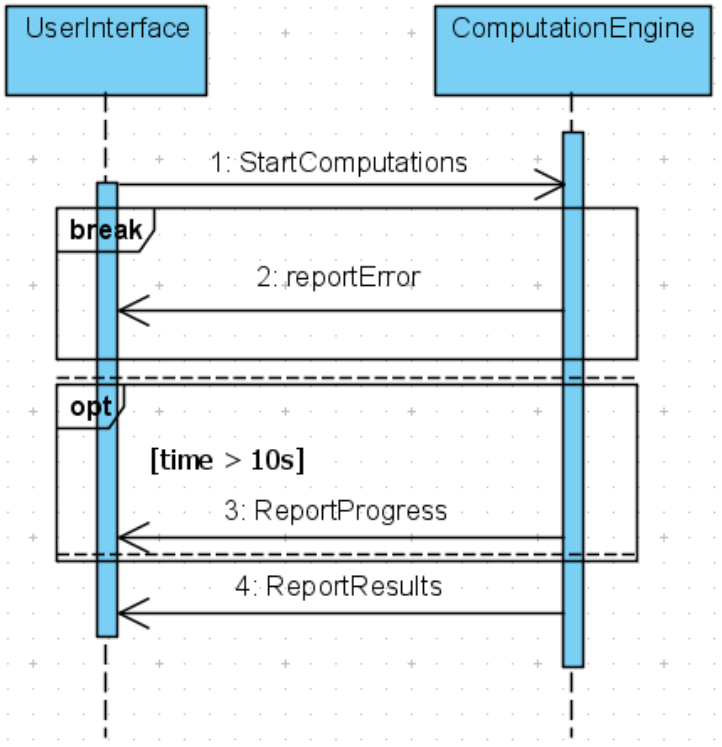
- Część diagramu zostanie wykonana tylko jeśli spełniony będzie warunek
- Odpowiada operatorowi alt z pustym domyślnym operandem
- Może być użyty zamiast wiadomości warunkowej

Przykład operatora opt



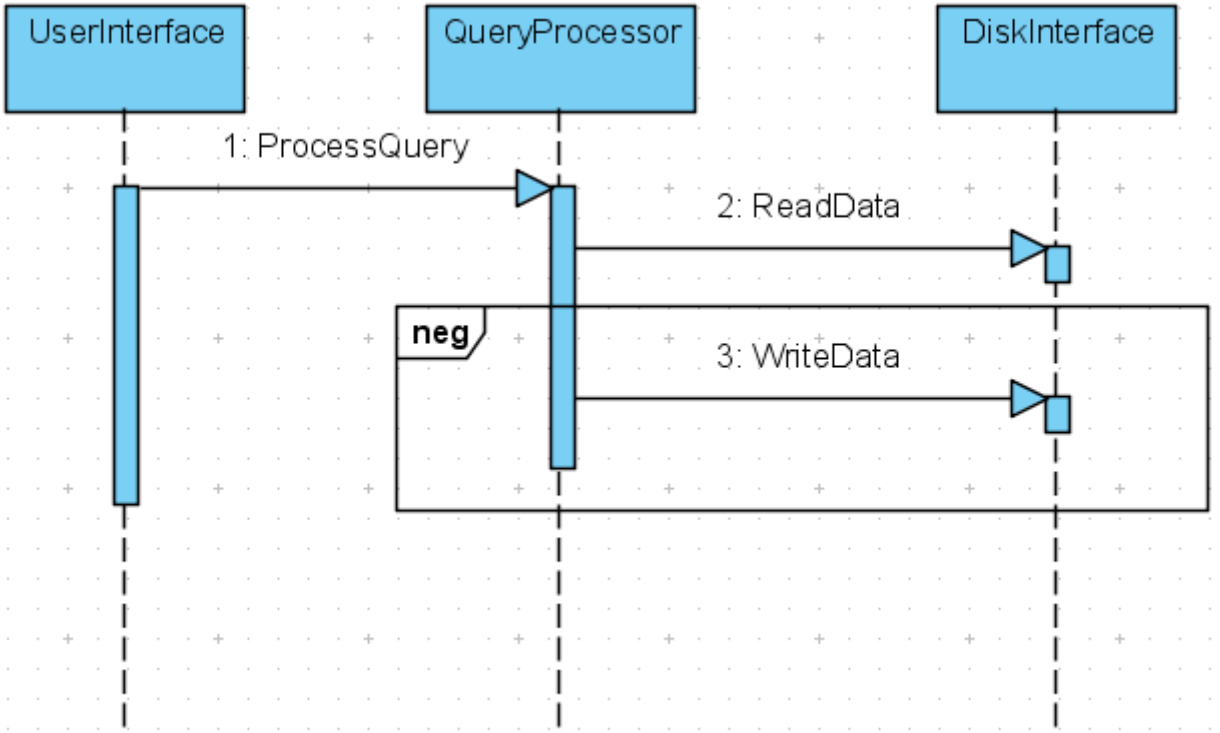
Operator break - przerwanie wykonania

- Pozwala zdefiniować fragment wykonany przy spełnieniu warunku
- Jeśli fragment jest wykonany, reszta specyfikacji wykonania jest pomijana



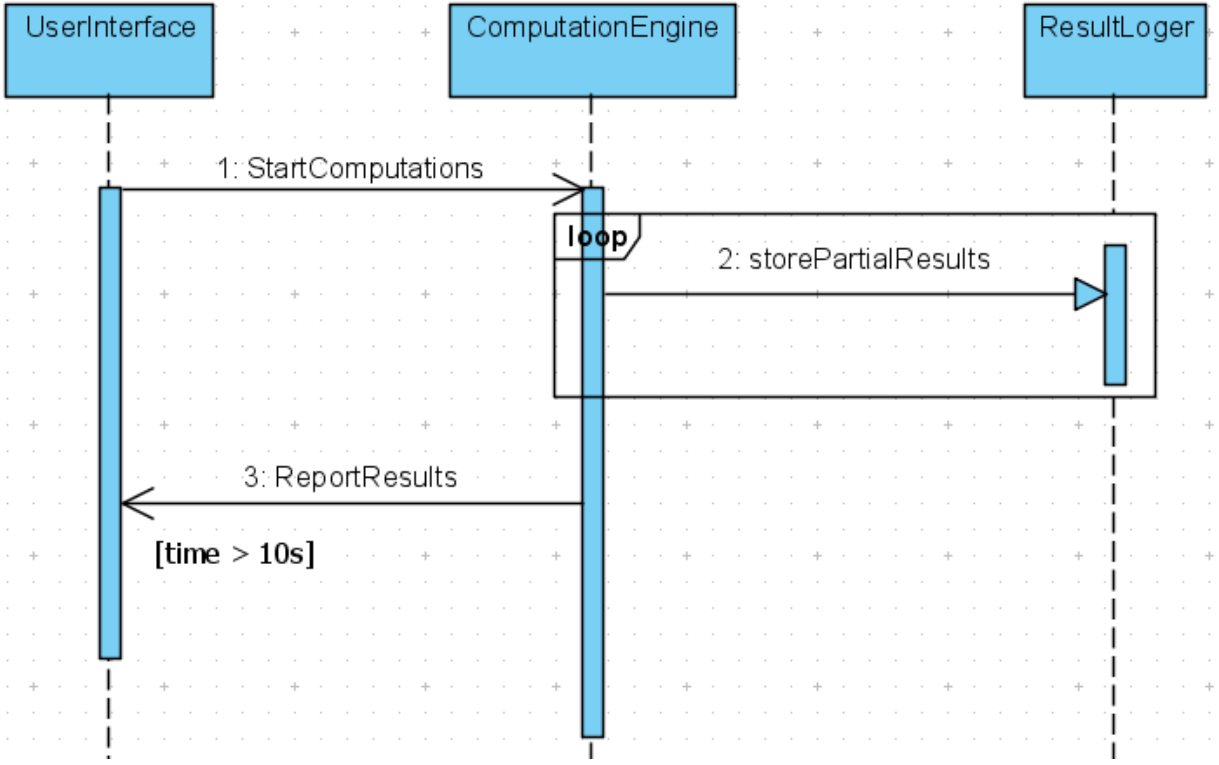
Operator neg - błędne zachowanie

- Wskazuje na fragment, który nie powinien być wykonany (jeśli jest wykonany, jest to błędne zachowanie)



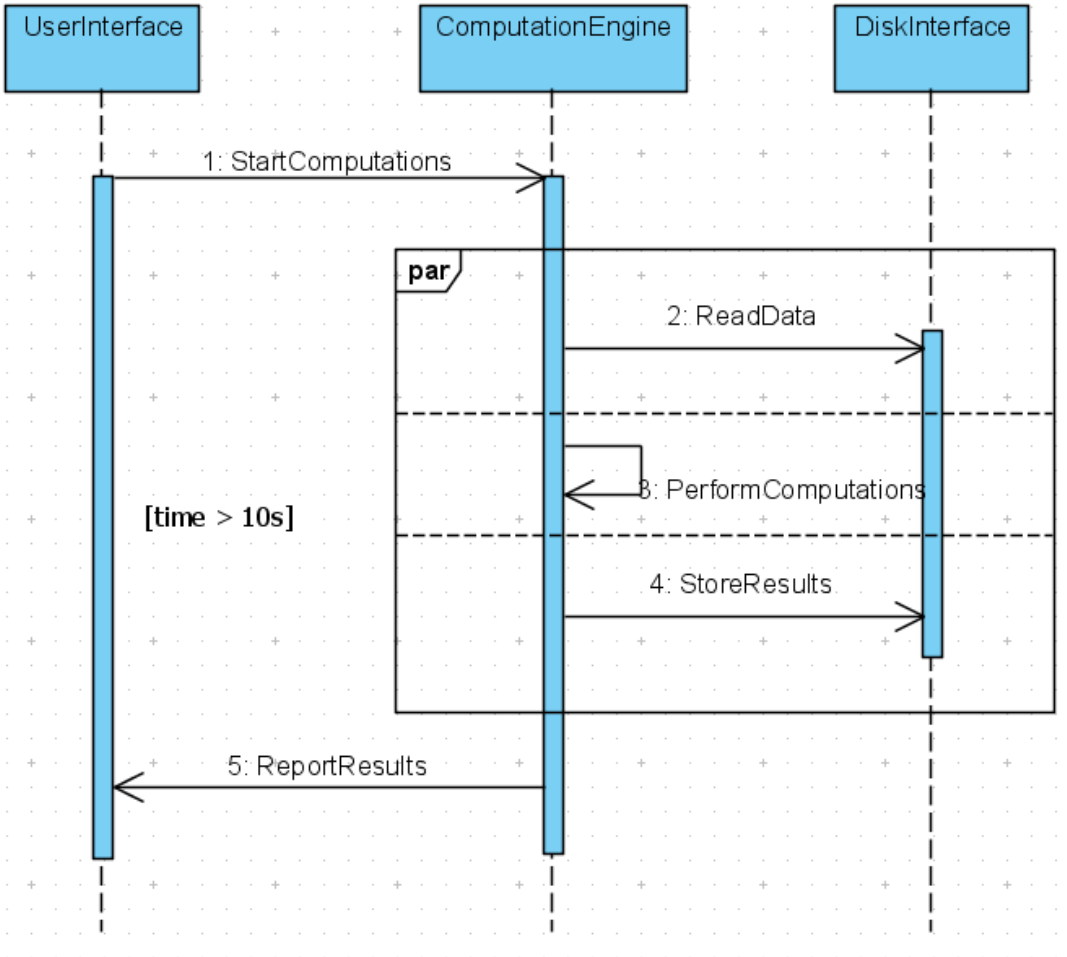
Operator loop - iteracja

- Umożliwia wielokrotne powtórzenie wybranego fragmentu
- Liczba interacji moze zostać określona



Operator par – wykonanie równoległe

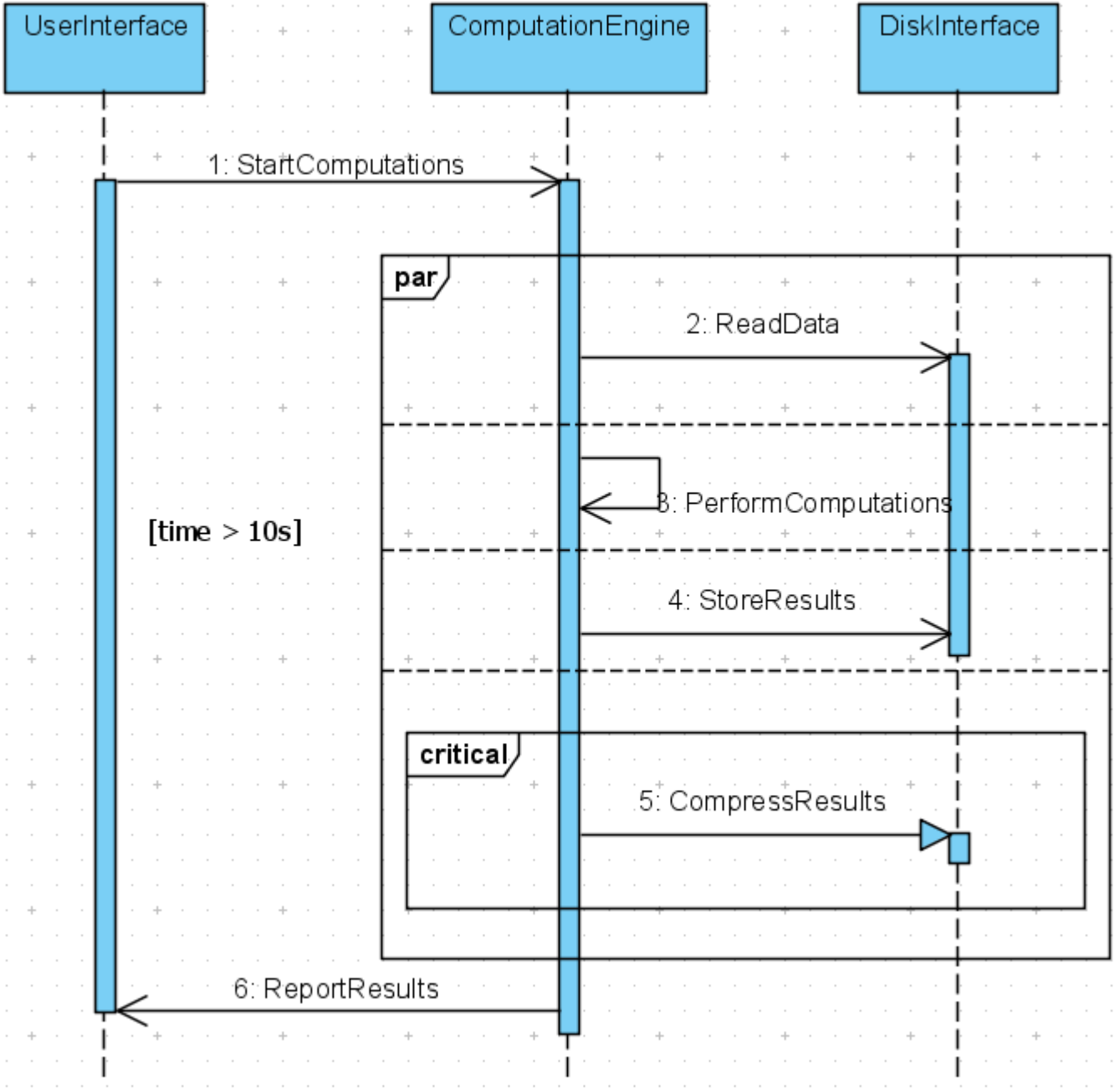
- Oznacza, że podfragmenty fragmentu objetego ramą są wykonywane równolegle



Operator critical - fragment o wysokim priorytecie

- Wskazuje fragment który, w momencie wykonania, zablokuje obiekty uczestniczące w wykonaniu, aż do zakończenia operacji
- Operacje dotyczące innych obiektów mogą być kontynuowane

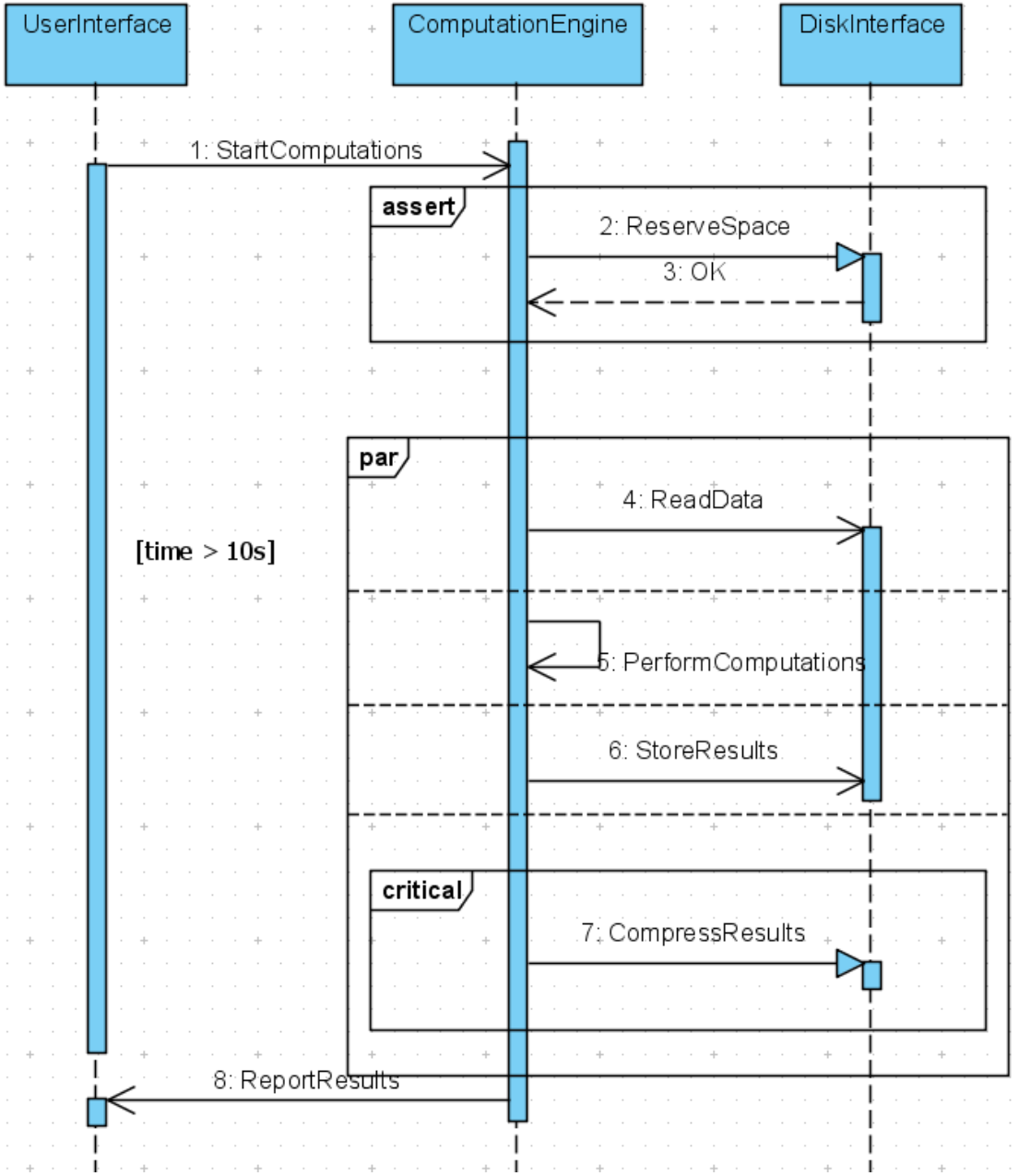
Przykładowa operacja krytyczna



Operator assert - wymagana sekwencja

- Pozwala określić sekwencję wiadomości która musi pojawić się w systemie dokładnie tak, jak określono

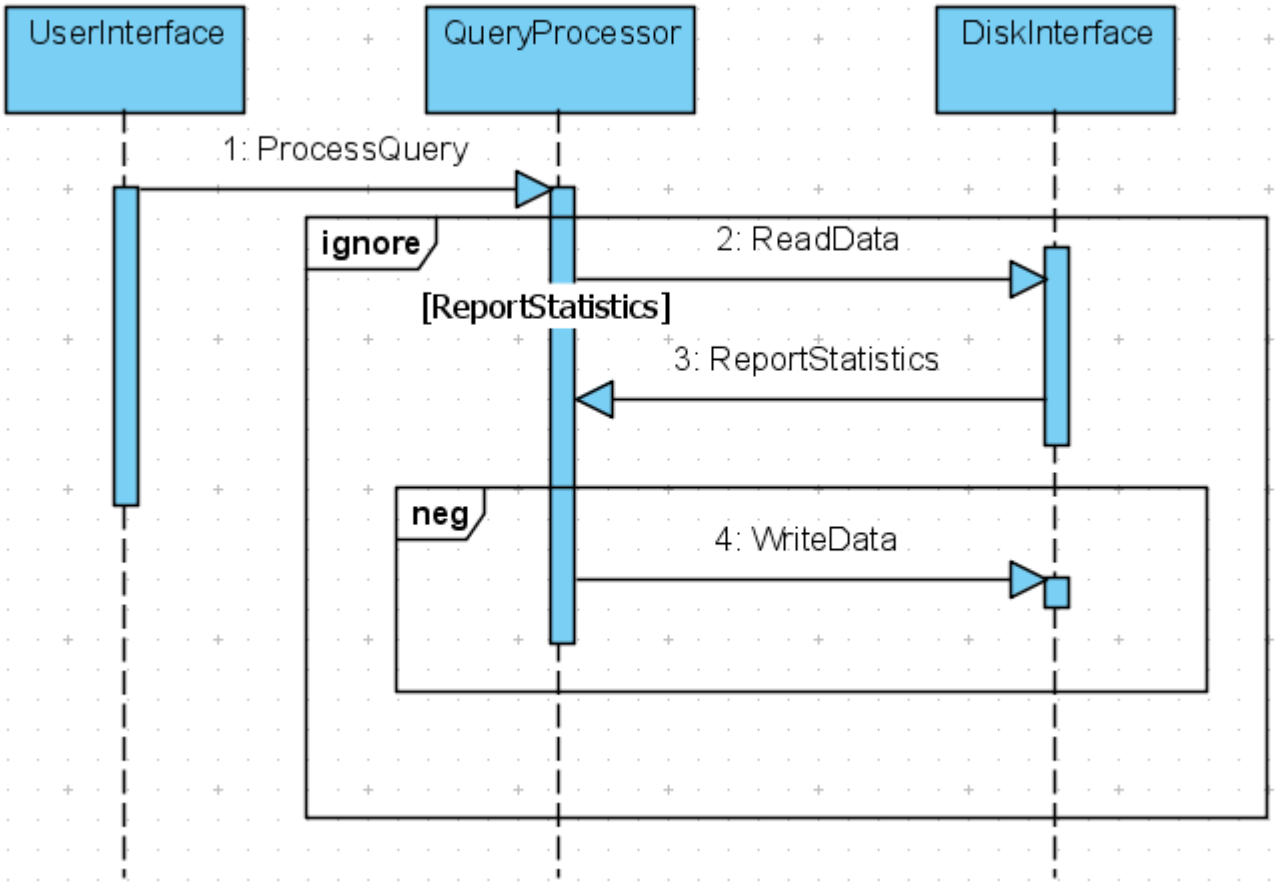
Przykład operatora assert



Operatory ignore i consider

- Ignore wskazuje wiadomości, które nie są istotne dla wykonywanego procesu
- Consider wskazuje na operacje istotne

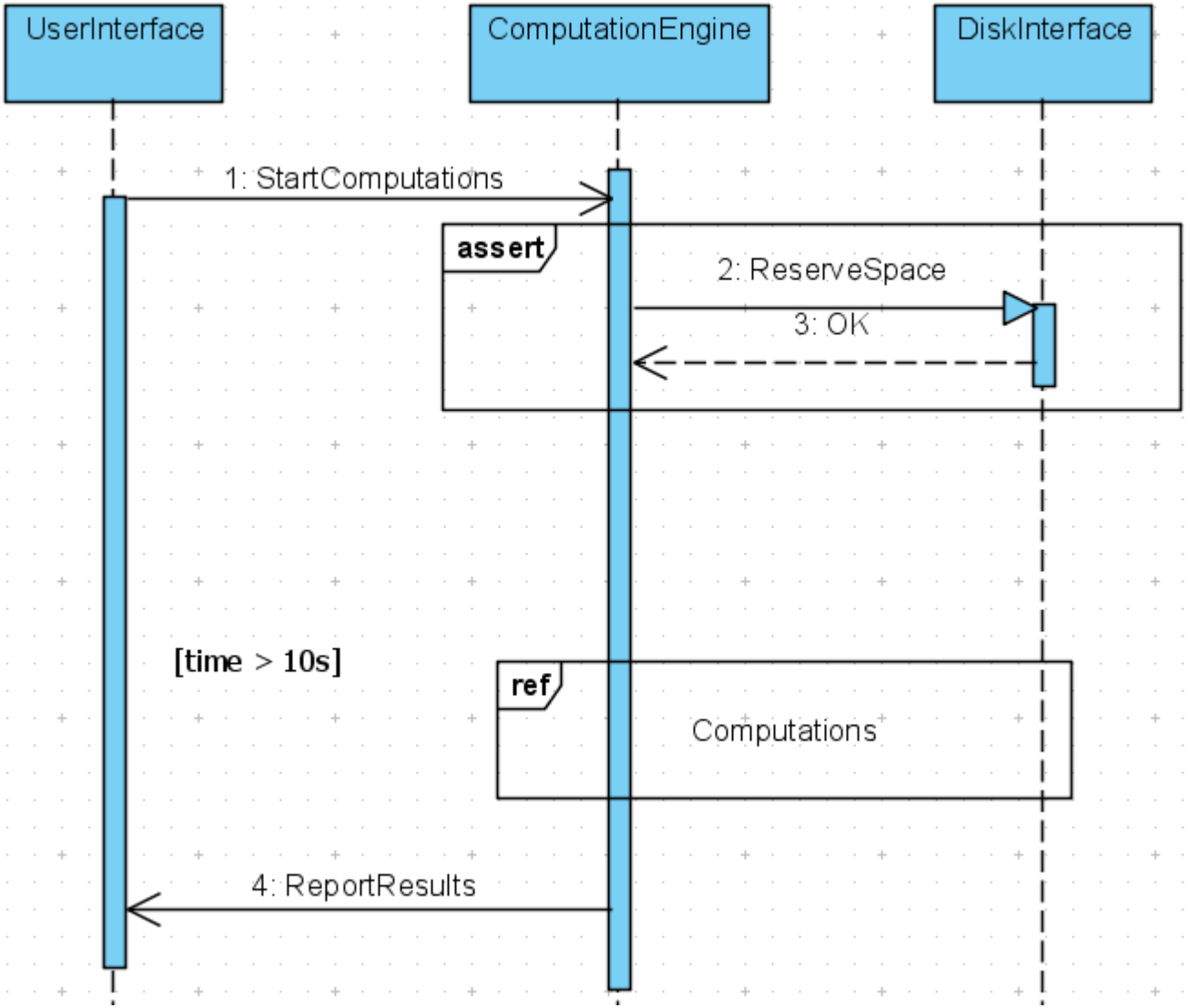
Przykład operatora ignore



Poddiagramy sekwencji

- Duże diagramy wygodnie jest dzielić na mniejsze fragmenty
- Takie poddiagramy mogą być zobrazowane na diagramie głównym poprzez tzw. Wystąpienie interakcji
- Ma ono postać ramy z nagłówkiem “ref”

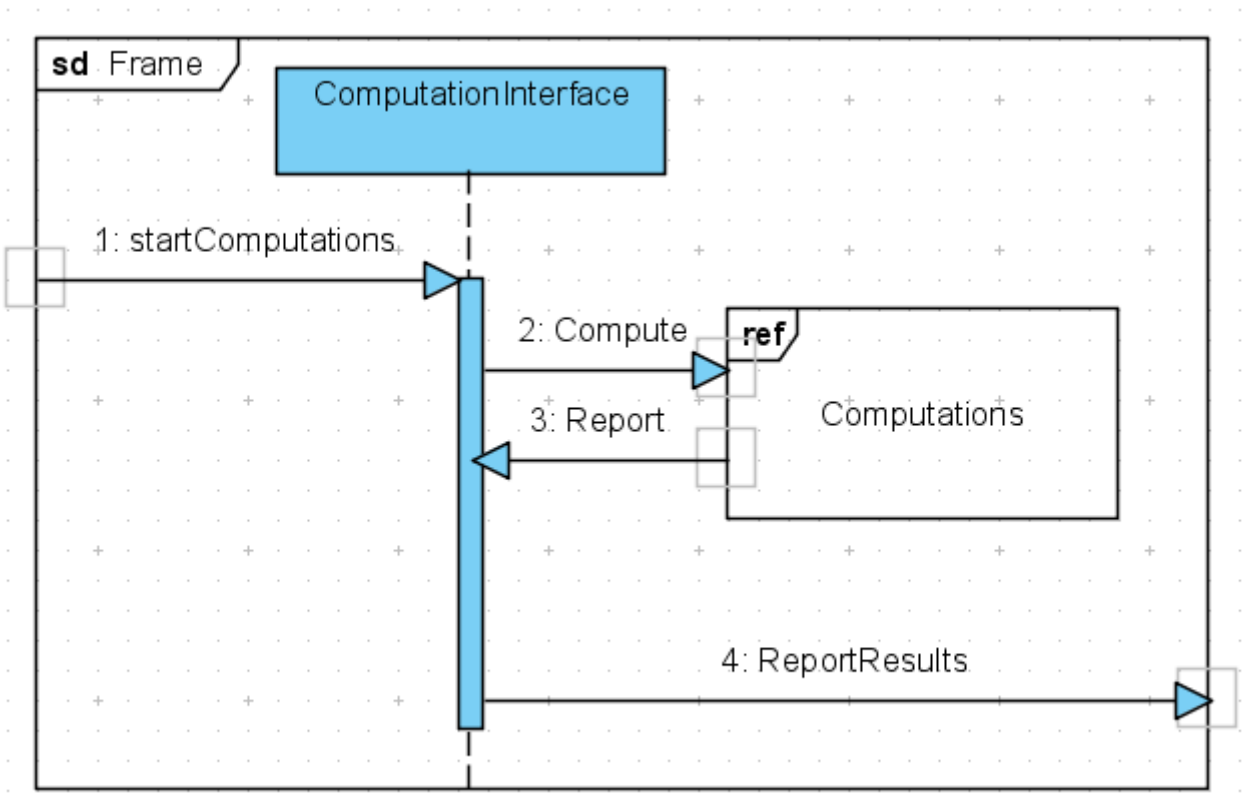
Przykładowy region ref



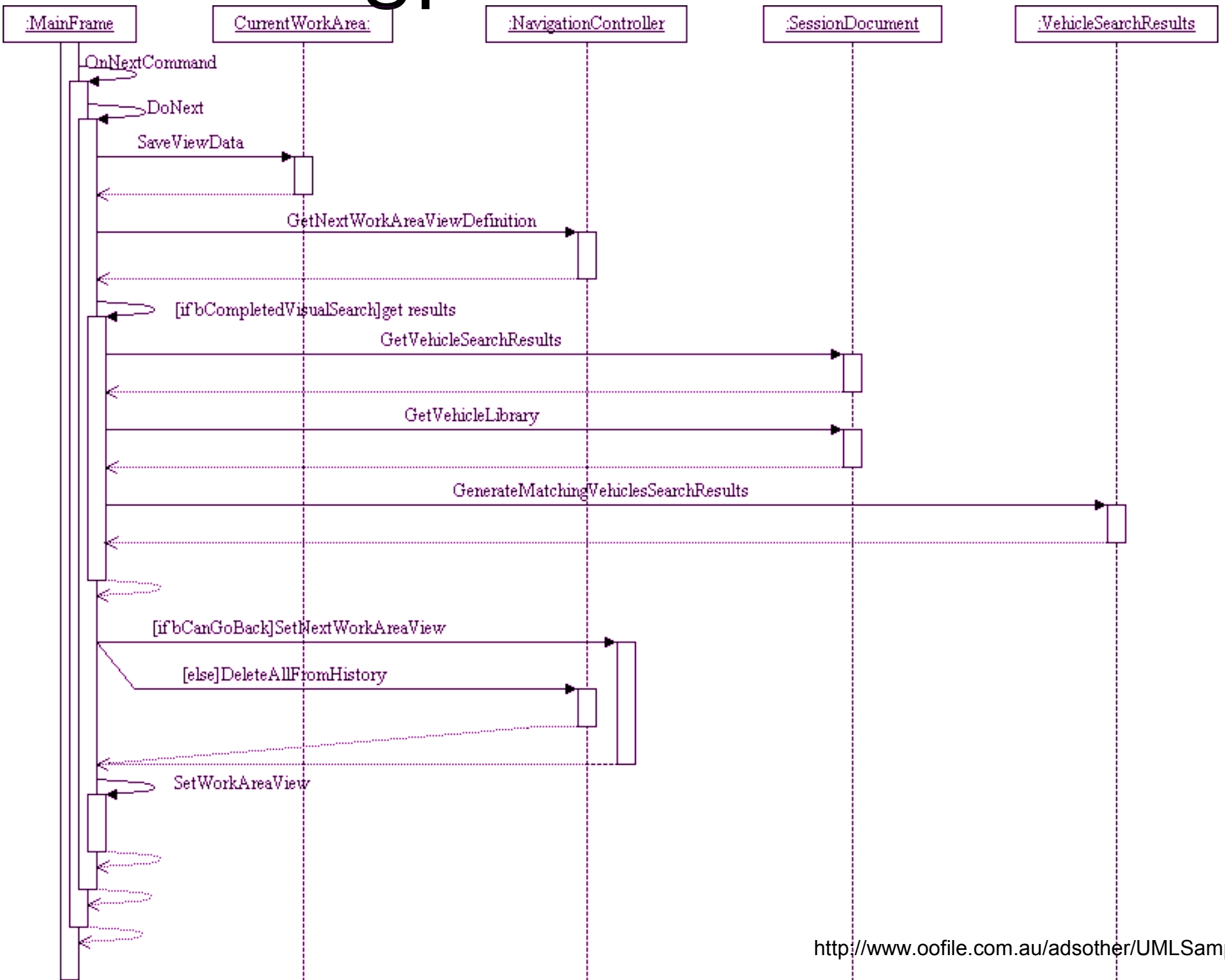
Bramy

- Służą komunikacji pomiędzy diagramami (fragmentami diagramów)
- Reprezentowane przez małe kwadraty na krawędzi diagramu

Przykład bramy

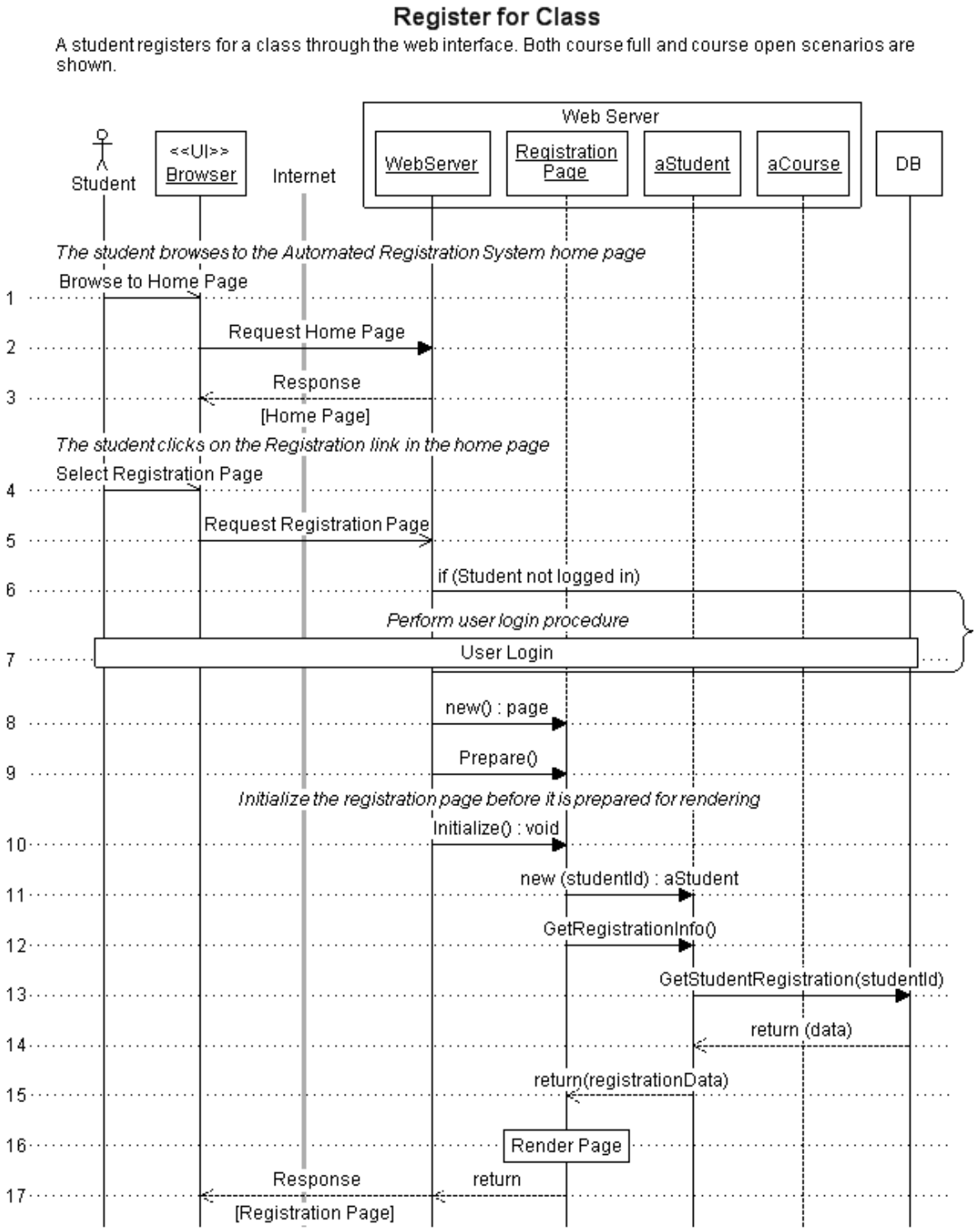


UI

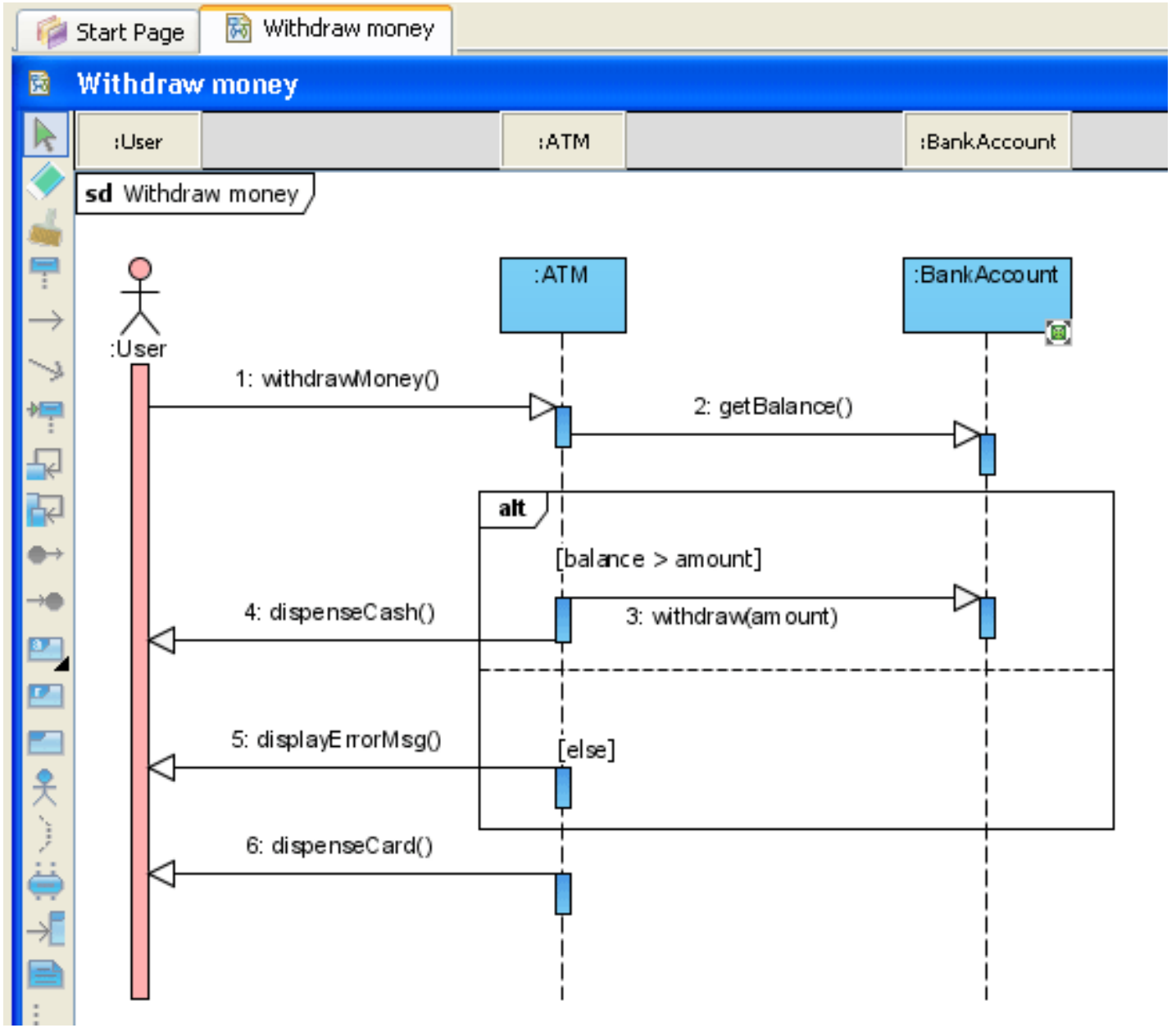


http://www.oofile.com.au/adsother/UMLSamples.html

Rejestracja

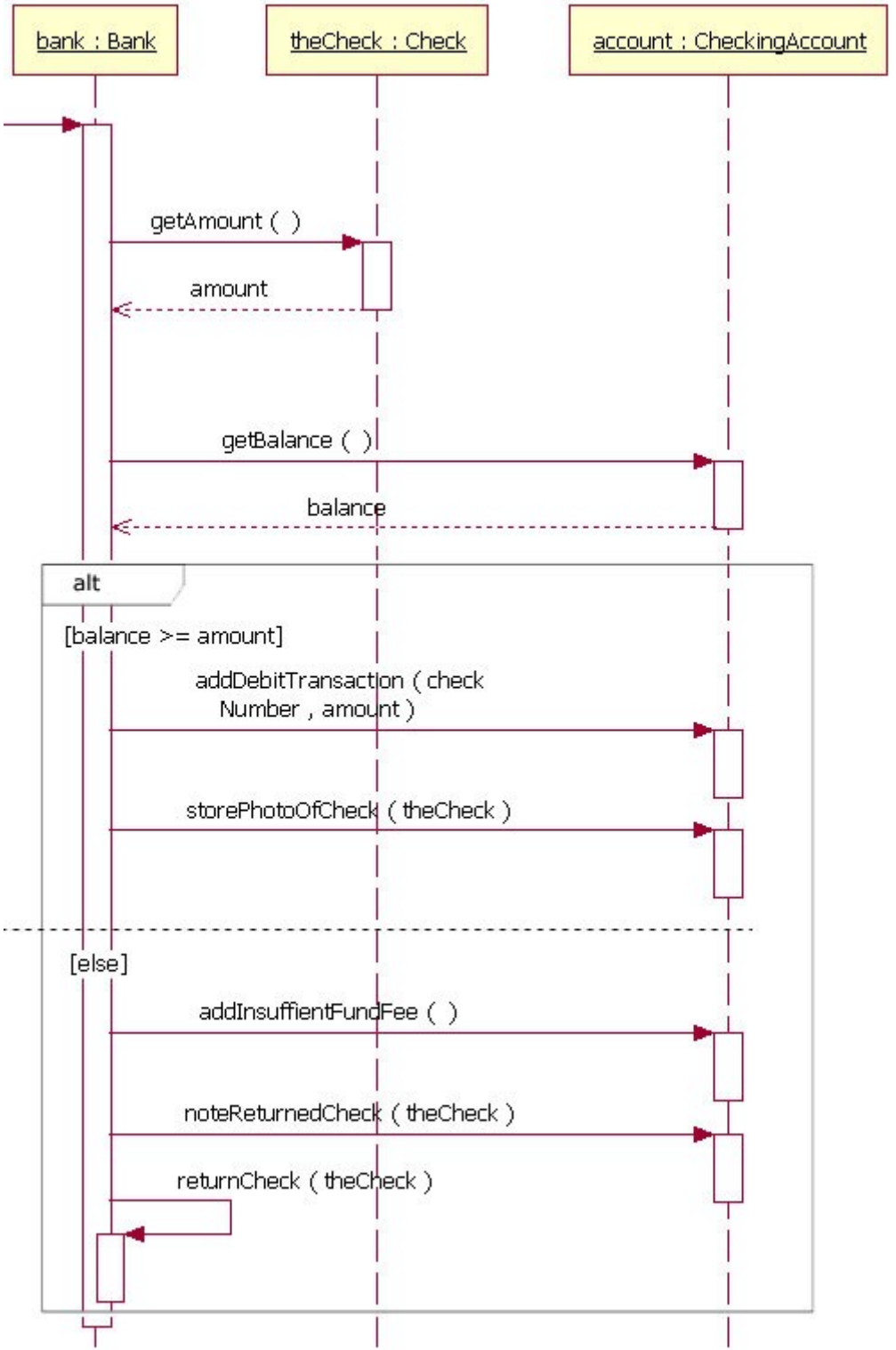


Bankomat



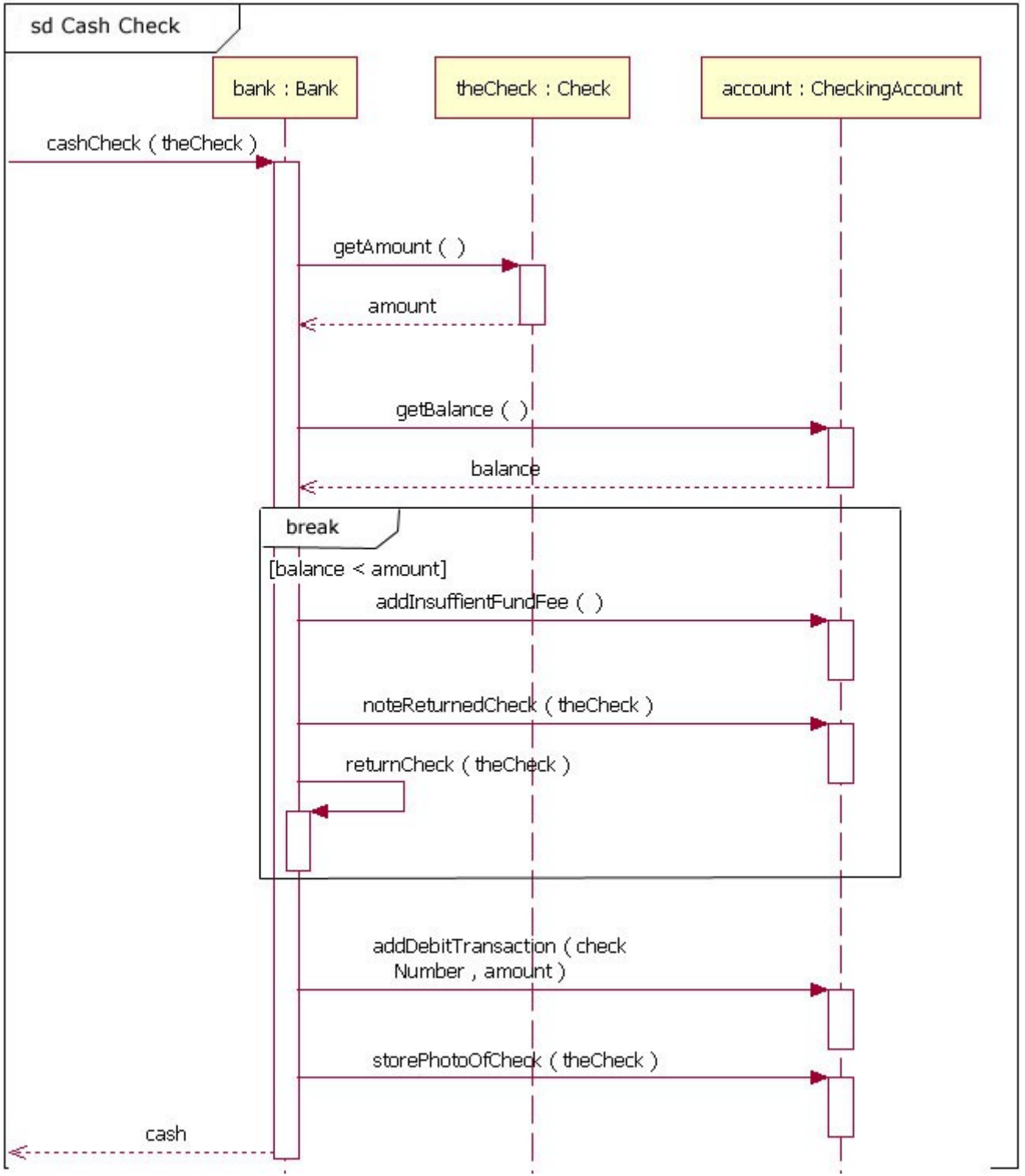
http://www.visual-paradigm.com/VPGallery/diagrams/index.html

Czek



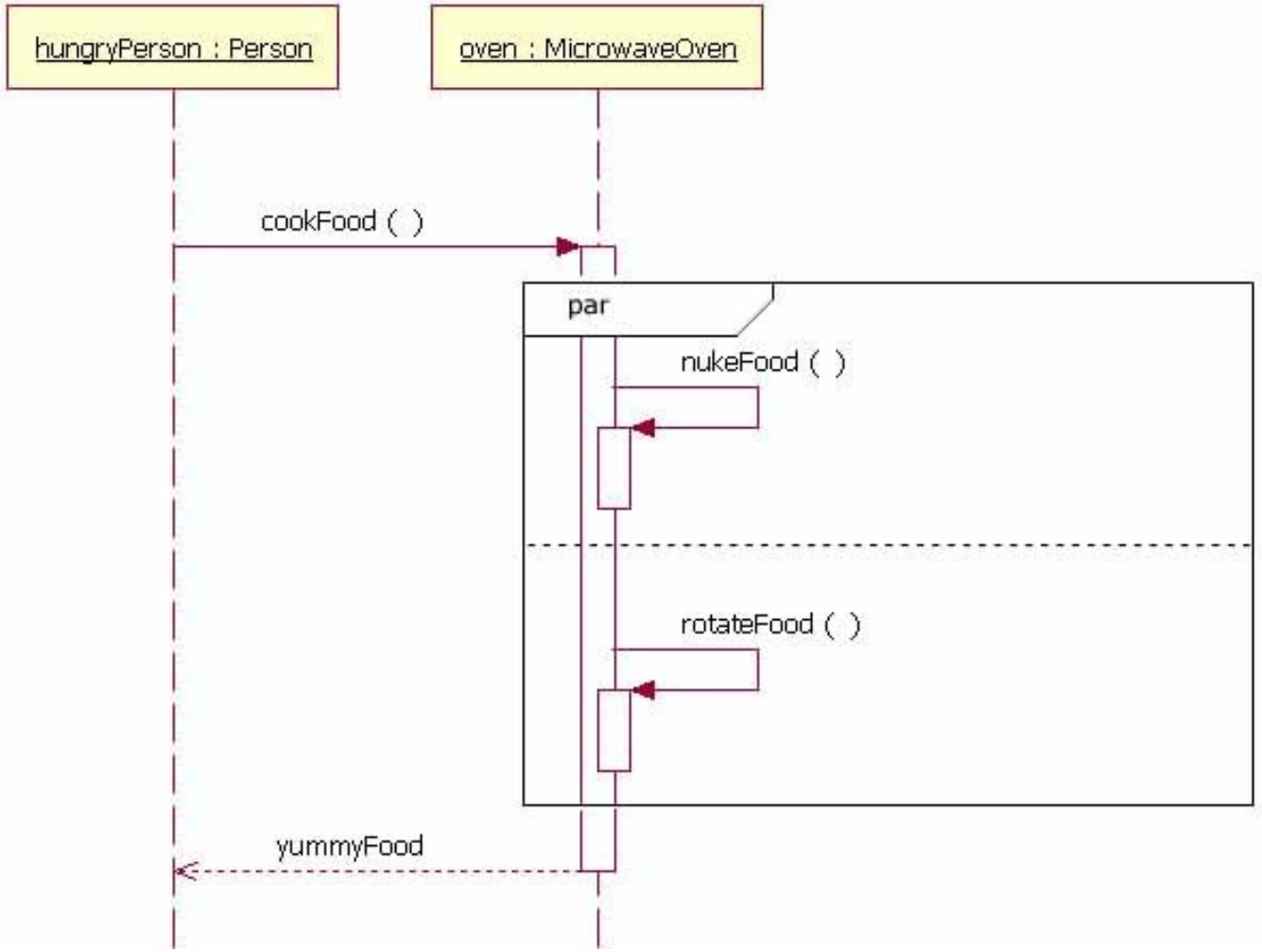
http://www-128.ibm.com/developerworks/rational/library/3101.html

Czek 2



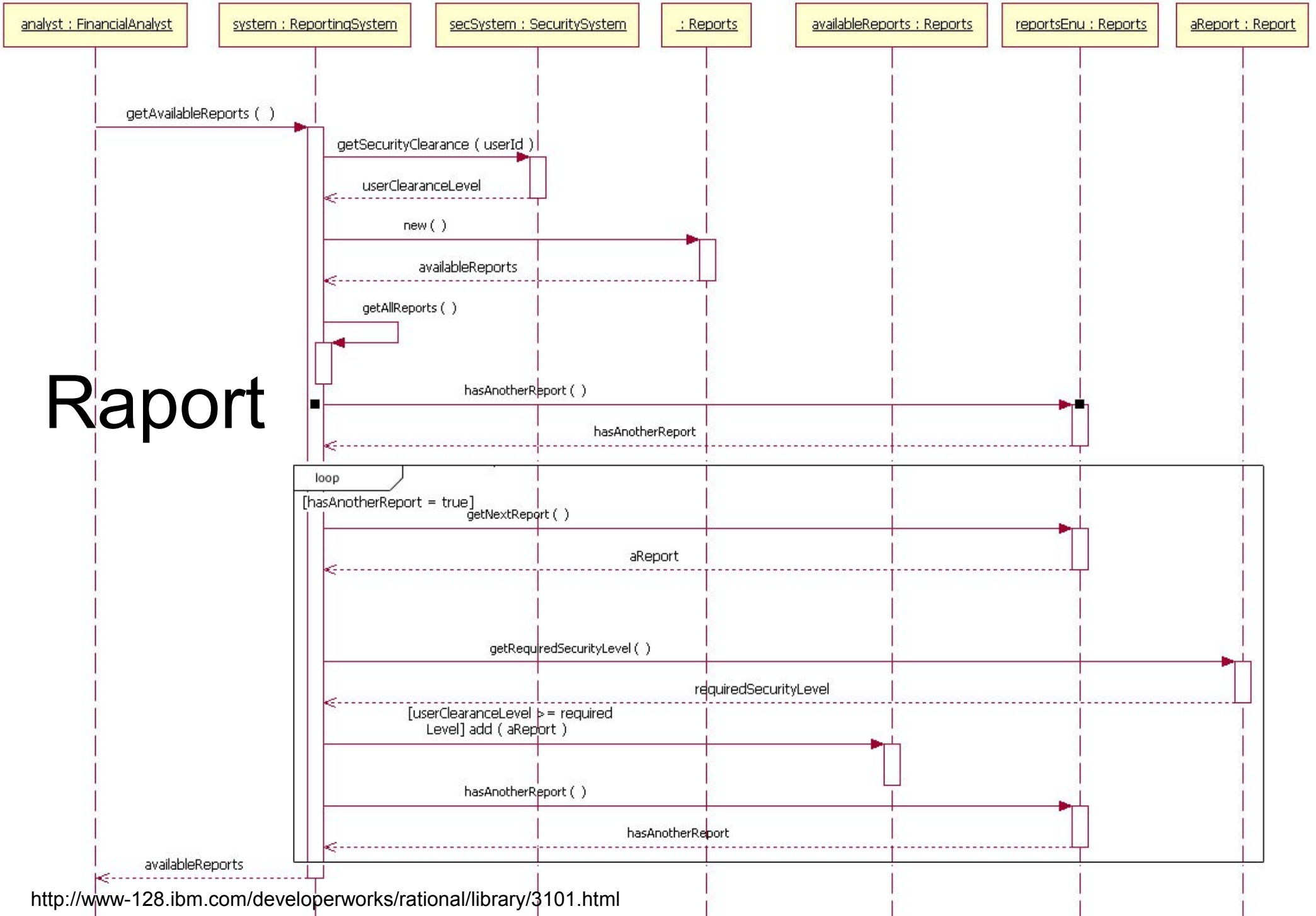
http://www-128.ibm.com/developerworks/rational/library/3101.html

Mikrofalówka



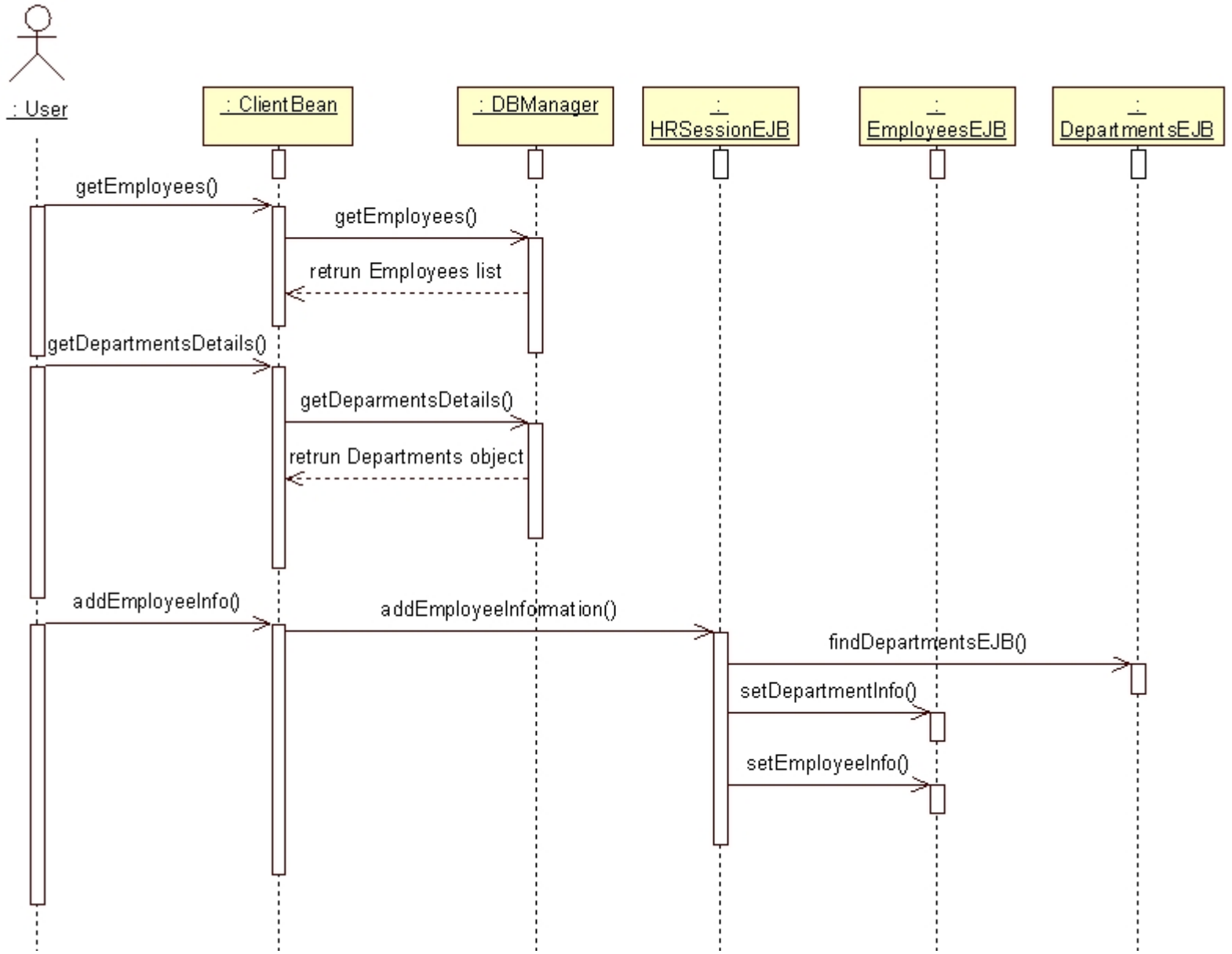
http://www-128.ibm.com/developerworks/rational/library/3101.html

Raport



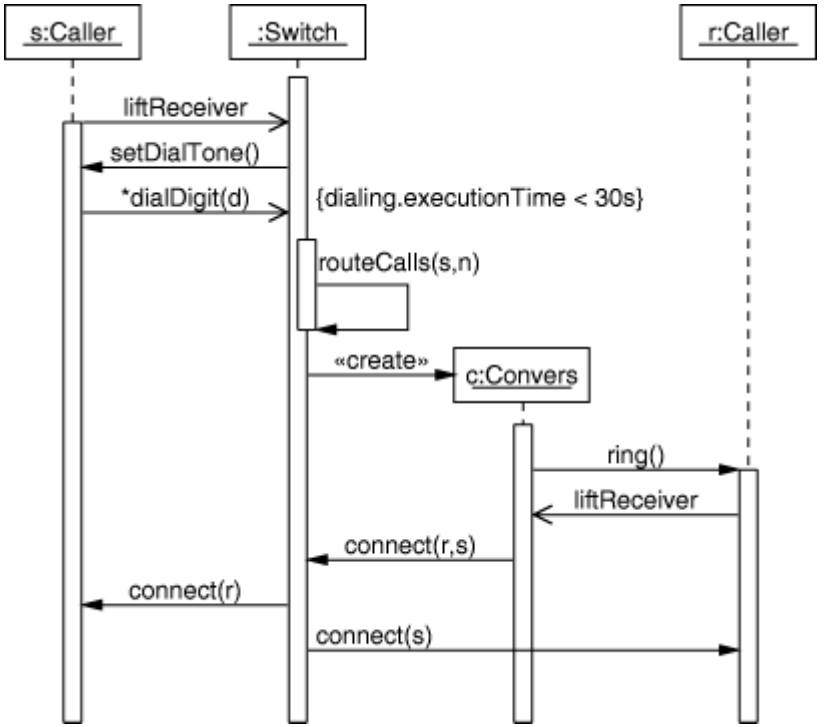
http://www-128.ibm.com/developerworks/rational/library/3101.html

Dostęp do danych



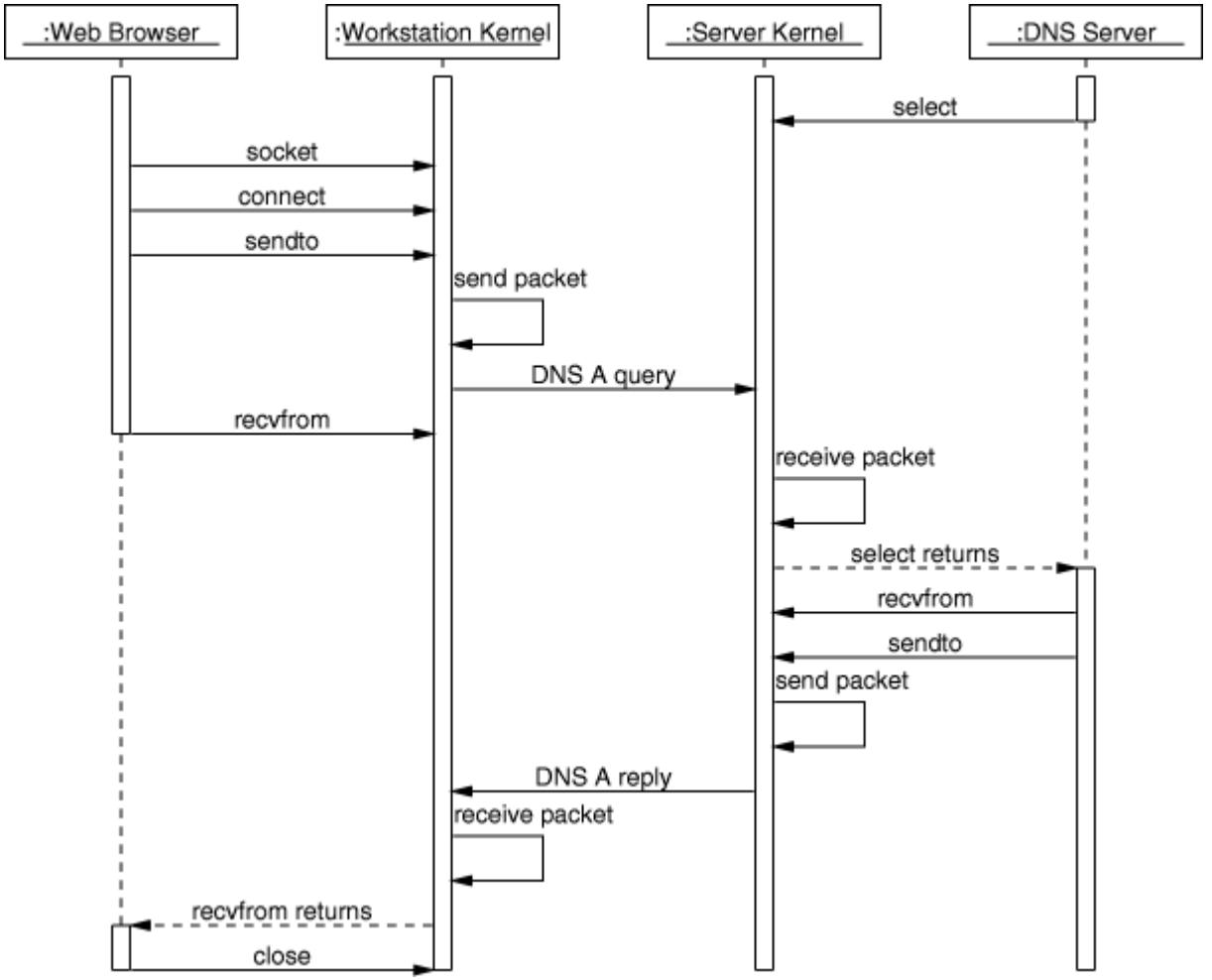
http://www.oracle.com/technology/sample_code/tech/java/j2ee/designpattern/presentationtier/fastlanereader/images/FLR_SeqDiagram.jpg

Wybieranie numeru

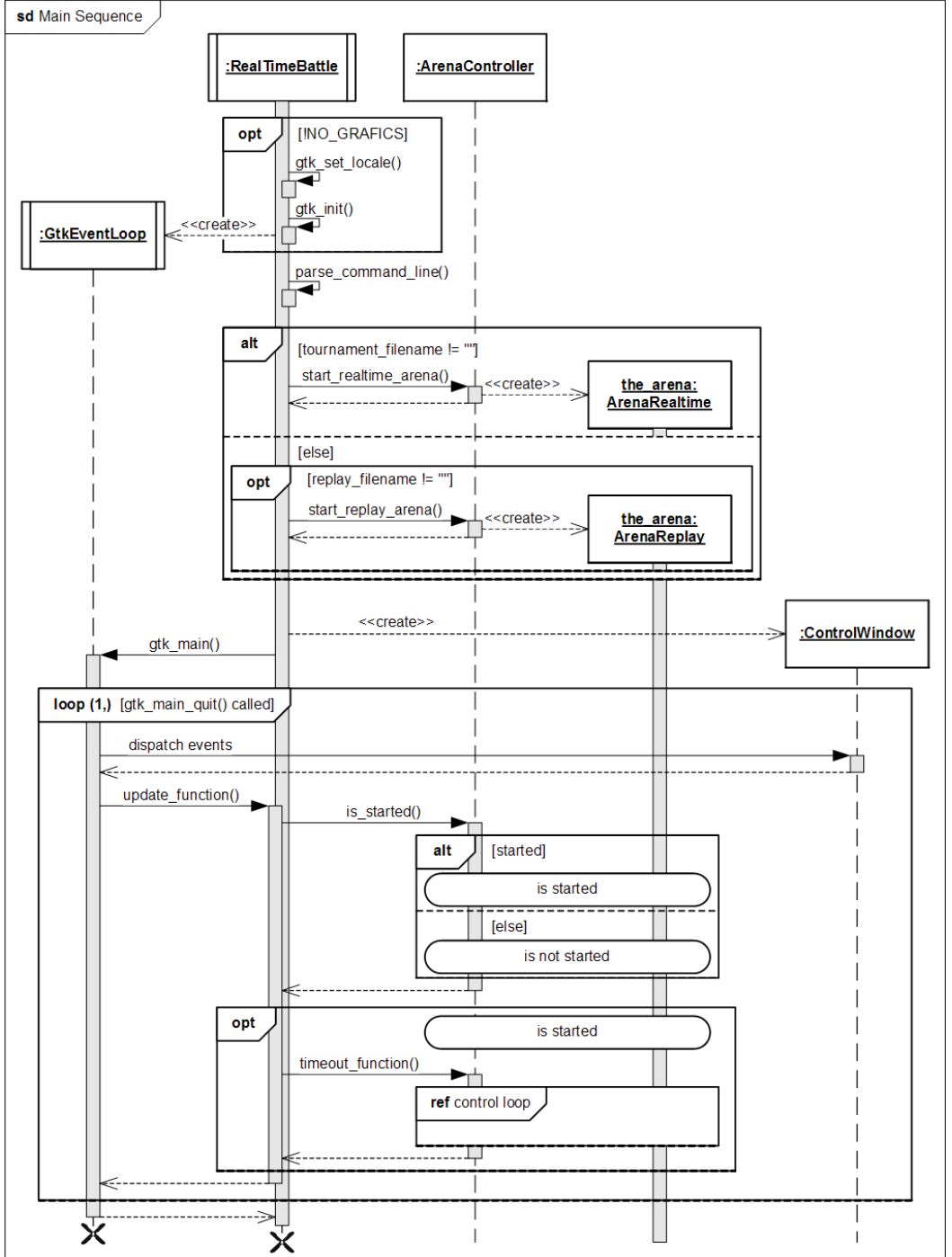


http://www.umlgraph.org/doc/uml-184.gif

DNS



http://www.umlgraph.org/doc/dnsq.gif



http://rtb-team.sourceforge.net/analysis-diagrams/images/UML/RealTimeBattle%20-%20Main%20Sequence%20(UML%202.0%20sequence%20diagram).png

Diagramy komunikacji (communication)

- Zastosowanie zbliżone do diagramów sekwencji
- Wykorzystują inne podejście:
  - Wiadomości nie są uszeregowane na osi czasu,
  - Obiekty mogą być umieszczone gdziekolwiek na diagramie
- Aby zaznaczyć kolejność wiadomości stosuje się numerację
- Diagramy komunikacji mogą być przekształcone (nawet automatycznie) w diagramy sekwencji – i odwrotnie

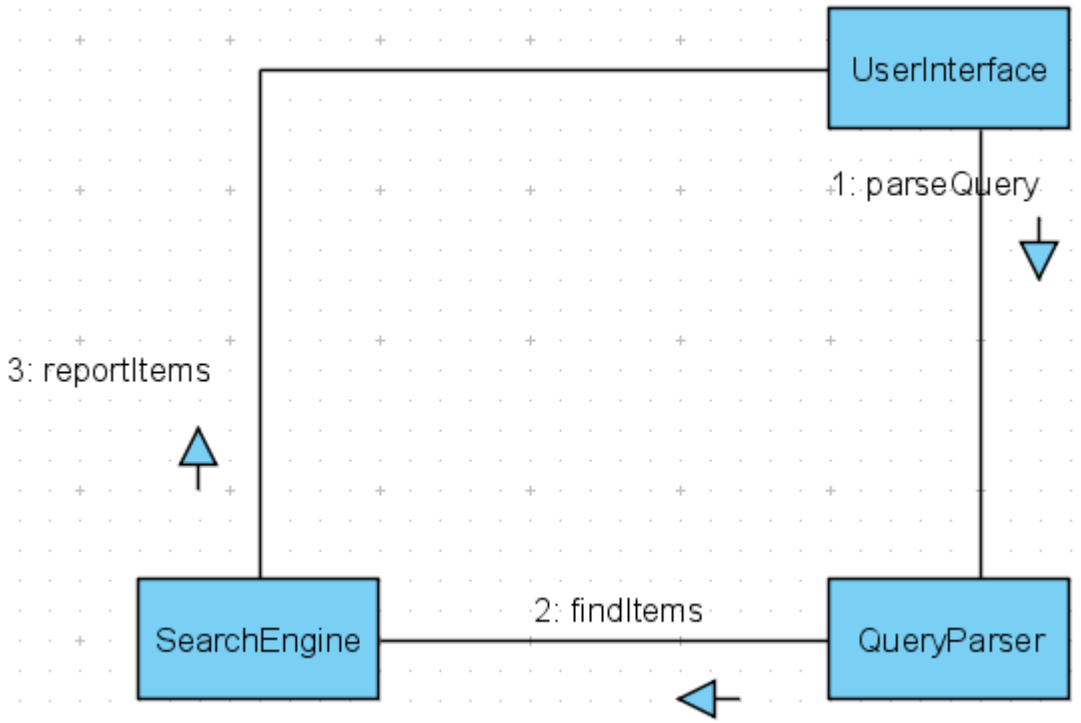
Ograniczenia

- Nie wszystkie elementy diagramów sekwencji mogą zostać zastosowane
- Brak:
  - Wiadomości zgubionych i znalezionych
  - Fragmentów złożonych

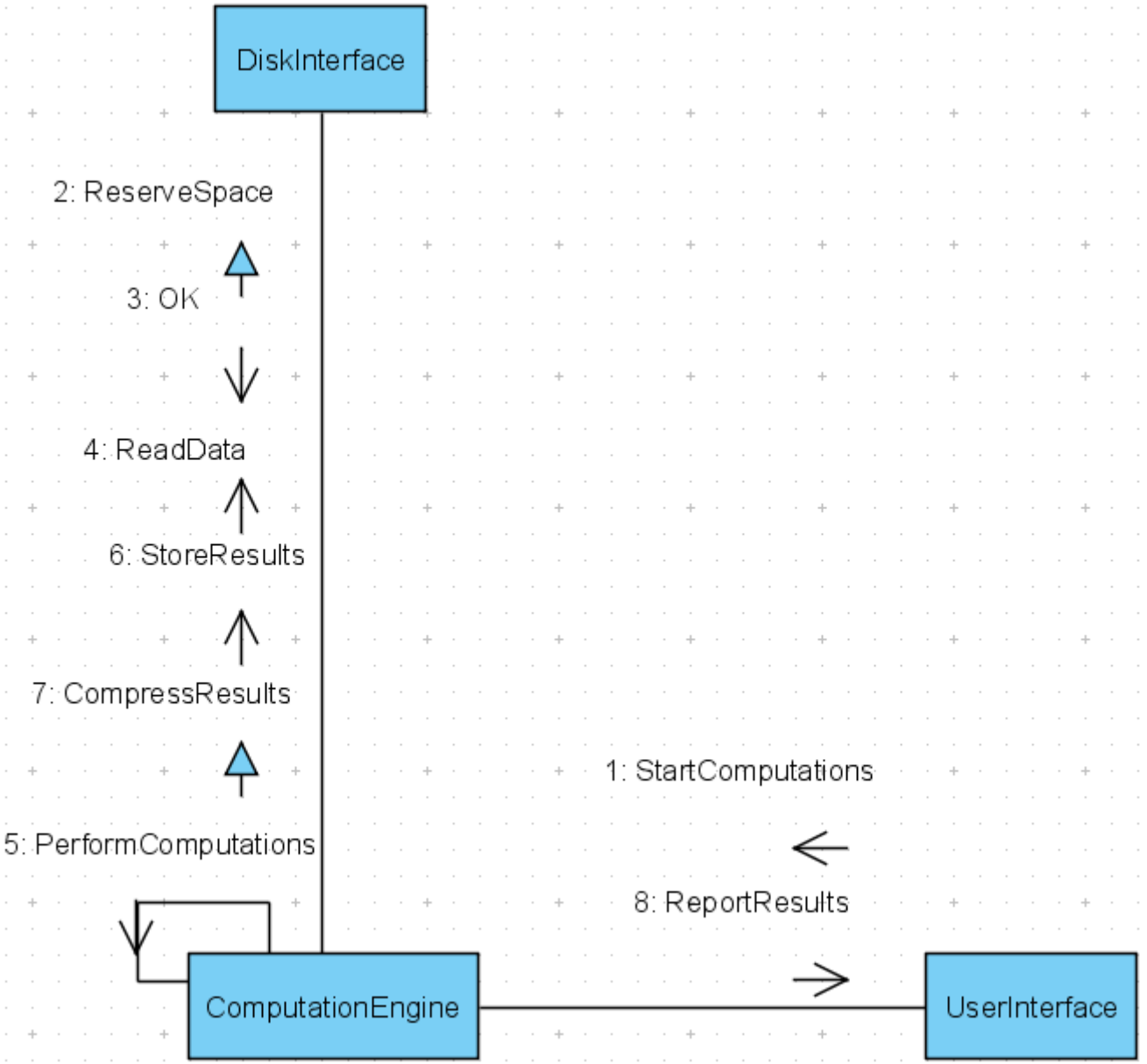
Elementy diagramu komunikacji

- Objekt – jak w diagramie sekwencji, ale bez linii życia
- Połączenie – wskazuje, że dwa obiekty komunikują się. Nie symbolizuje żadnej określonej wiadomości. Obrazowane linią łączącą obiekty
- Wiadomość – jak w diagramie sekwencji. Obrazowana krótką strzałką i opisem wiadomości, umieszczonymi w pobliżu połączenia

Przykładowy diagram komunikacji



Przykładowy diagram komunikacji



Przykładowy diagram komunikacji

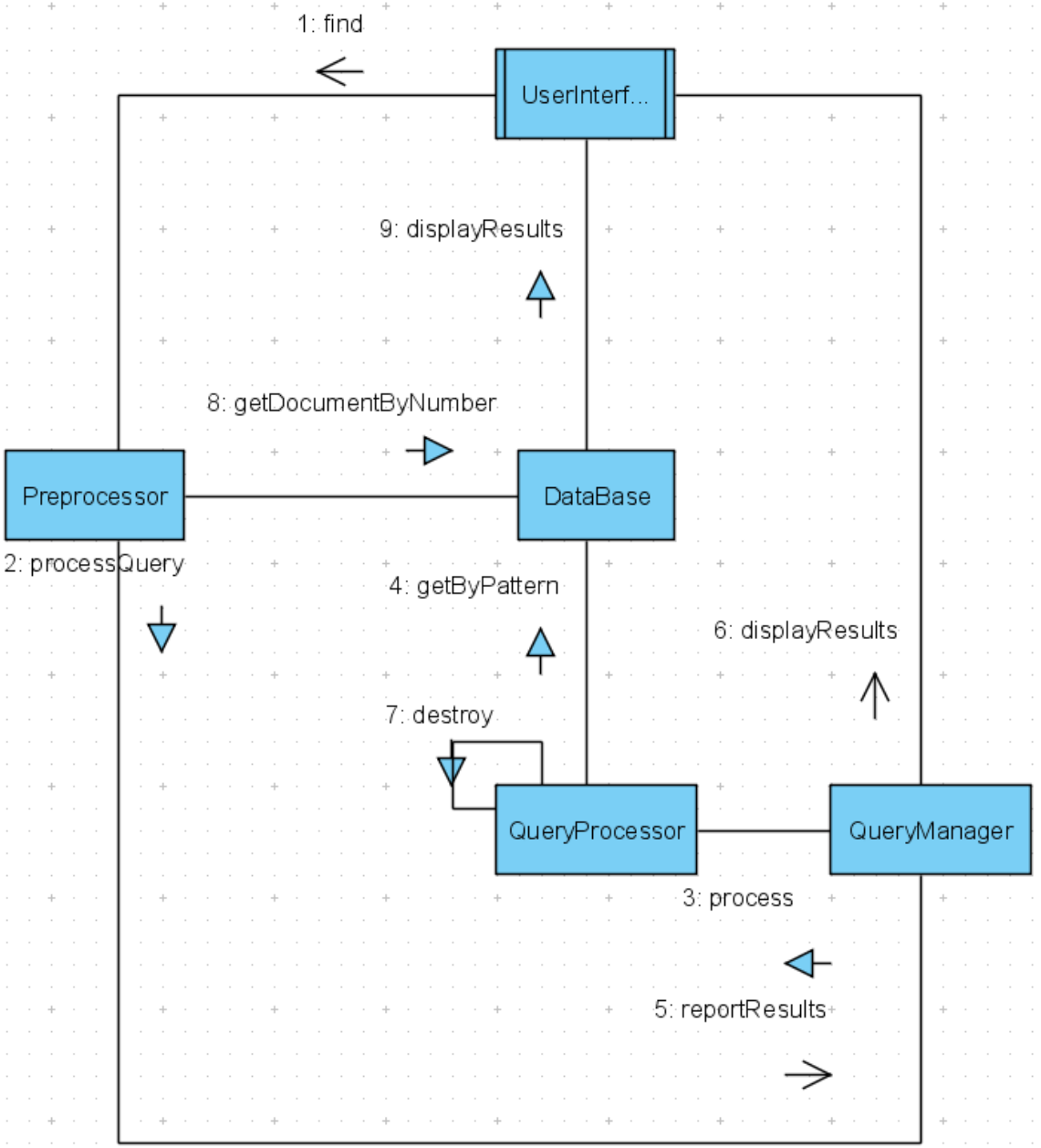


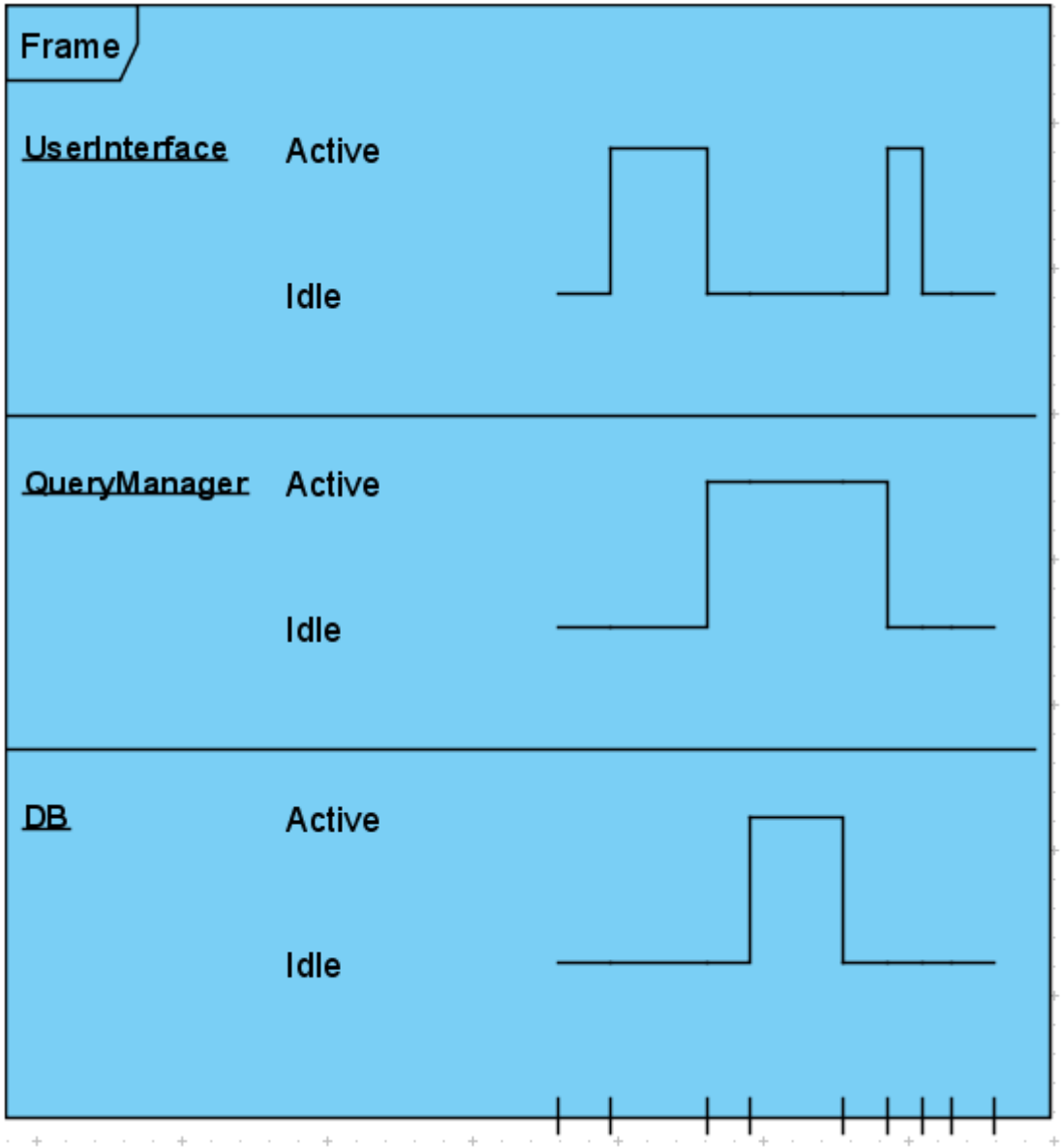
Diagram przebiegów czasowych (timing)

- Używany do ilustracji zmiany stanów obiektów
- Zmiany opisane są w ścisłym odniesieniu do czasu
- Są przydatne jeśli precyzyjne umiejscowienie w czasie jest niezbędne (multimedia, aplikacje czasu rzeczywistego)
- Są powiązane z diagramami sekwencji i maszyn stanowych

Diagram przebiegów czasowych (timing)

- Podstawowy diagram składa się z ramy, zawierającej:
  - Skalę czasu (oś pozioma, dolna krawędź)
  - Nazwy klasyfikatorów których dotyczą stany
  - Nazwy stanów (dla każdego klasyfikatora). Typowe stany to: aktywny, bezczynny, oczekujący itp.
  - Linie życia (dla każdego klasyfikatora) pokazujące zmiany stanów w czasie

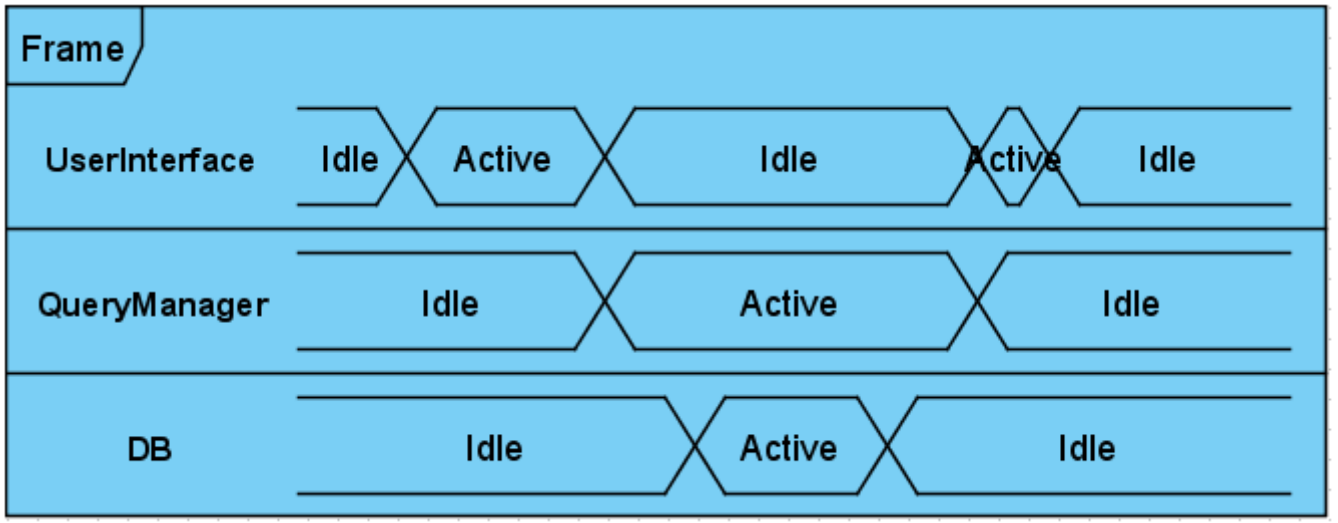
Przykładowy diagram



Notacja alternatywna

Diagram można przedstawić w alternatywnej kompaktowej notacji

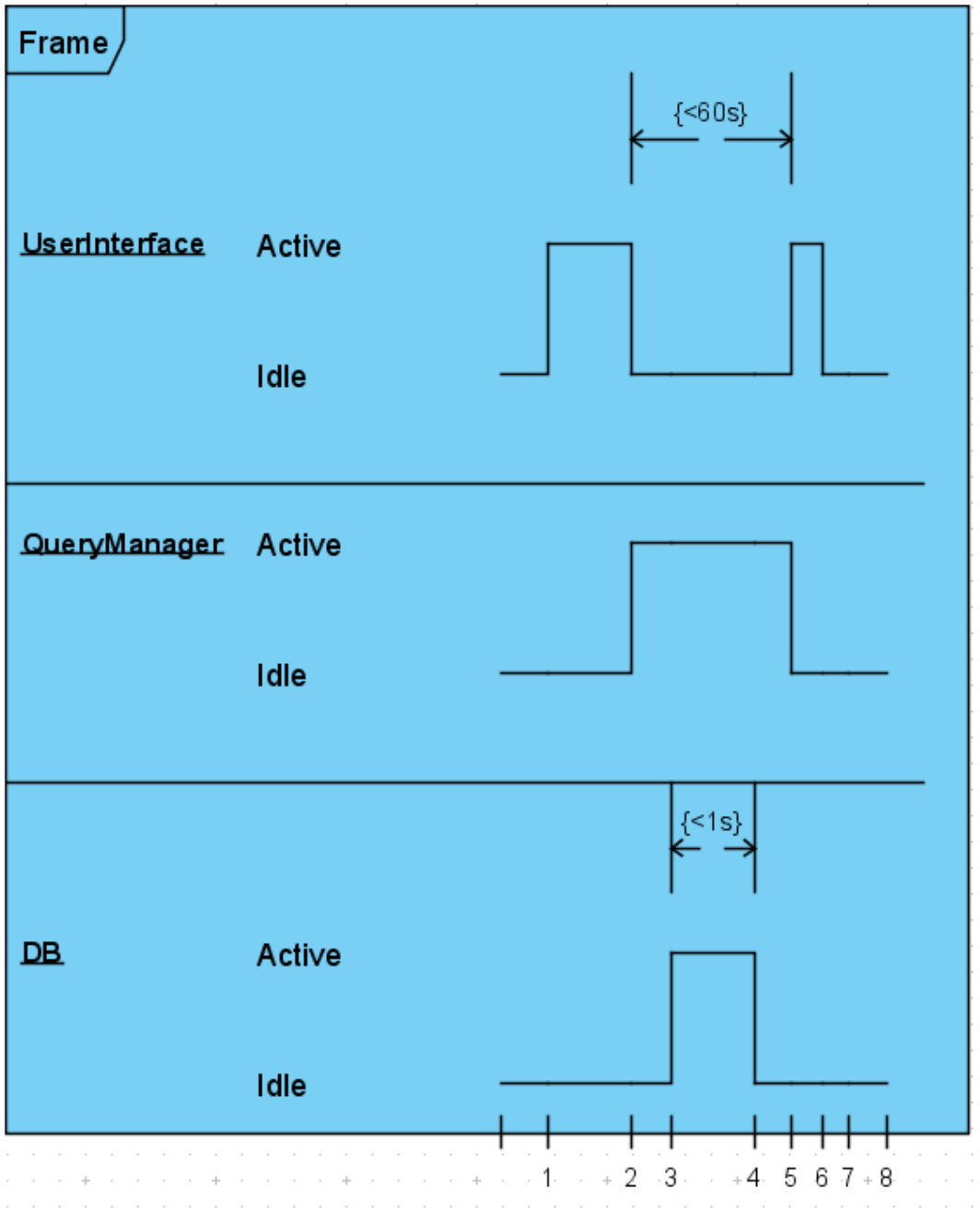
Nie wszystkie elementy są dopuszczalne w takiej notacji



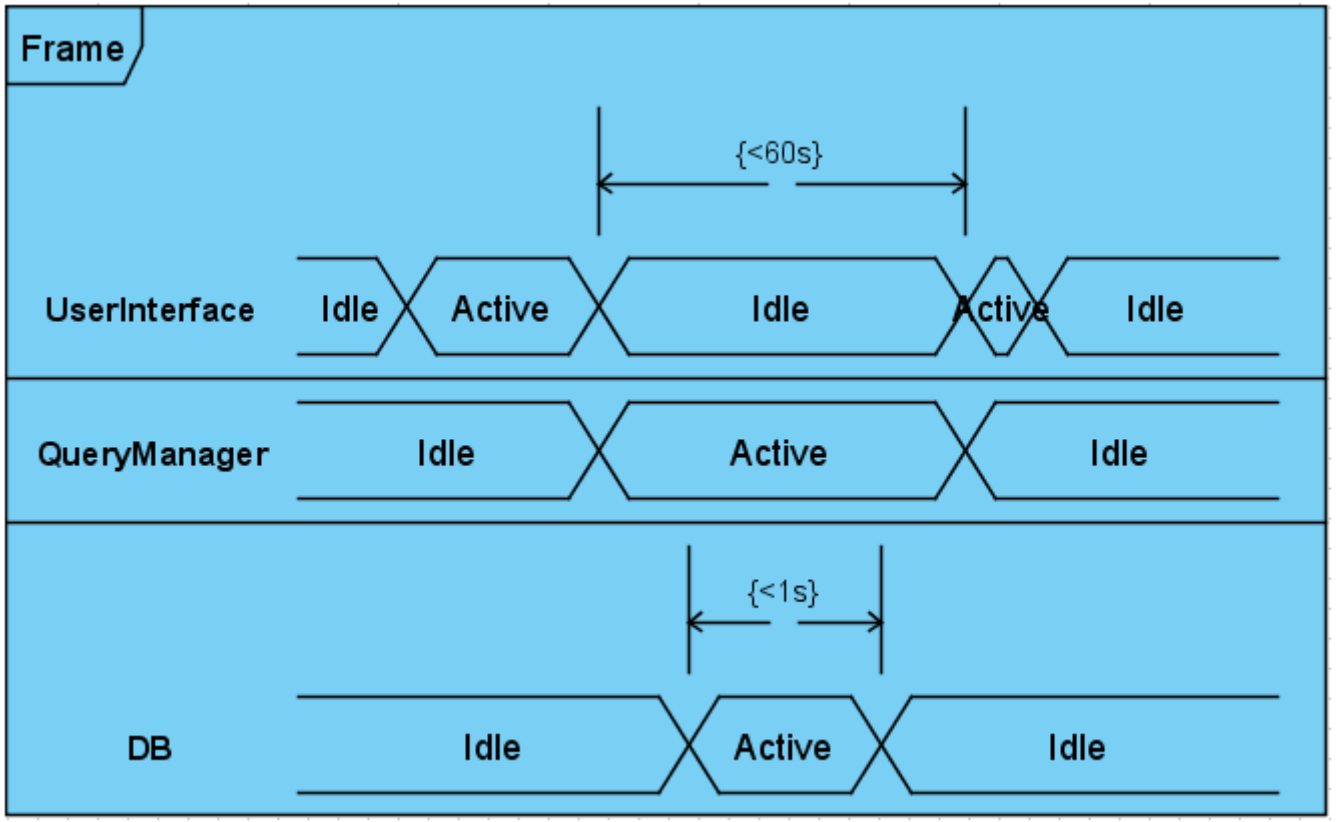
Ograniczenia czasowe

- Można określić ograniczenia precyzujące czas trwania stanów
- Mogą zawierać każdy rodzaj wyrażeń, są umieszczane nad linią życia
- Mogą być wyświetlane zarówno w diagramie pełnym, jak i kompaktowym

Przykład ograniczeń czasowych



Przykład ograniczeń czasowych



Pobudzenia

- Można określić zdarzenia powodujące zmiany stanów
- Są wyświetlane jako wyrażenia w pobliżu punktu zmiany stanu
- Nie są widoczne w trybie kompaktowym

Przykład pobudzeń

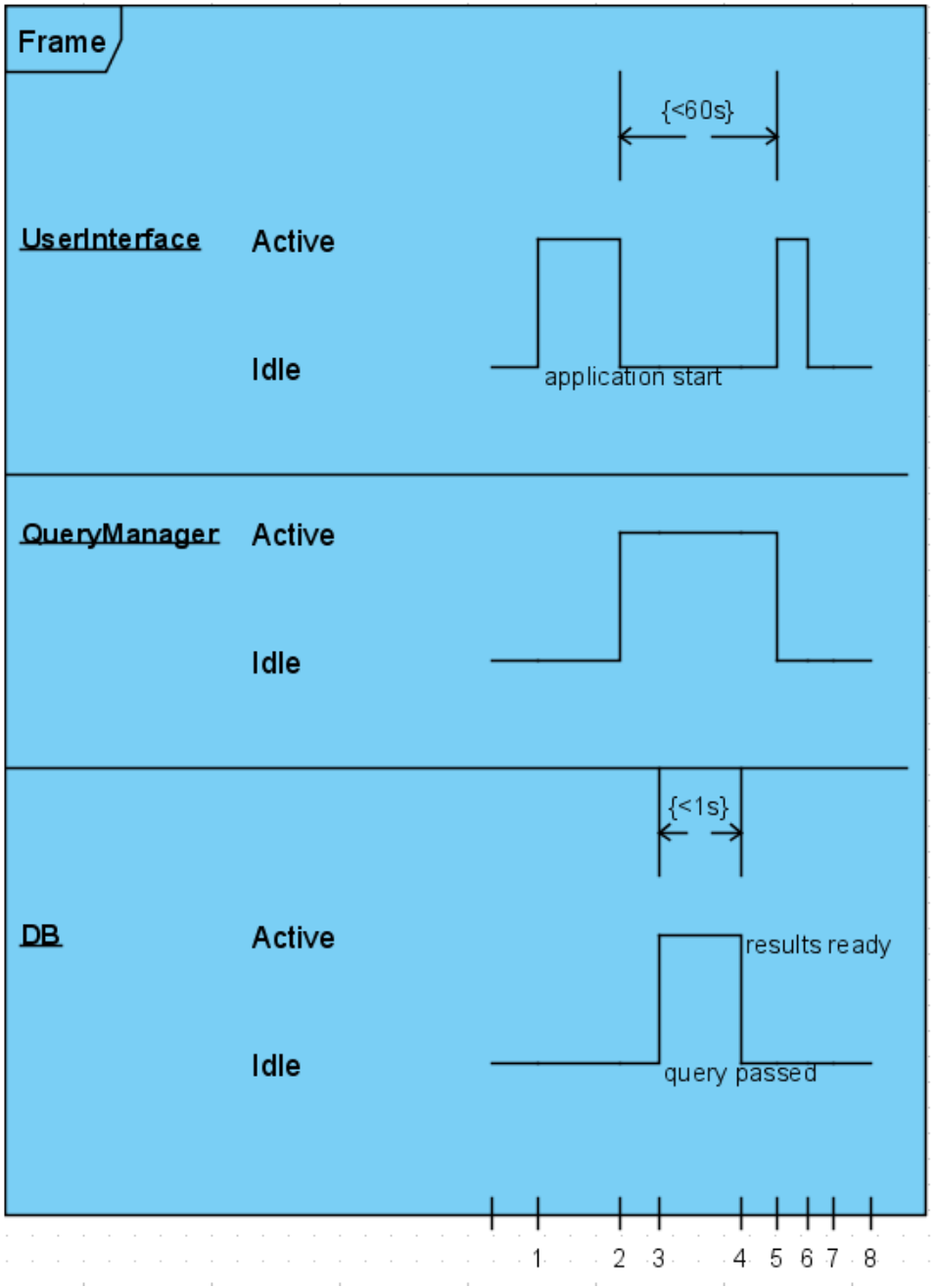


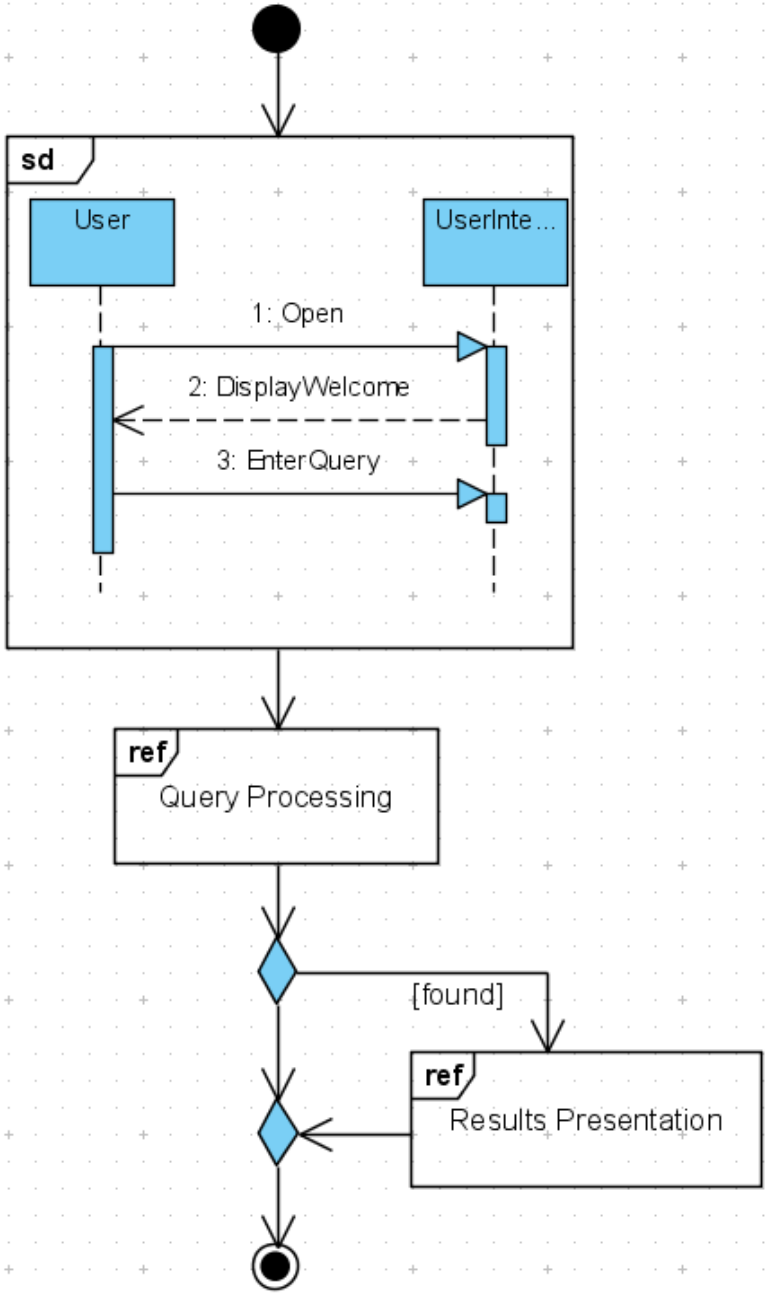
Diagram opisu interakcji

- Umożliwia połączenie diagramów sekwencji, komunikacji i przebiegów czasowych przy pomocy notacji podobnej do diagramów czynności
- Przydatne w dużych systemach

Diagram opisu interakcji

- Diagramy sd, cd i td mogą być reprezentowane na dwa sposoby:
  - jako region REF wskazujący na inny diagram
  - Jako rama zawierająca pełną specyfikację diagramu
- Oba podejścia mogą być łączone w jednym diagramie
- Mogą występować elementy diagramów czynności: przepływy, węzły początkowe i końcowe, węzły decyzji, złączenia itd.

Przykład diagramu opisu interakcji



Diagramy komponentów (component)

- Opisują interakcje między komponentami (modułami) systemu
- Komponent jest częścią systemu która wchodzi w interakcje z innymi komponentami poprzez interfejsy. W stosunku do interfejsów występuje związek dziedziczenia (realizacji)
- Komponent jest związany z pojęciem wielożywalności
- Komponenty sa obrazowane symbolem podobnym do symbolu klasy, mogą być stereotypowane

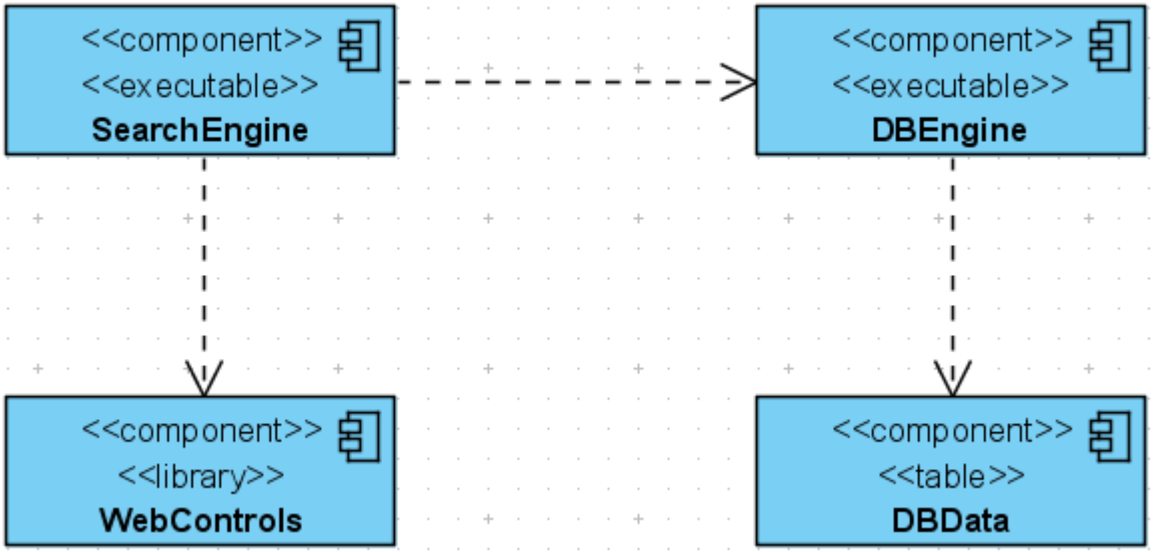
Diagramy komponentów

- Typowe komponenty to:
  - Pliki wykonywalne
  - Biblioteki
  - Bazy danych
  - Podsystemy
  - Usługi

Zależność

- Najprostszy sposób powiązania komponentów to zależność
- Jest obrazowana strzałką z przerywaną linią skierowanś od komponentu zależnego (użytkującego pewne usługi) do niezależnego (zapewniającego pewne usługi).

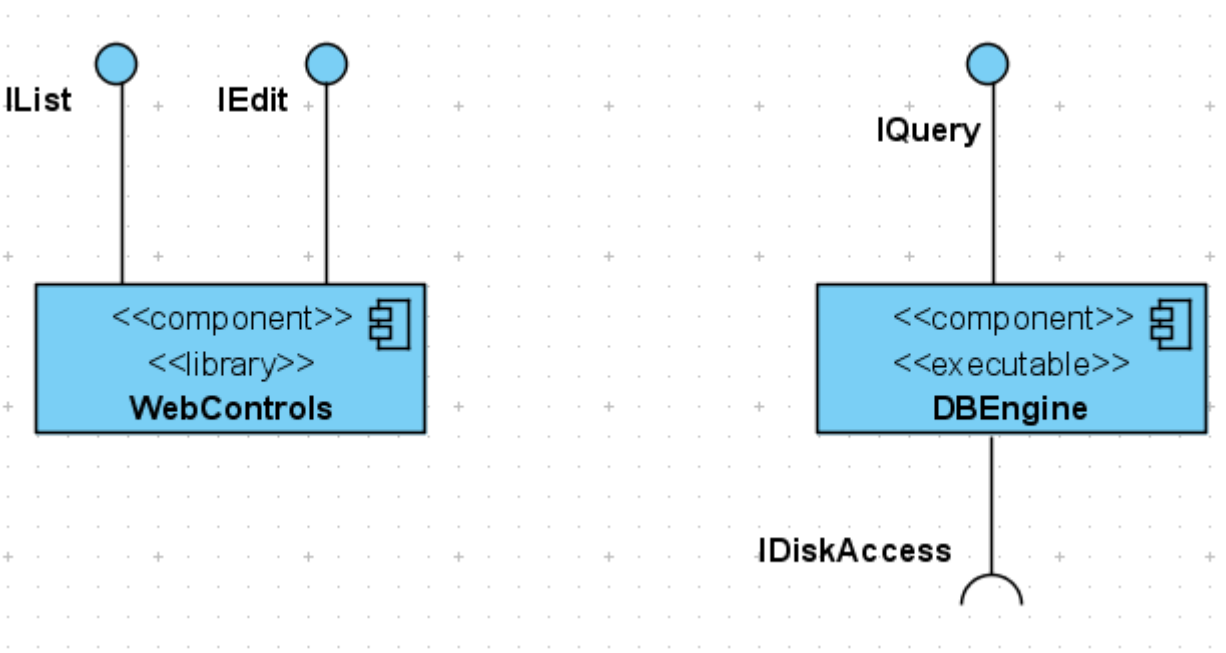
Przykładowy diagram komponentów



Interfejsy

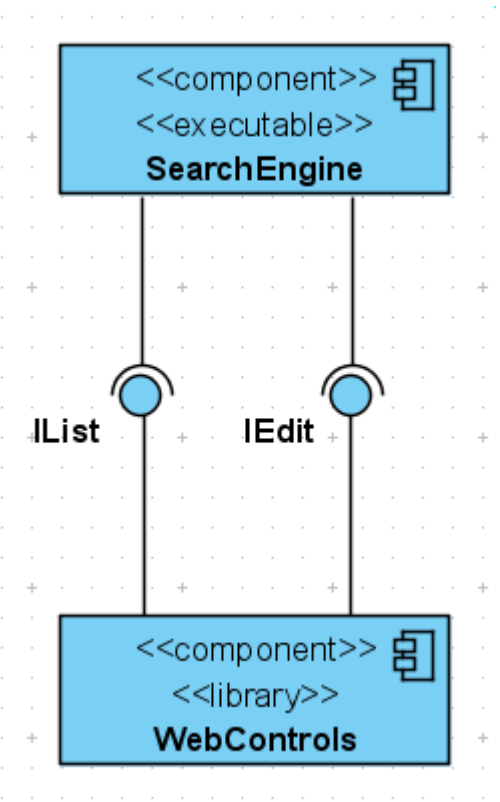
- Więcej szczegółów o komponentach i ich związkach można podać zaznaczając ich interfejsy
- Dwie odrębne sytuacje są możliwe:
  - Komponent realizuje interfejs, czyli implementuje funkcjonalność związaną z interfejsem i może ją zaoferować innym komponentom. Jest to obrazowane przez kółko
  - Komponent jest zależny od interfejsu, czyli potrzebuje usług innego komponentu, implementującego interfejs. Jest to obrazowane przez gniazdo (połowa okręgu)

Przykładowe interfejsy



Interfejsy

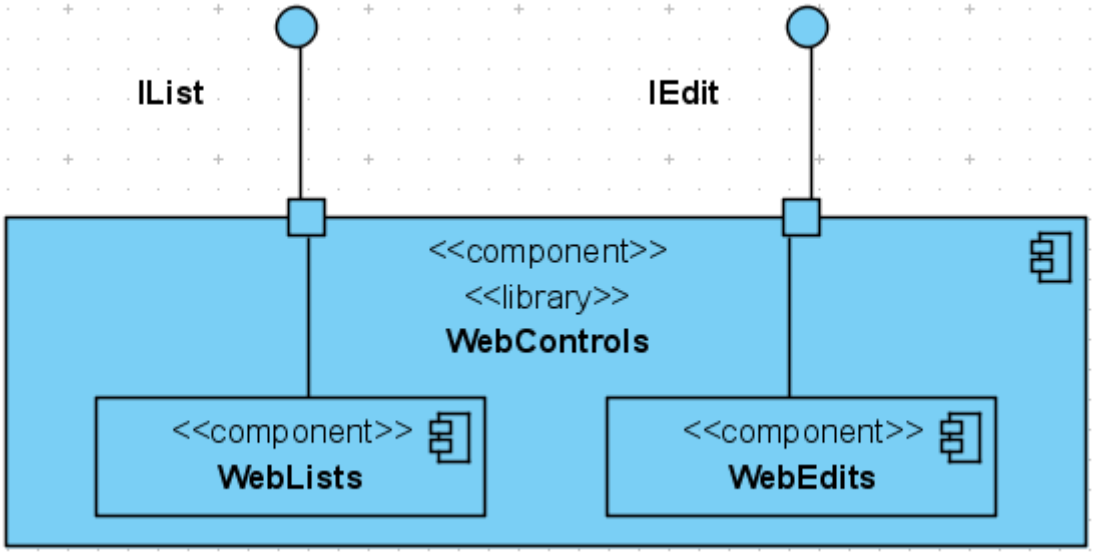
- Połączenia kółko-gniazdo są stosowane aby precyzyjnie określić zależność



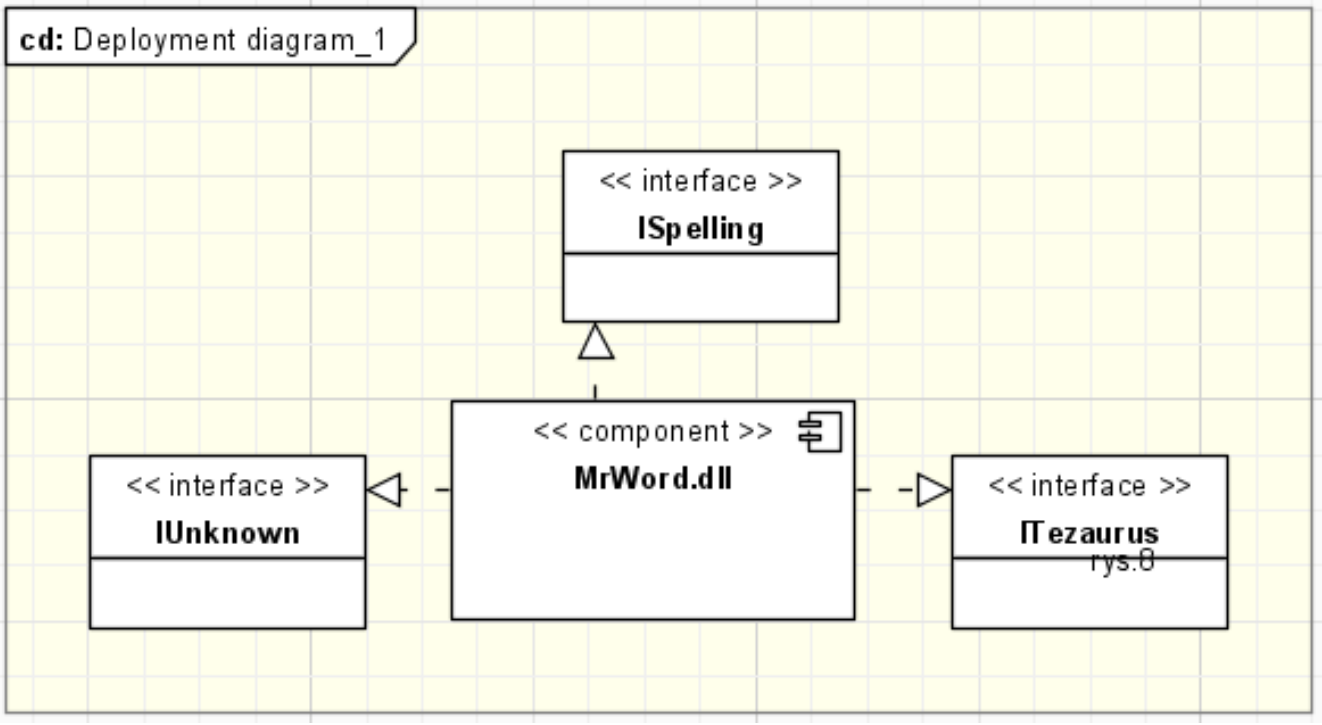
Struktura wewnętrzna

- Więcej szczegółów można przekazać prezentując strukturę wewnętrzną - subkomponenty
- W takim przypadku można pokazać które sub-komponenty używają interfejsów głównego komponentu
- Do tego celu stosuje się porty
- Obrazowane są małymi kwadratami na krawędzi komponentu

Komponent z portami

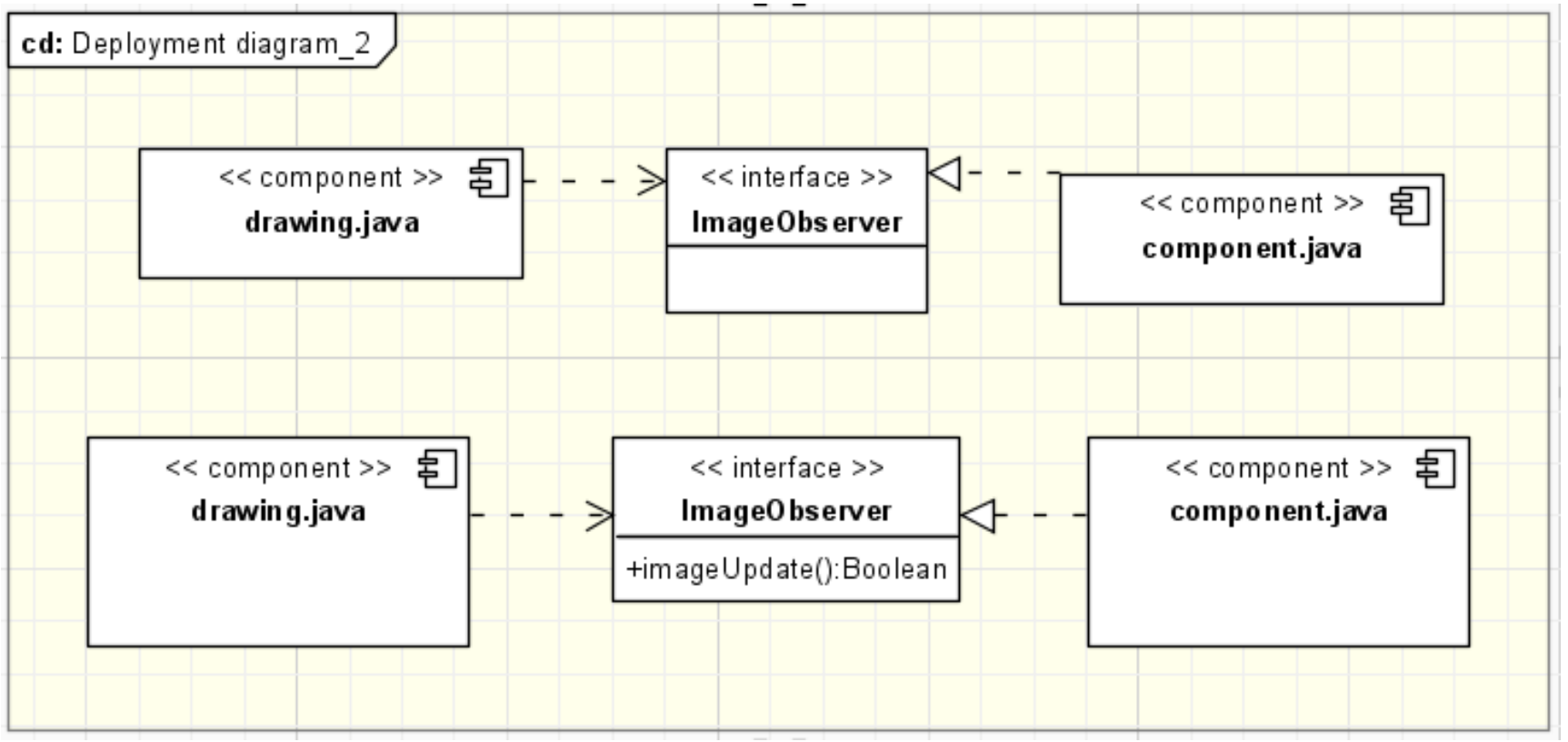


Przykładowy diagram



rys.8

Przykładowy diagram



Diagramy wdrożenia (deployment)

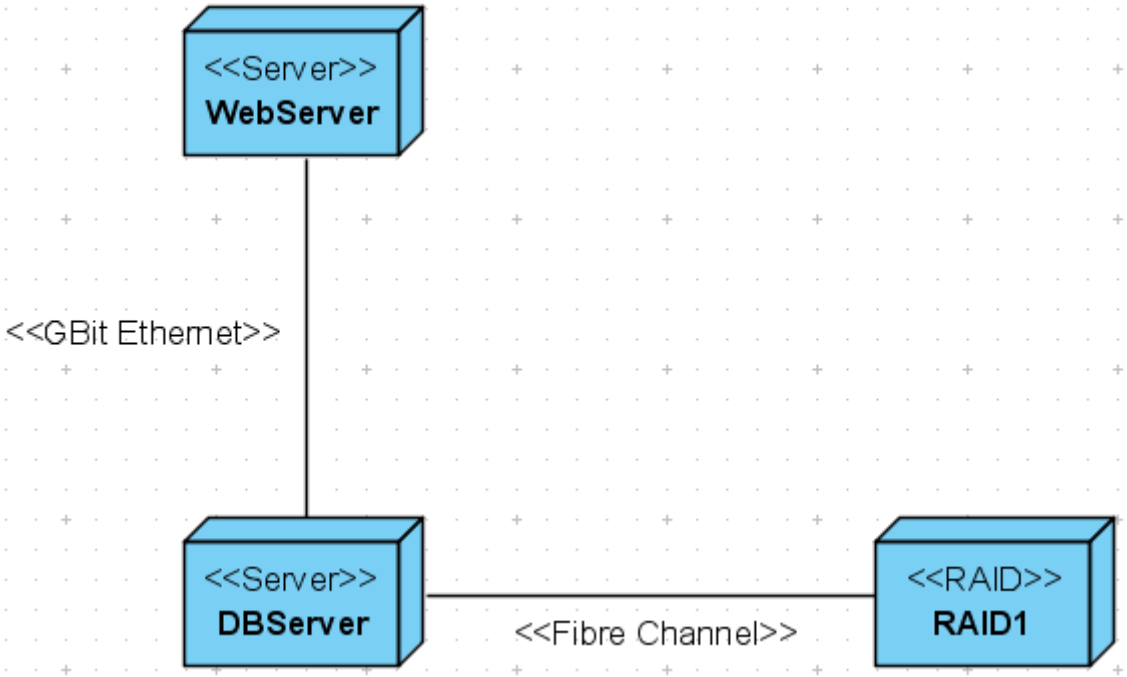
- Opisują sprzętowe elementy systemu i wiążą je z uprzednio opisanymi elementami
- Stosują pojęcie artefaktu
- Artefakt jest dowolnym elementem systemu. Artefakt jest elementem złożonym z ciągu bitów. Może być to klasa, plik, diagram, model, baza danych, dokument itd.
- Artefakty są umieszczane w węzłach
- Węzły mogą być połączone

Węzły

- Reprezentują fizyczne bądź logiczne jednostki mogące “przechowywać” artefakty
- Są dwa różne typy węzłów:
  - Urządzenia. Są to materialne elementy, takie jak komputery, drukarki, switche itp.
  - Środowiska wykonania. Są to systemy programowe, takie jak systemy operacyjne, systemy baz danych itp.
- Powyższe typy są rozróżniane poprzez stereotypy. Dodatkowe stereotypy mogą podawać bardziej precyzyjne informacje, np. serwer, drukarka, Debian itp.

Ścieżki komunikacyjne

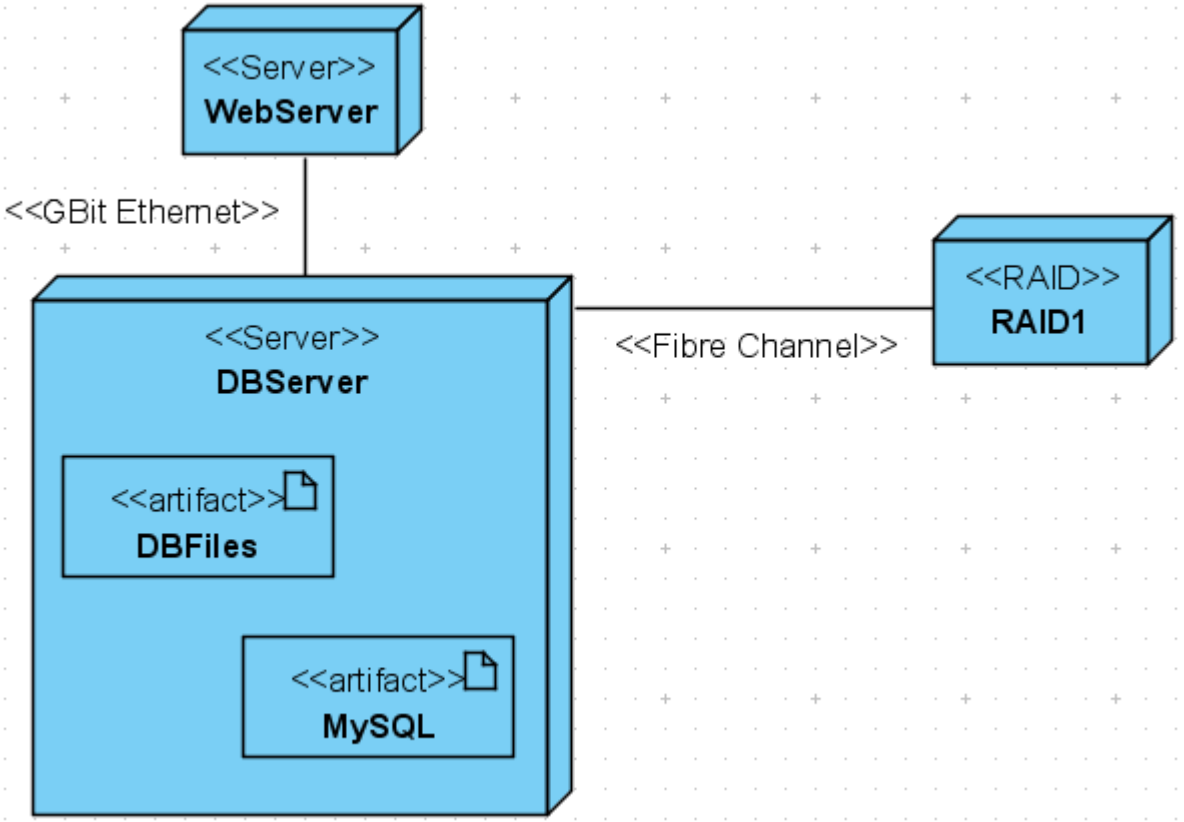
- Reprezentują połączenia między węzłami
- Często są to połączenia sieciowe
- Są obrazowane przez asocjacje, często stereotypowane



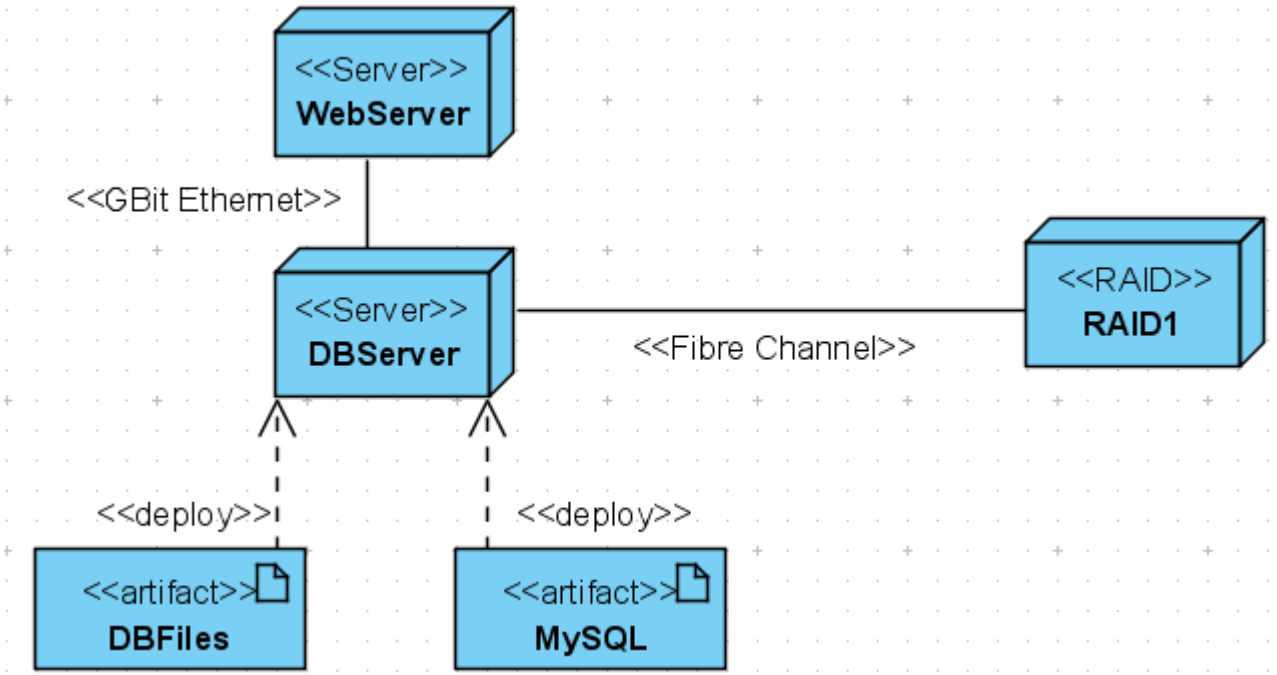
Artefakty w węzłach

- Artefakty są umieszczane w odpowiednich węzłach
- Fakt ten może być zobrazowany na dwa sposoby:
  - Poprzez umieszczenie symbolu artefaktu wewnątrz symbolu węzła
  - Poprzez połączenie symboli poprzez zależność ze stereotypem <<deploy>>

Przykładowy diagram



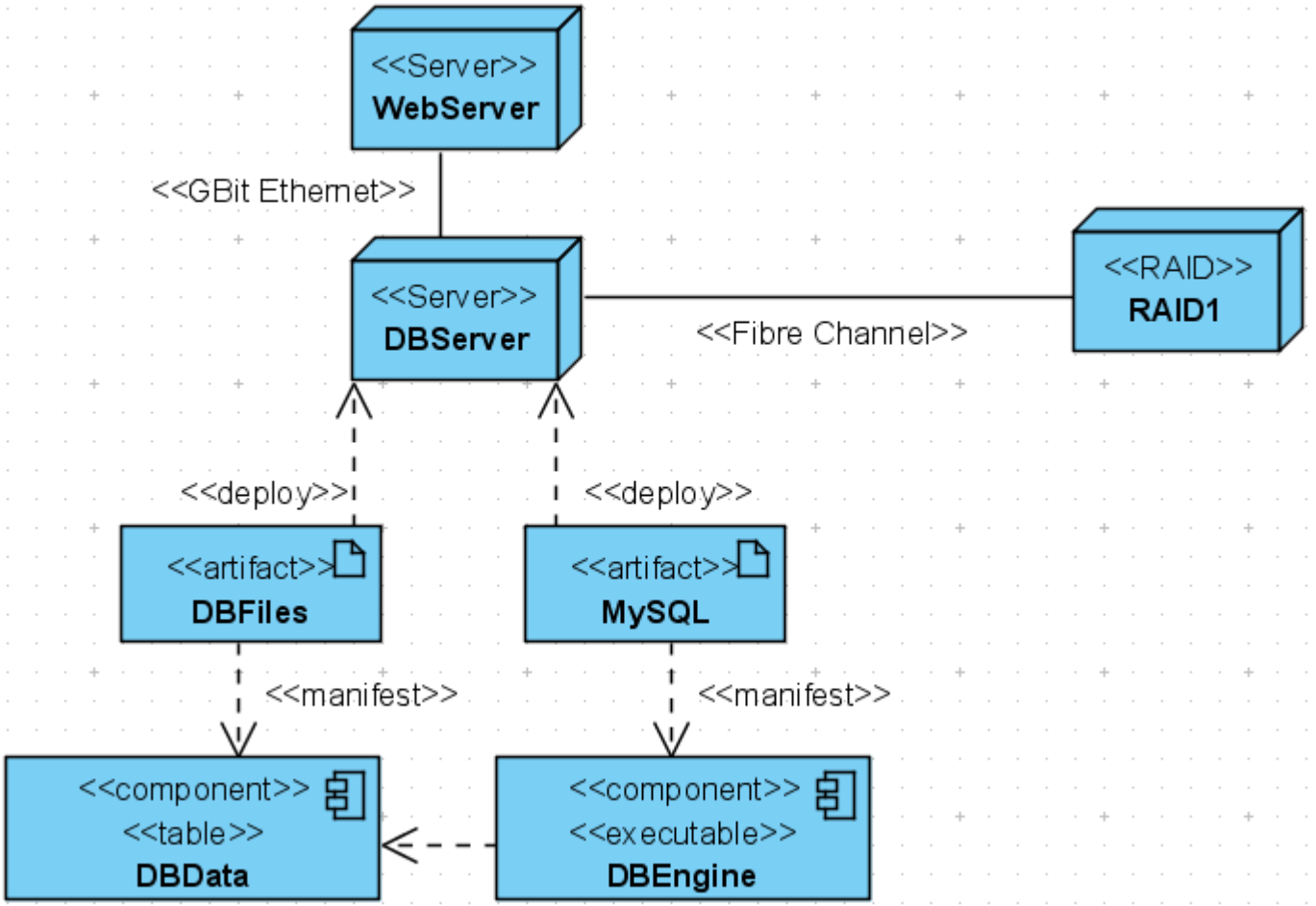
Przykładowy diagram



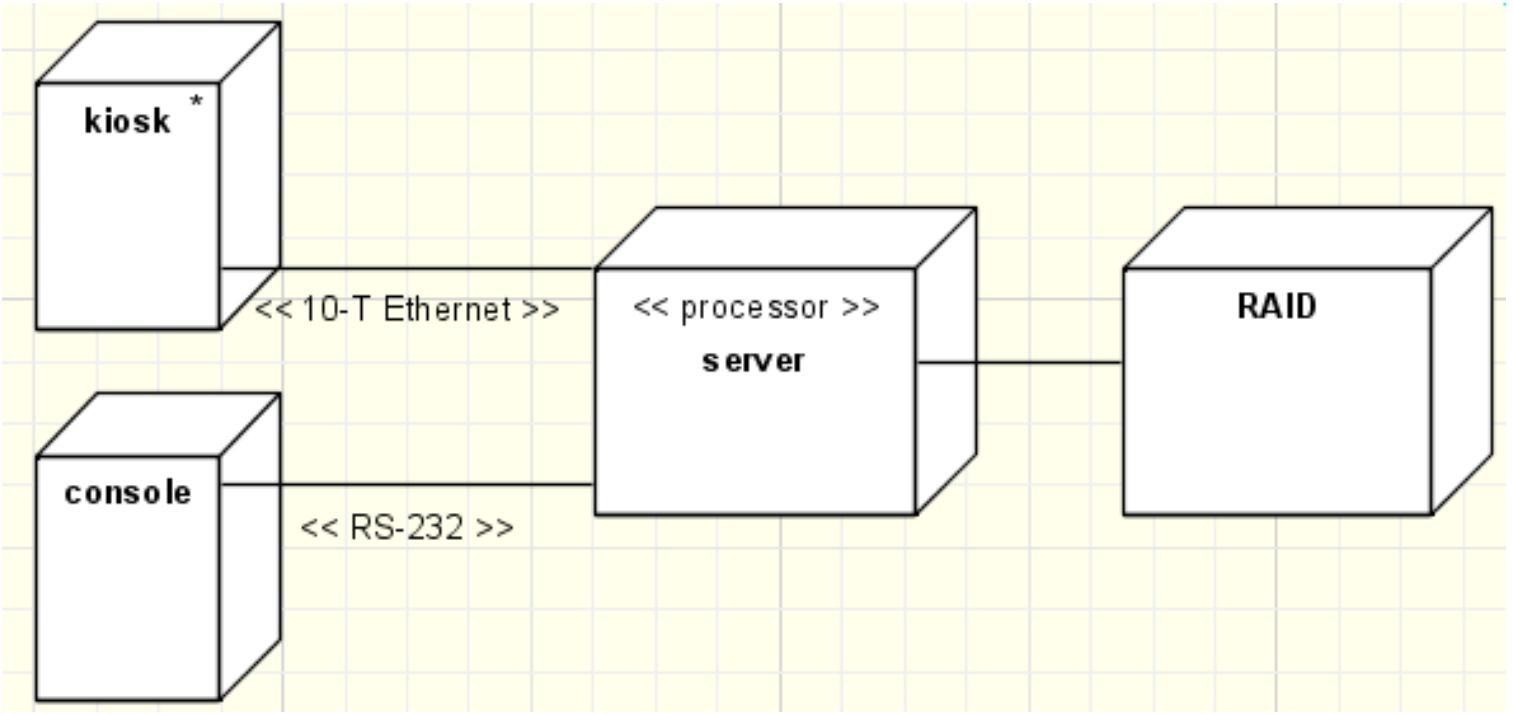
Manifestacja

- Artefakty mogą manifestować komponenty, czyli pokazywać, że posiadają dany komponent i w związku z tym jest on dostępny w węźle w którym zostały umieszczone
- Jest to obrazowane zależnością ze stereotypem <<manifest>>

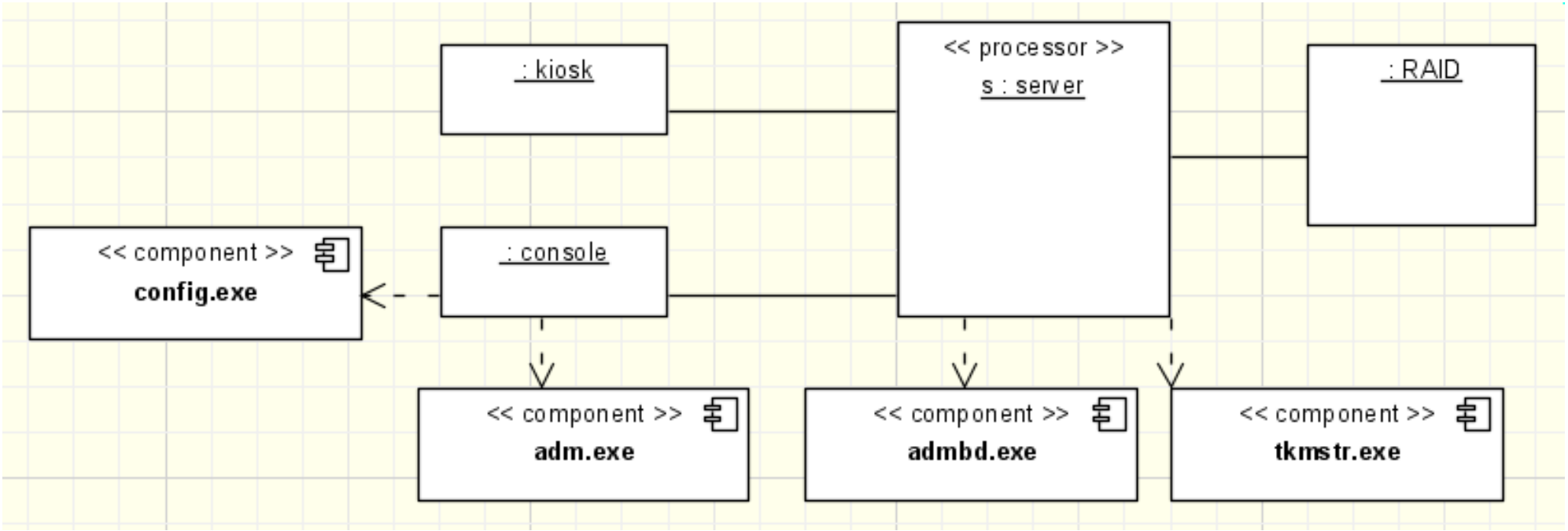
Przykładowa manifestacja



Przykładowy diagram



Przykładowy diagram



Przykładowy diagram

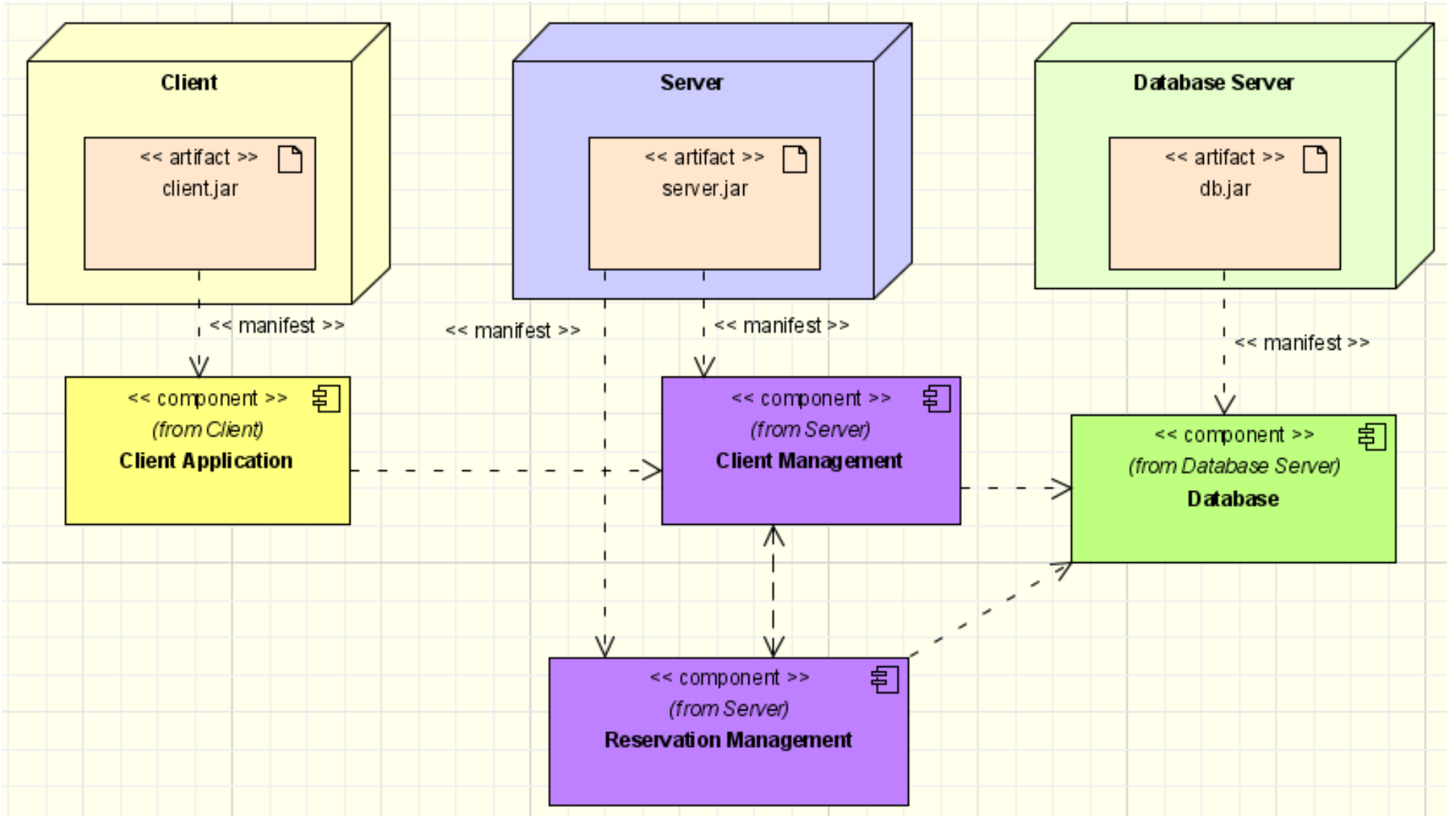


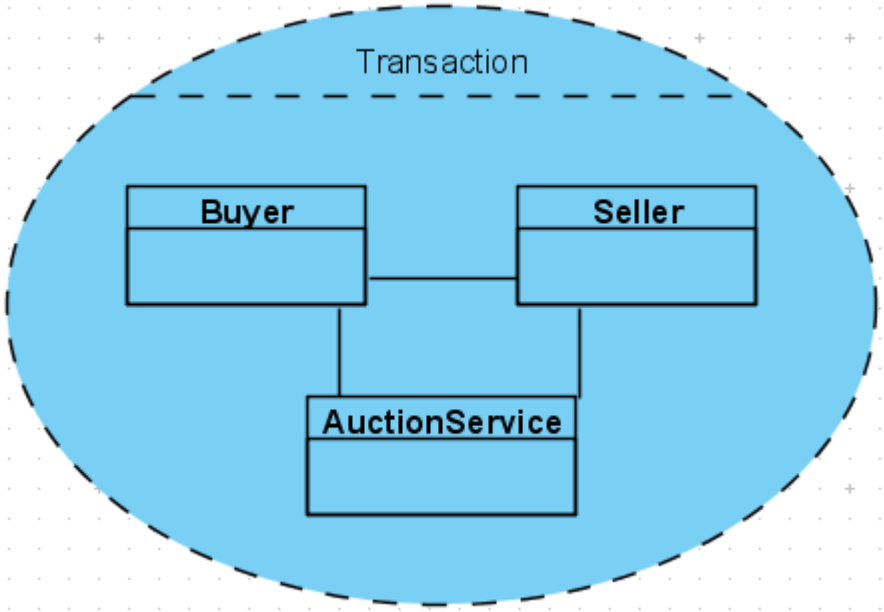
Diagram struktur połączonych (composite structure)

- Umożliwia pokazanie współpracy między klasyfikatorami
- Istotnym elementem jest symbol współpracy, owal z przerywaną linią
- Współpraca przedstawia klasyfikatory które “pracują” wspólnie z jakiegoś powodu

Diagram struktur połączonych

- Klasyfikatory mogą być umieszczone wewnątrz owalu współpracy
- Alternatywnie, mogą być umieszczone poza owalem i połączone z nim asocjacjami
- Asocjacje mogą być nazwane aby odzwierciedlić rolę klasyfikatorów we współpracy

Diagram struktur połączonych



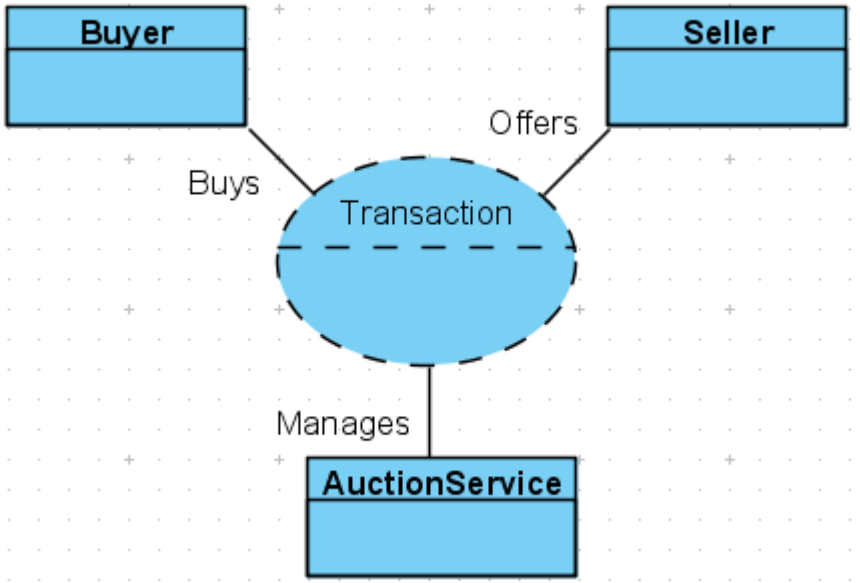


Diagram pakietów

- Umożliwia porządkowanie i grupowanie elementów modeli
- Główne elementy to pakiety, zależności i zawieranie pakietów
- Każdy element modelu może być umieszczony w pakiecie
- Nazwa pakietu jest umieszczona w symbolu pakietu, w zakładce (jeśli elementy są umieszczone w symbolu pakietu) lub w zakładce ramy (jeśli rama jest stosowana jako oznaczenie pakietu)

Diagram pakietów

- Pakiety mogą być łączone przez zależności
- Są trzy możliwości, opisane przez stereotypy
  - Import. Oznacza, że elementy pakietu wskazywanego mogą być bezpośrednio użyte w pakiecie wskazującym
  - Merge. Oznacza, że pomiędzy elementami pakietów które mają tę samą nazwę zachodzi generalizacja (dziedziczenie)
  - Access. Tak jak import, ale nazwy należy poprzedzać nazwą pakietu

Diagram pakietów

- Pakiety mogą zawierać inne pakiety
- Można to zobrazować poprzez umieszczenie pakietu w pakiecie, lub poprzez związek zawierania
- Pakiety mogą być stereotypowane, typowe stereotypy to <<model>>, <<subsystem>>, <<framework>>
- Można określić poziom dostępu do elementów pakietu

Diagram pakietów

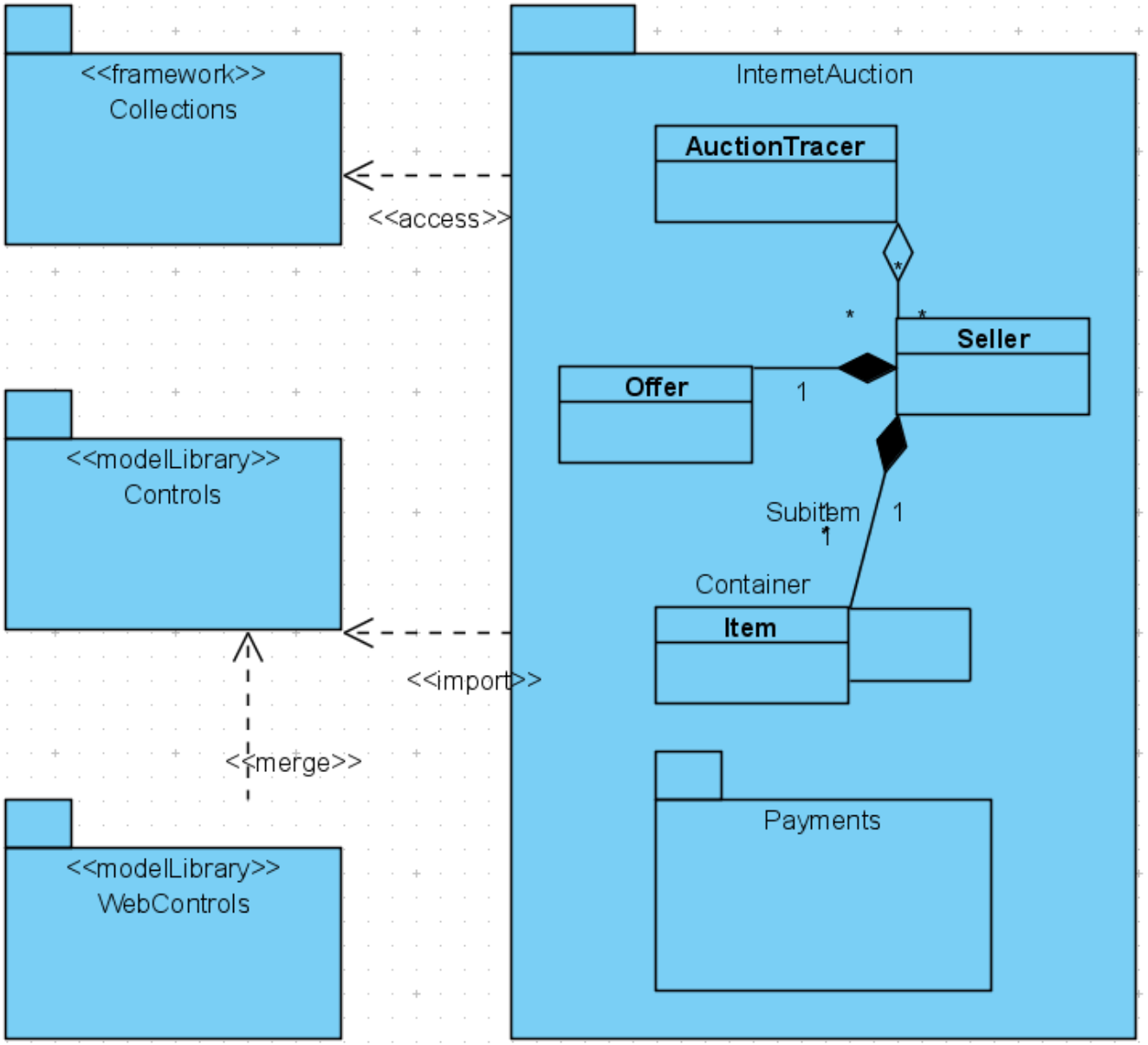


Diagram pakietów

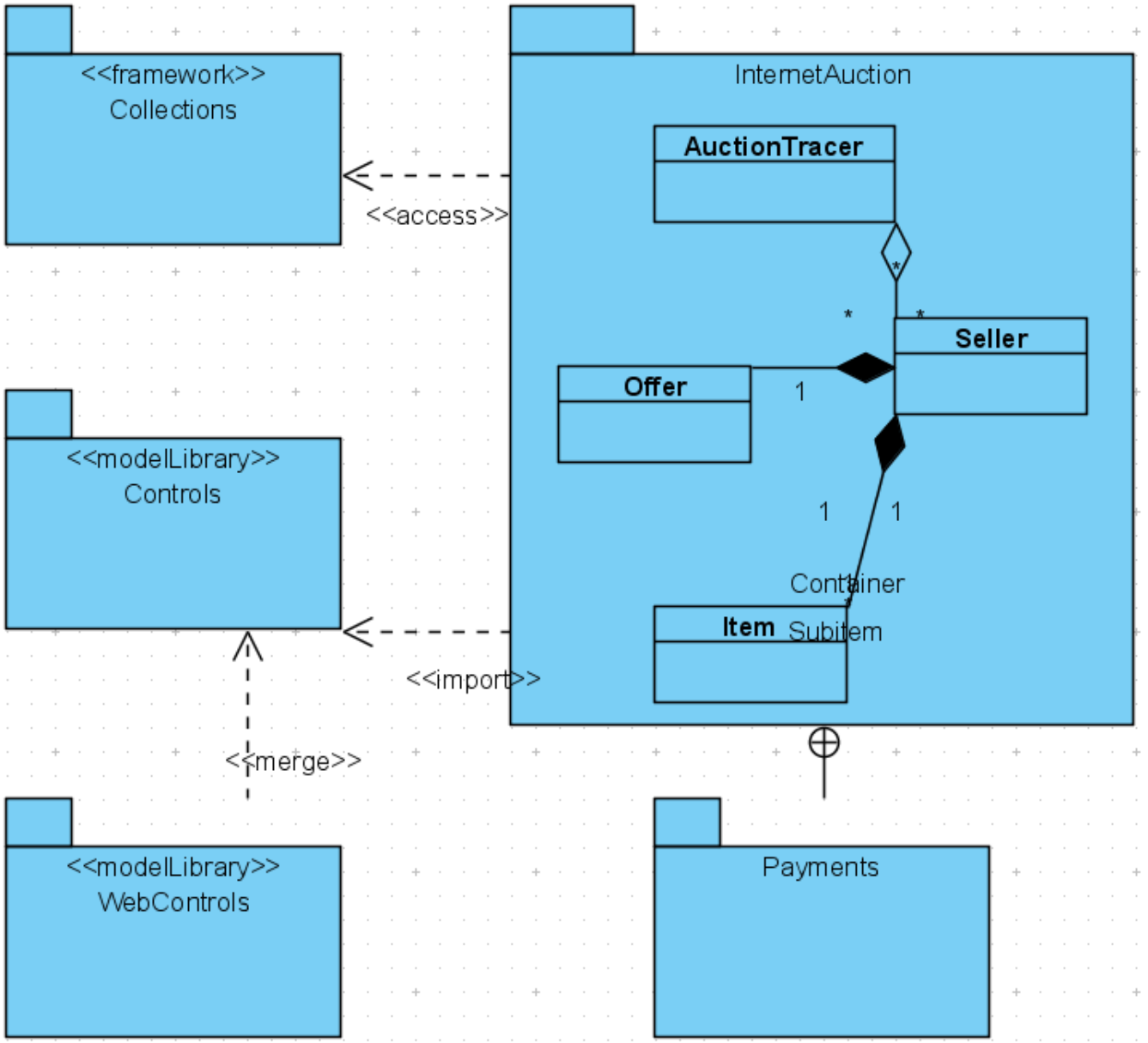
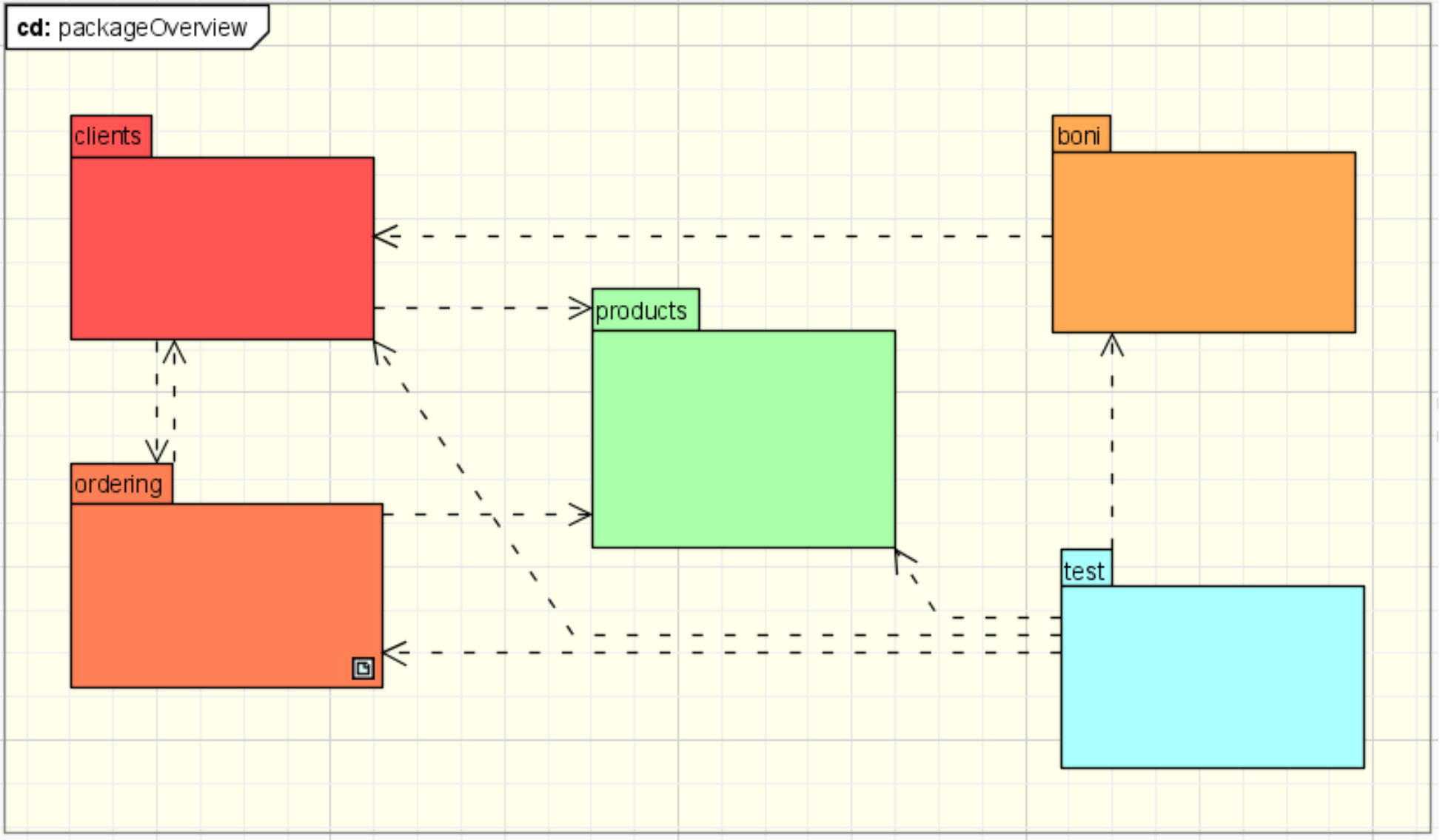


Diagram pakietów



Koncepcje specyficzne dla modelowania biznesowego

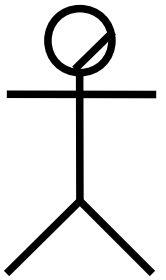
- Modelowanie biznesowe jest zwykle pierwszym krokiem w modelowaniu systemu
- Umożliwia uchwycenie najbardziej ogólnych elementów i zależności w systemie
- Jeśli docelowym produktem ma być system informatyczny, model biznesowy jest stopniowo przekształcany w model systemowy, a ten następnie w system informatyczny

Diagramy w modelowaniu biznesowym

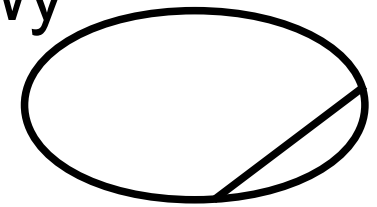
- W modelowaniu biznesowym używa się podzbioru diagramów języka UML
- Najczęściej stosowane są diagramy:
  - Przypadków użycia
  - Klas
  - Czynności
  - Sekwencji
  - Pakietów

Kategorie modelowania specyficzne dla diagramów biznesowych

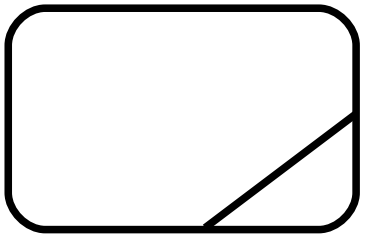
- Z uwagi na specyfikę modelowania biznesowego stosuje się modyfikacje istniejących kategorii modelowania oraz wprowadza się nowe kategorie
- Modyfikacje:
  - Aktor biznesowy – użytkownik działający w otoczeniu organizacji



  - Biznesowy przypadek użycia – proces biznesowy w interakcję z którym wchodzi aktor biznesowy

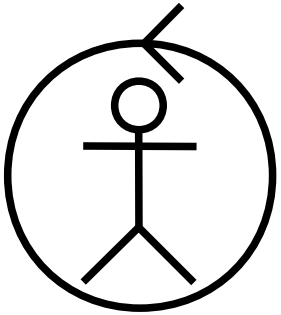


  - Czynność biznesowa

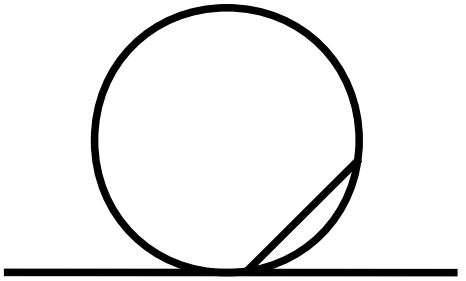


Kategorie modelowania specyficzne dla diagramów biznesowych

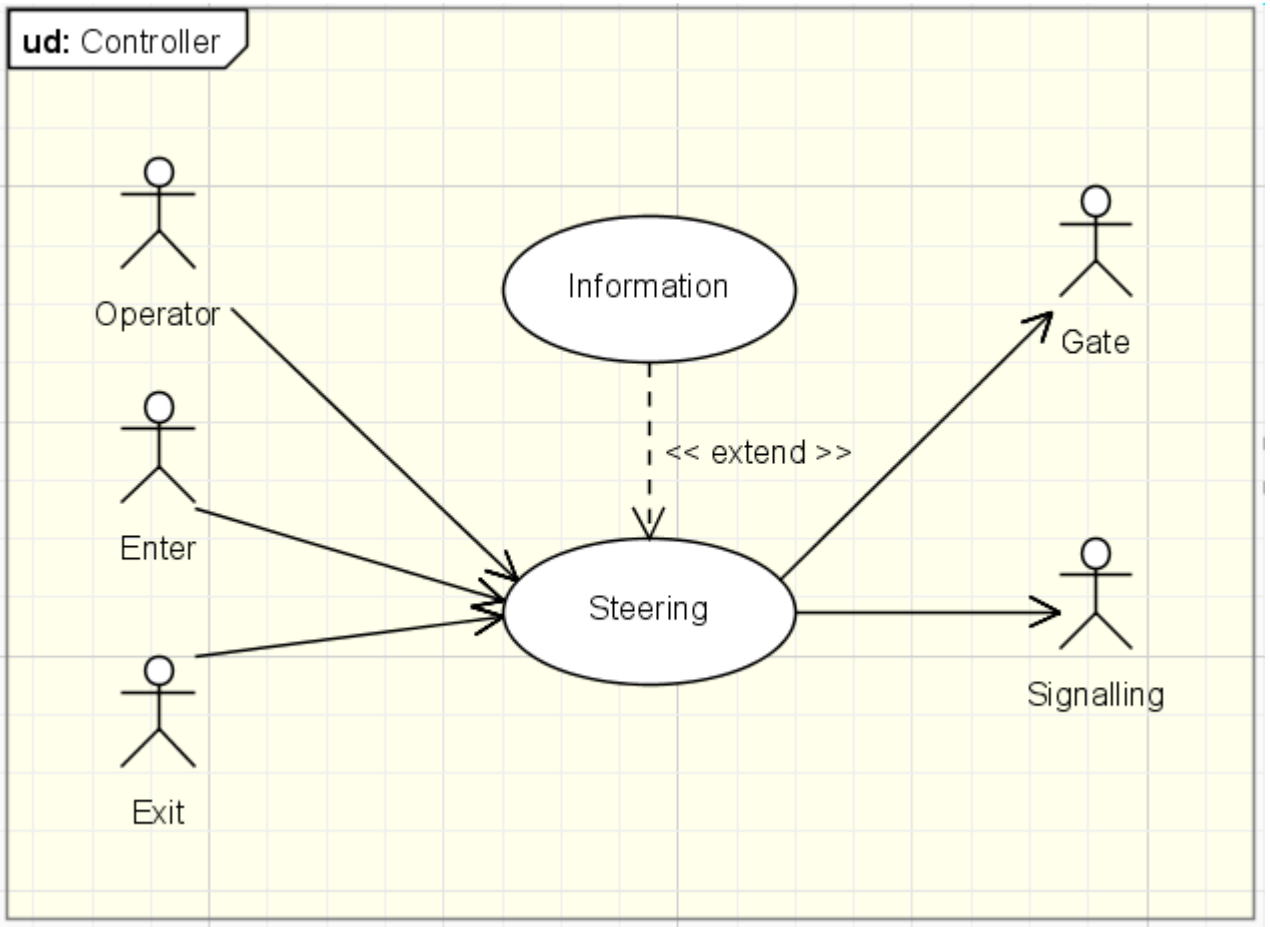
- Nowe kategorie:
  - Pracownik biznesowy – pracownik lub system funkcjonujący w ramach organizacji, współpracujący z innymi pracownikami biznesowymi i wykonujący operację na obiektach klas przechowujących



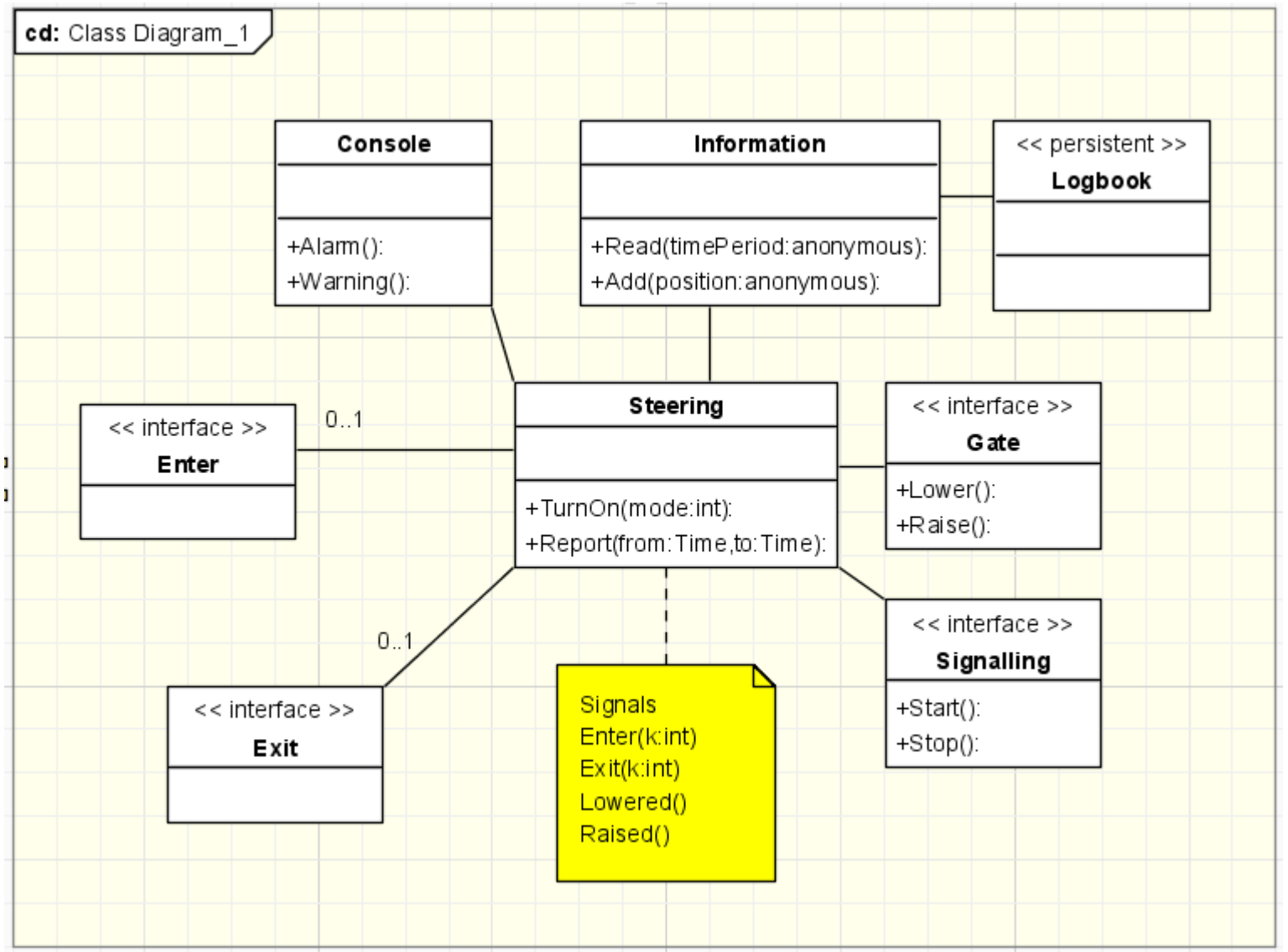
  - Biznesowa klasa przechowująca – byt użytkowany i przetwarzany przez pracowników biznesowych. Operacje obejmują dostęp, aktualizację, tworzenie, monitorowanie, usuwanie



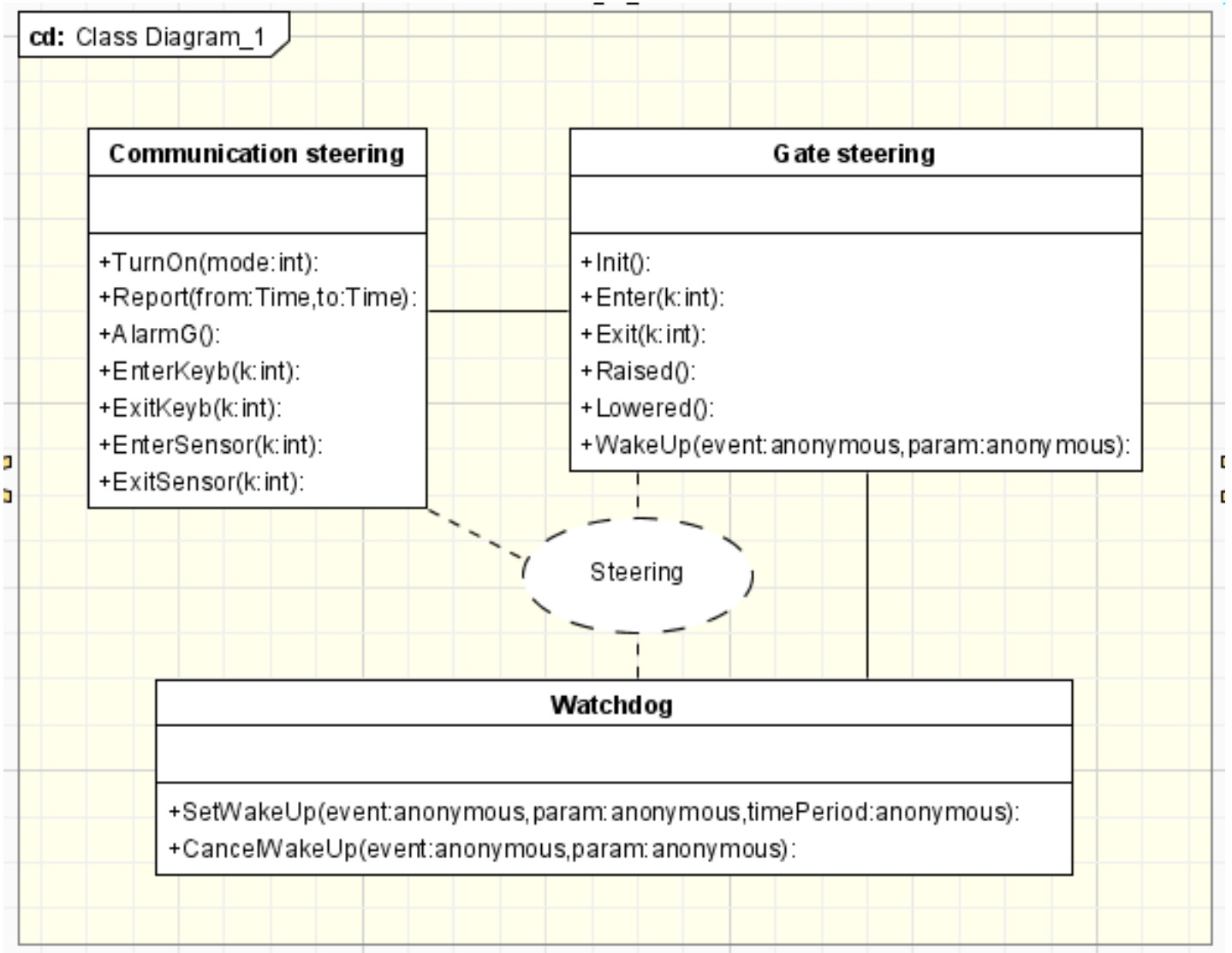
Przejazd kolejowy



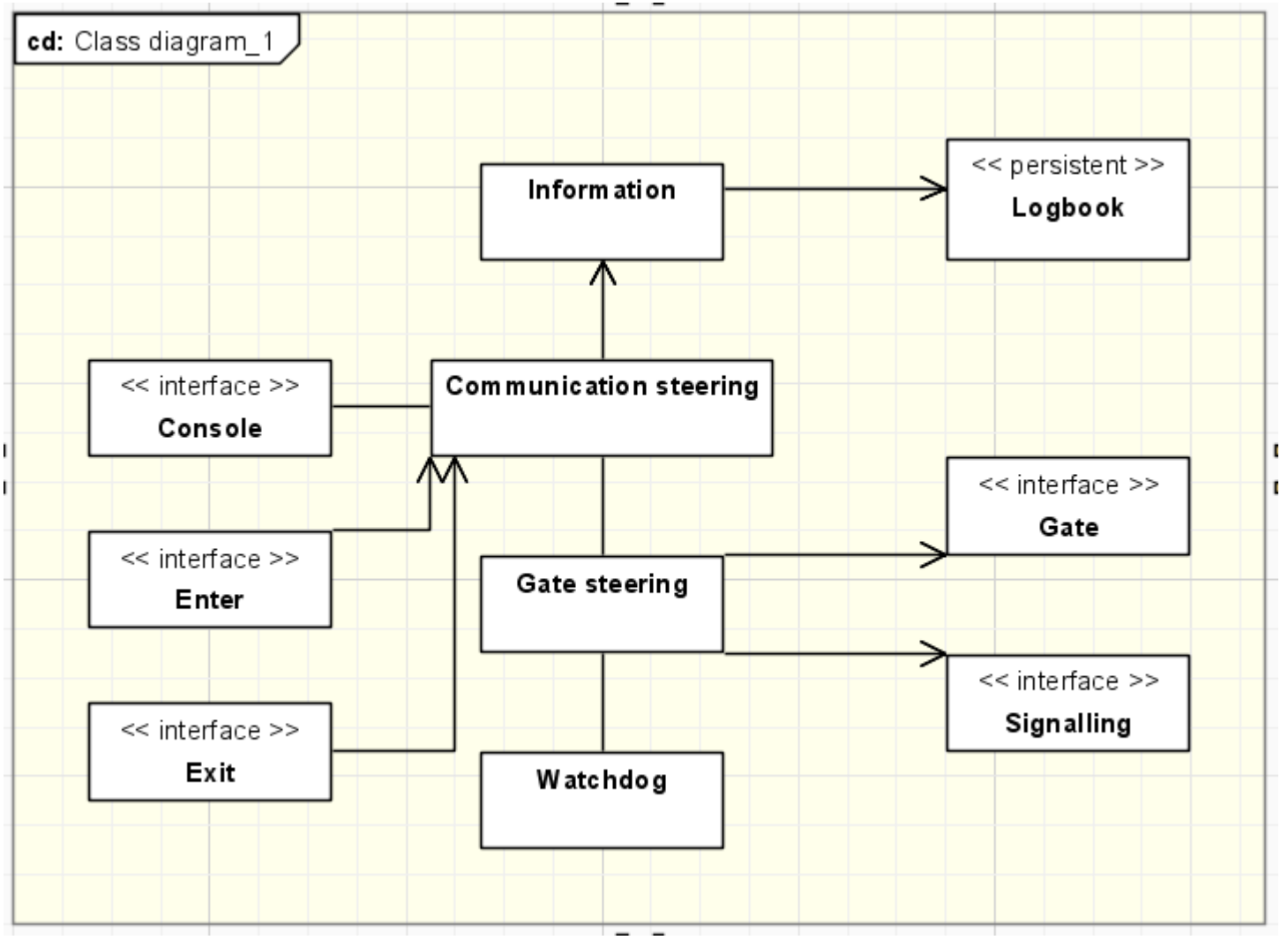
Przejazd kolejowy



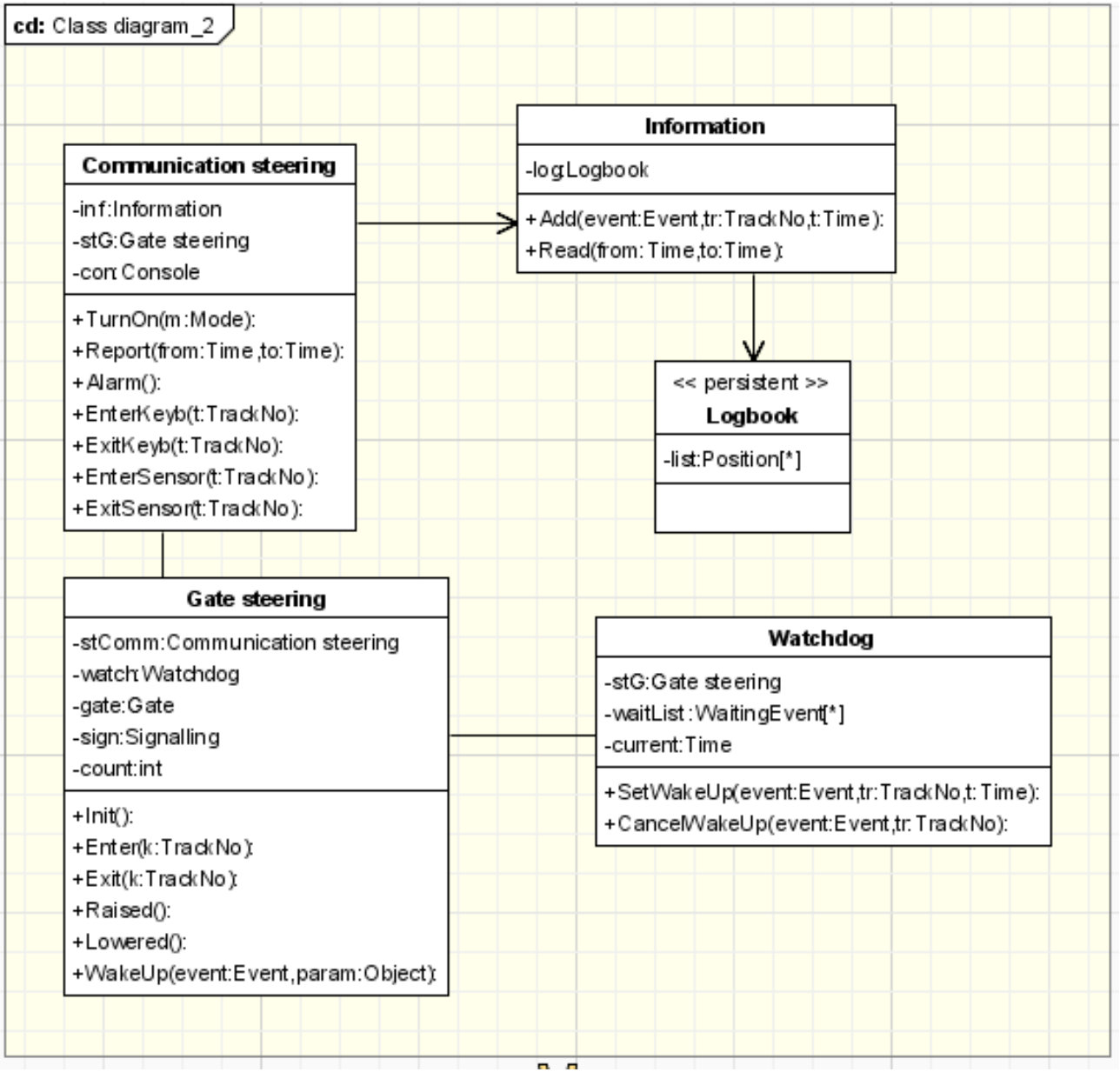
Przejazd kolejowy



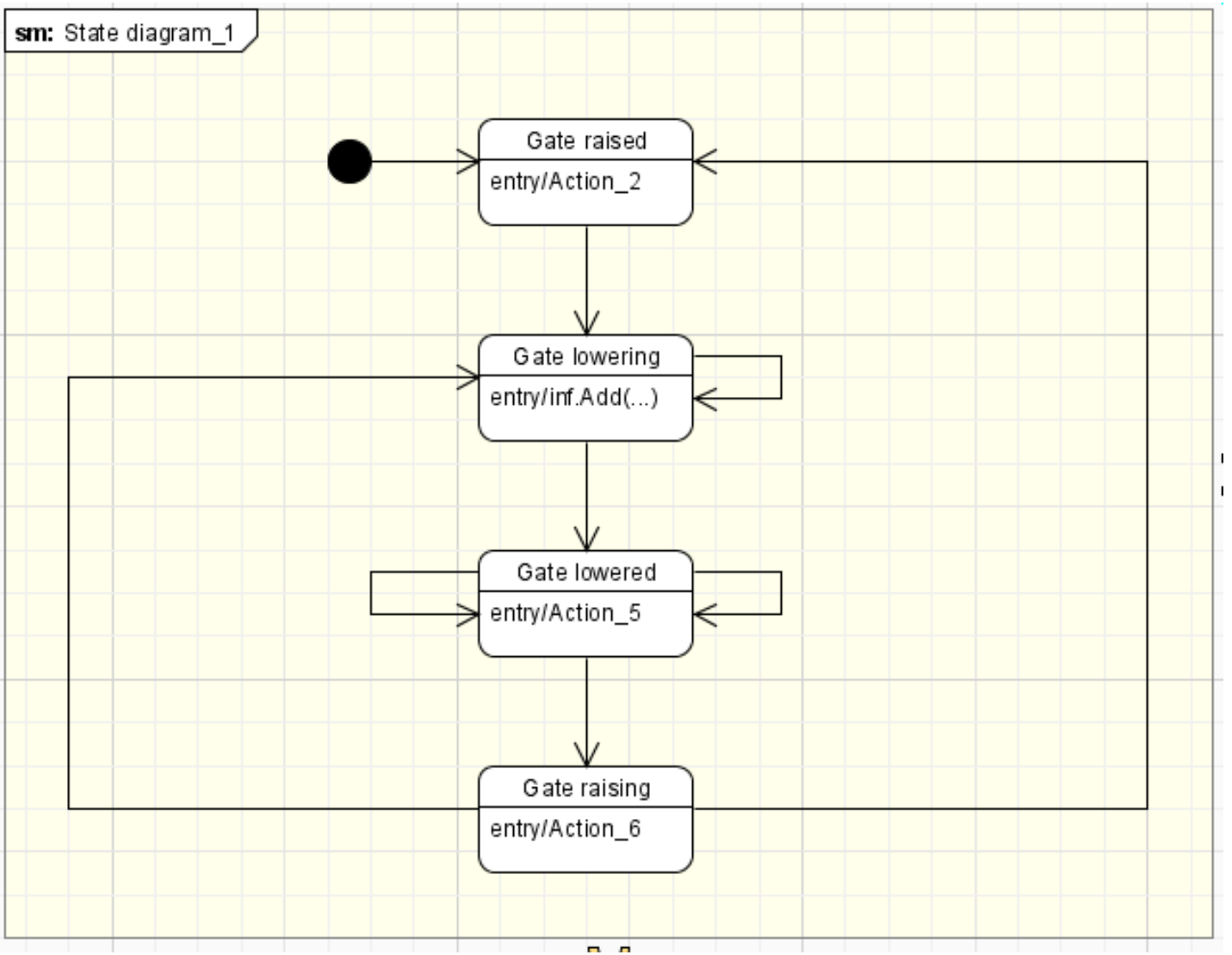
Przejazd kolejowy



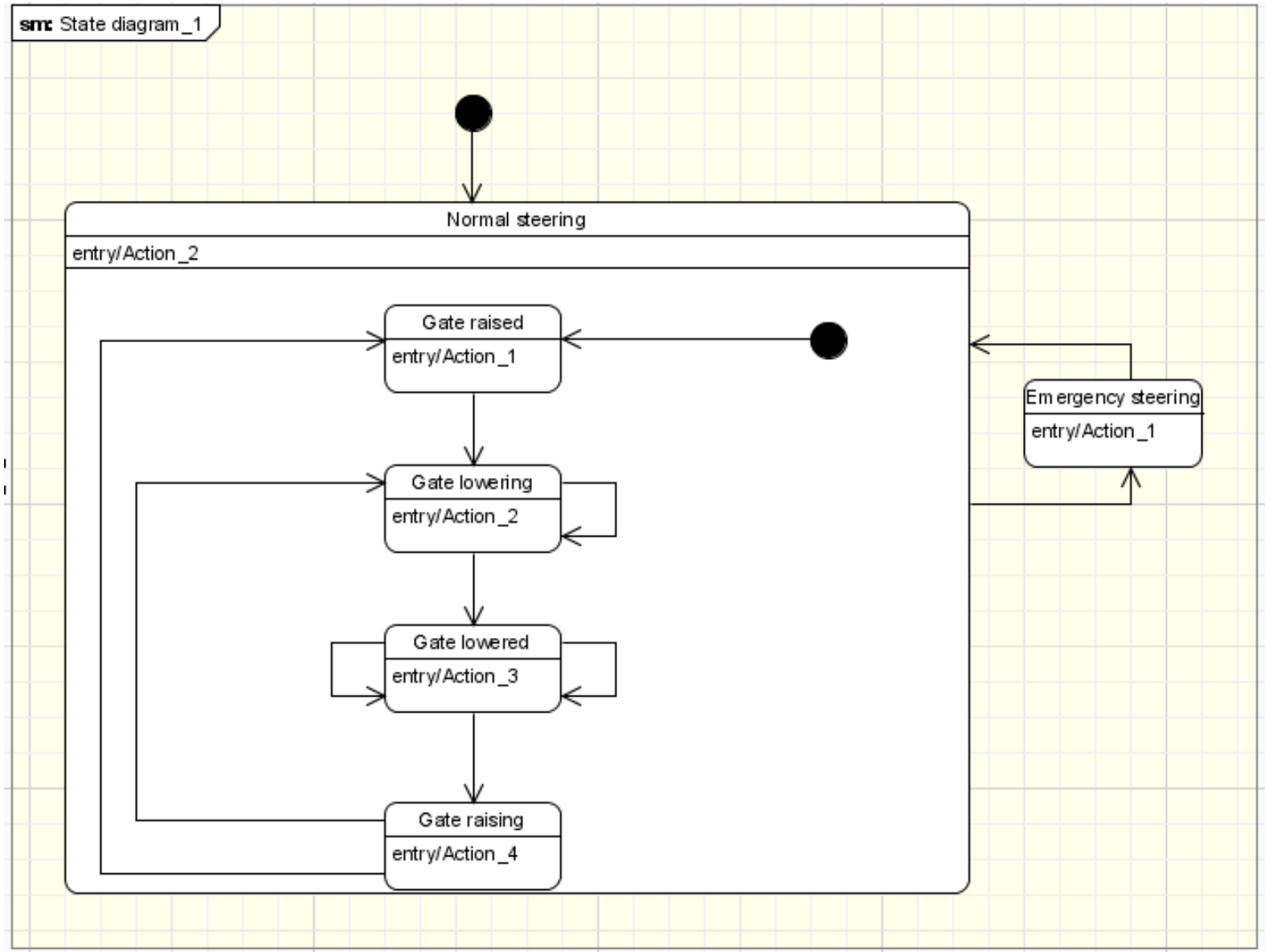
Przejazd kolejowy



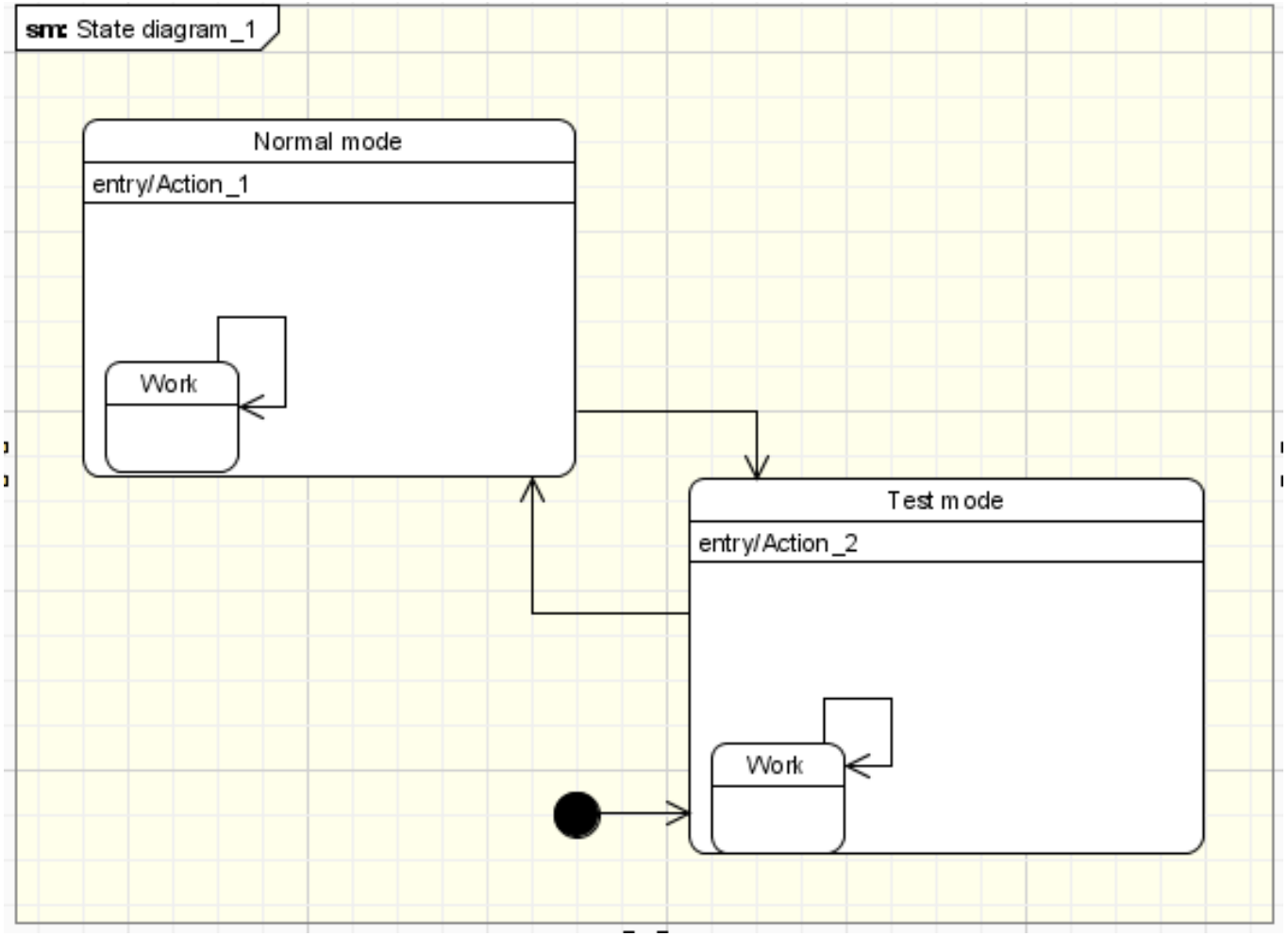
Przejazd kolejowy



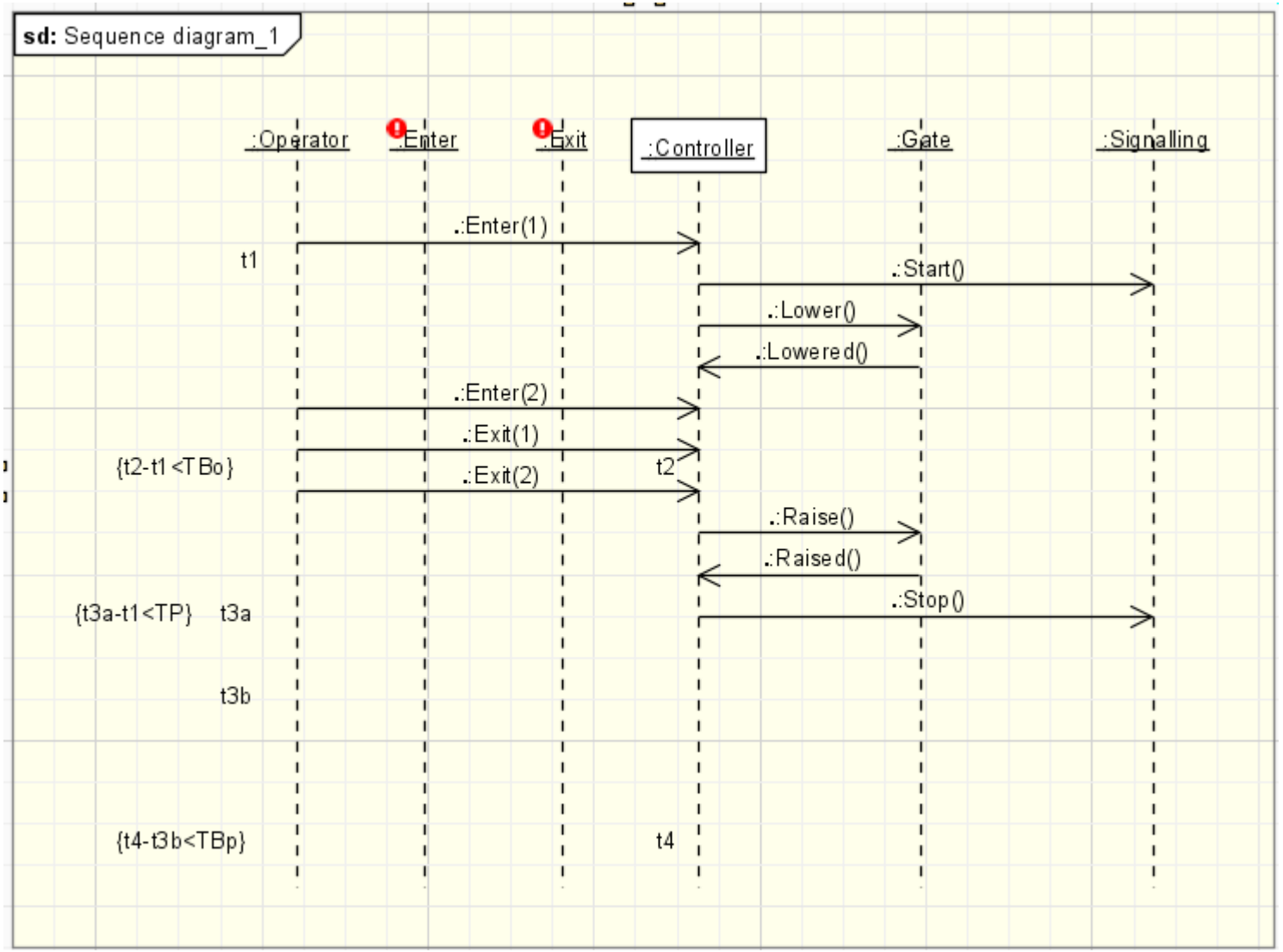
Przejazd kolejowy



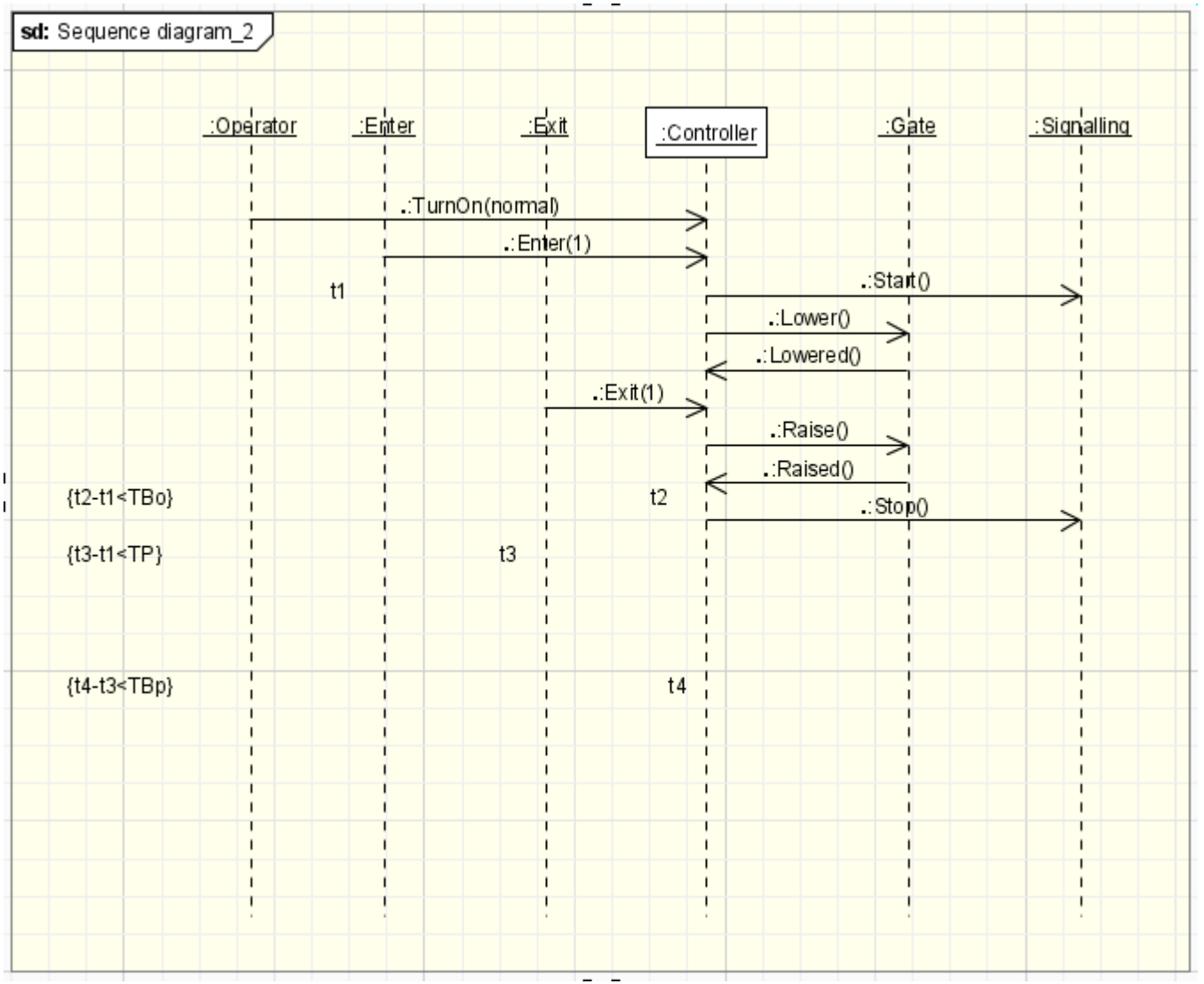
Przejazd kolejowy



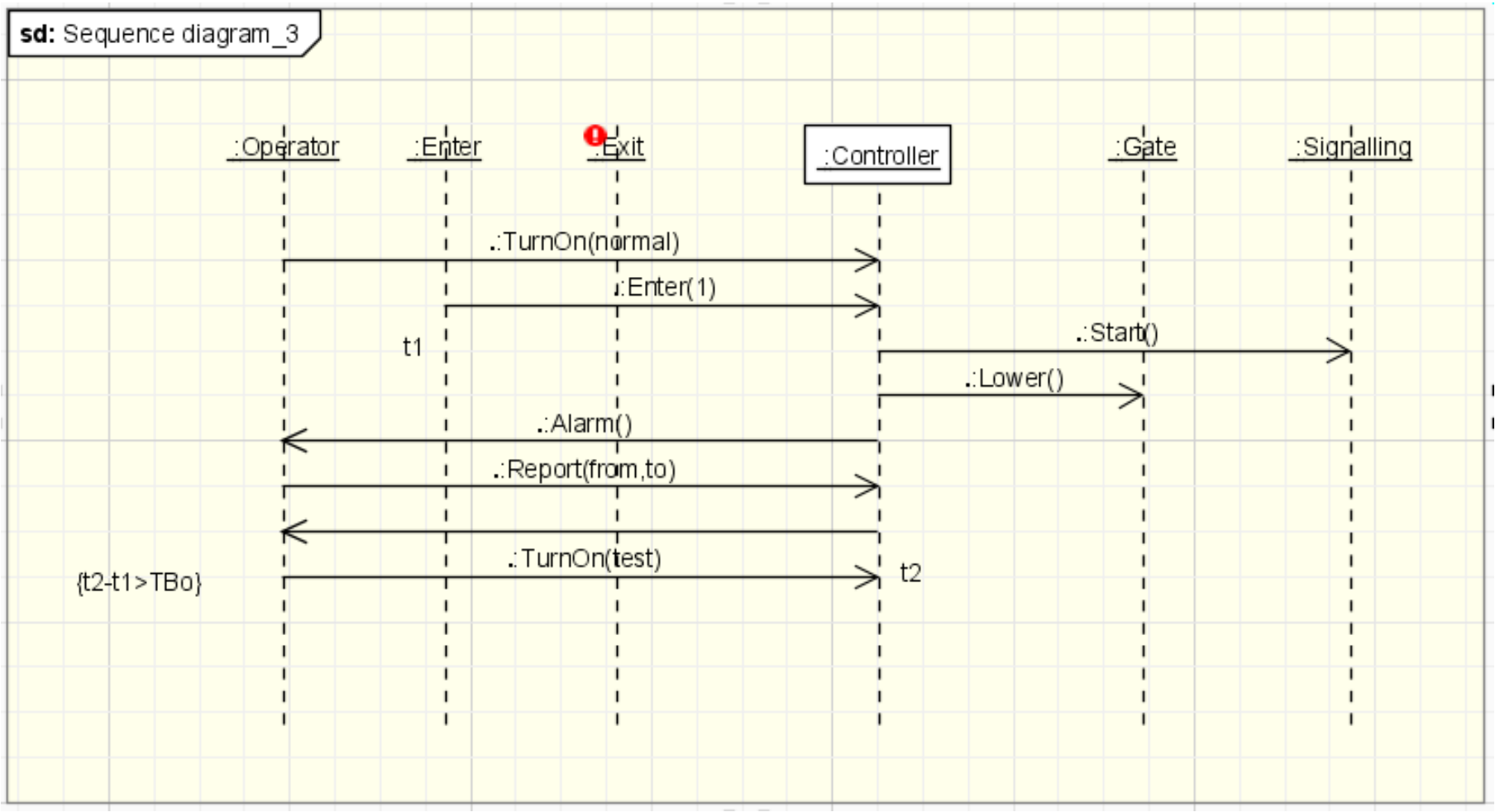
Przejazd kolejowy



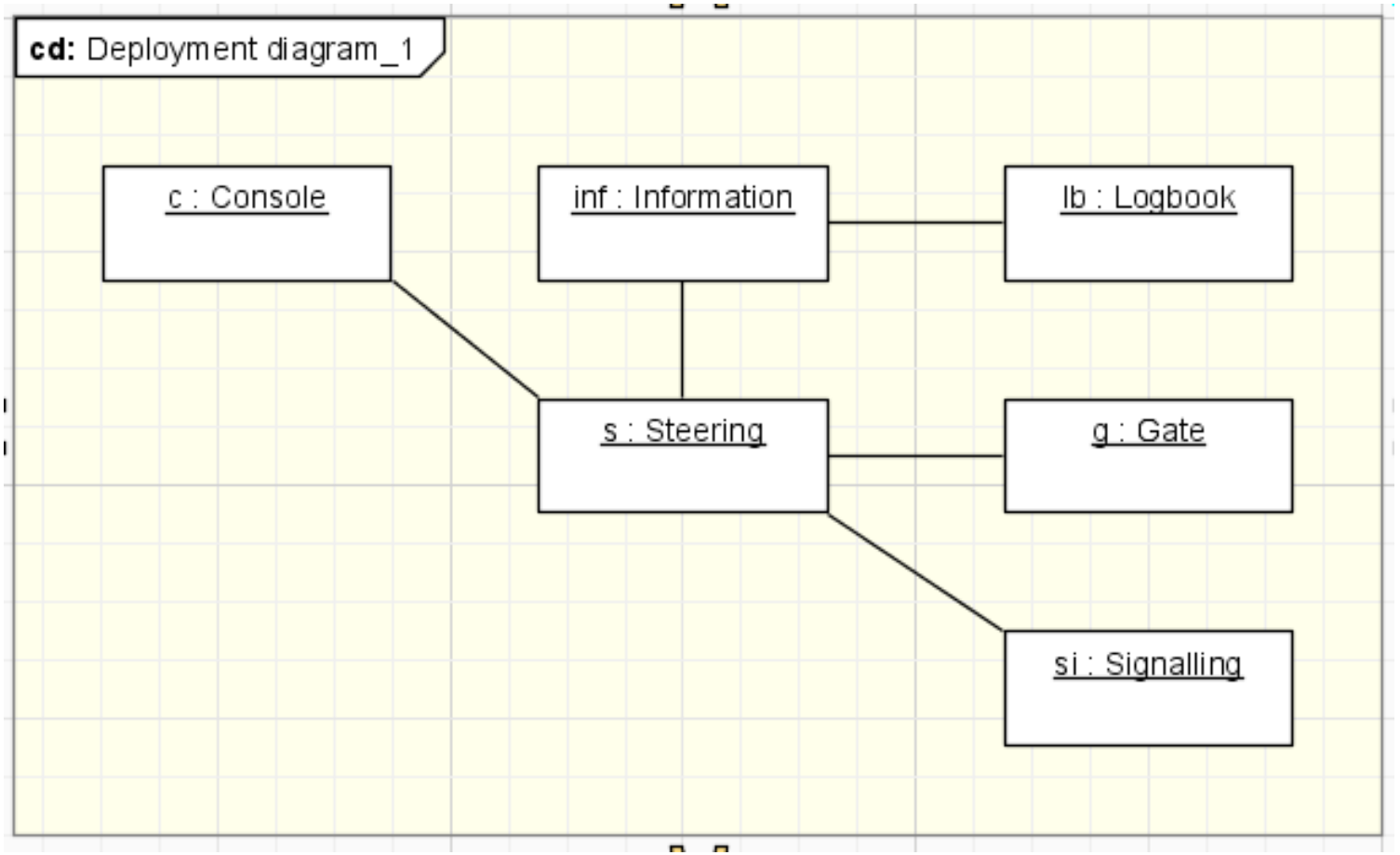
Przejazd kolejowy



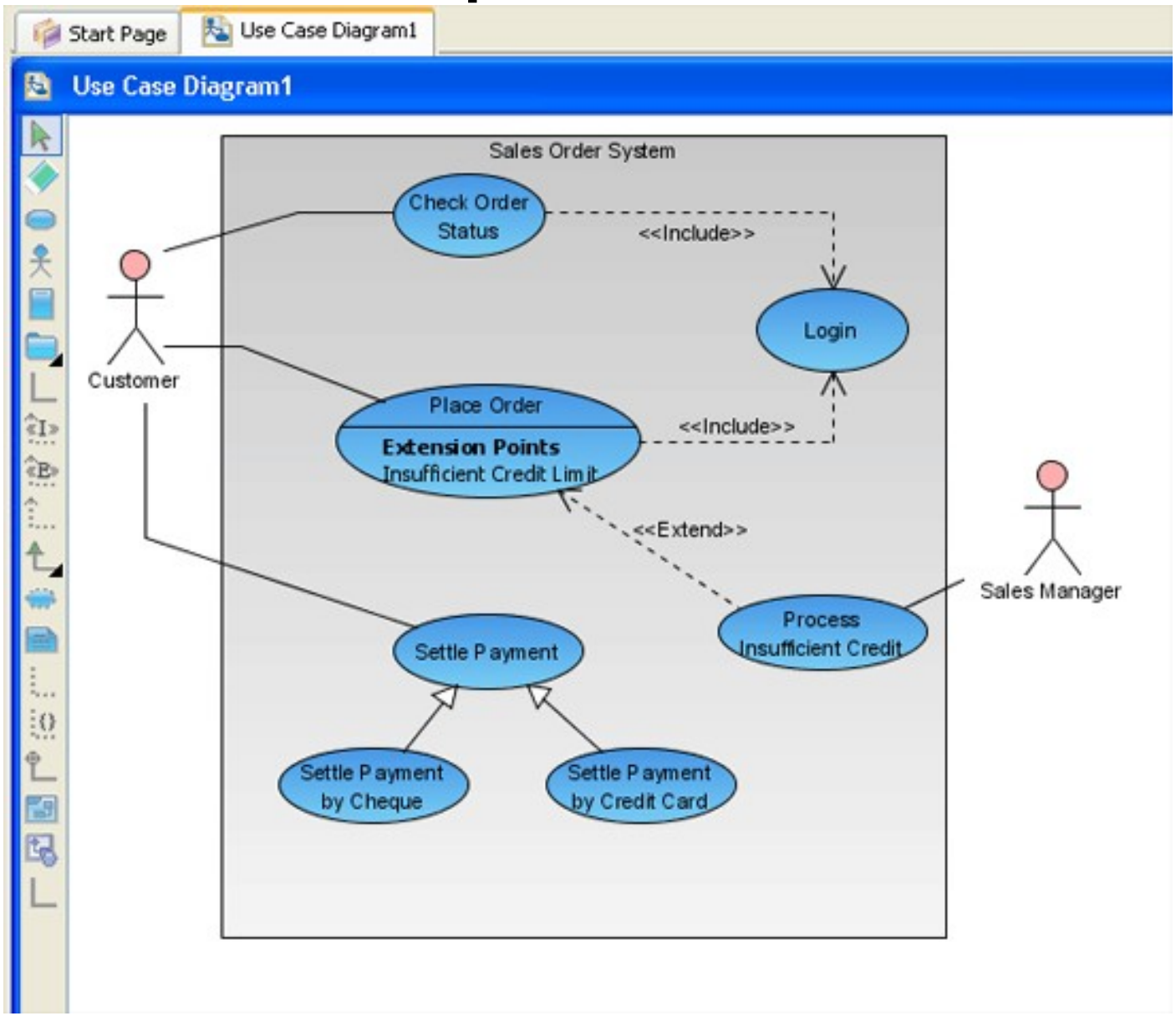
Przejazd kolejowy



Przejazd kolejowy

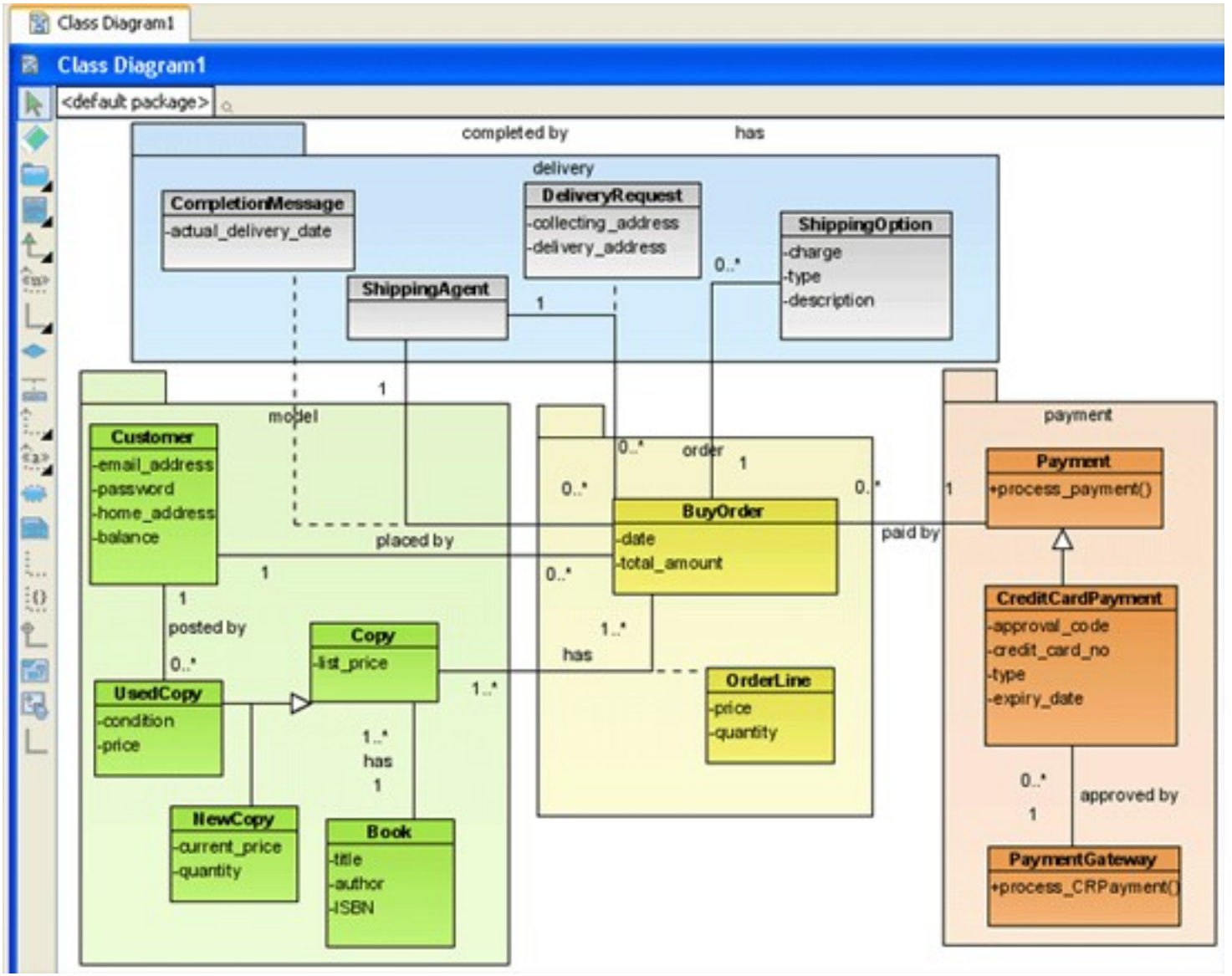


Sprzedaż



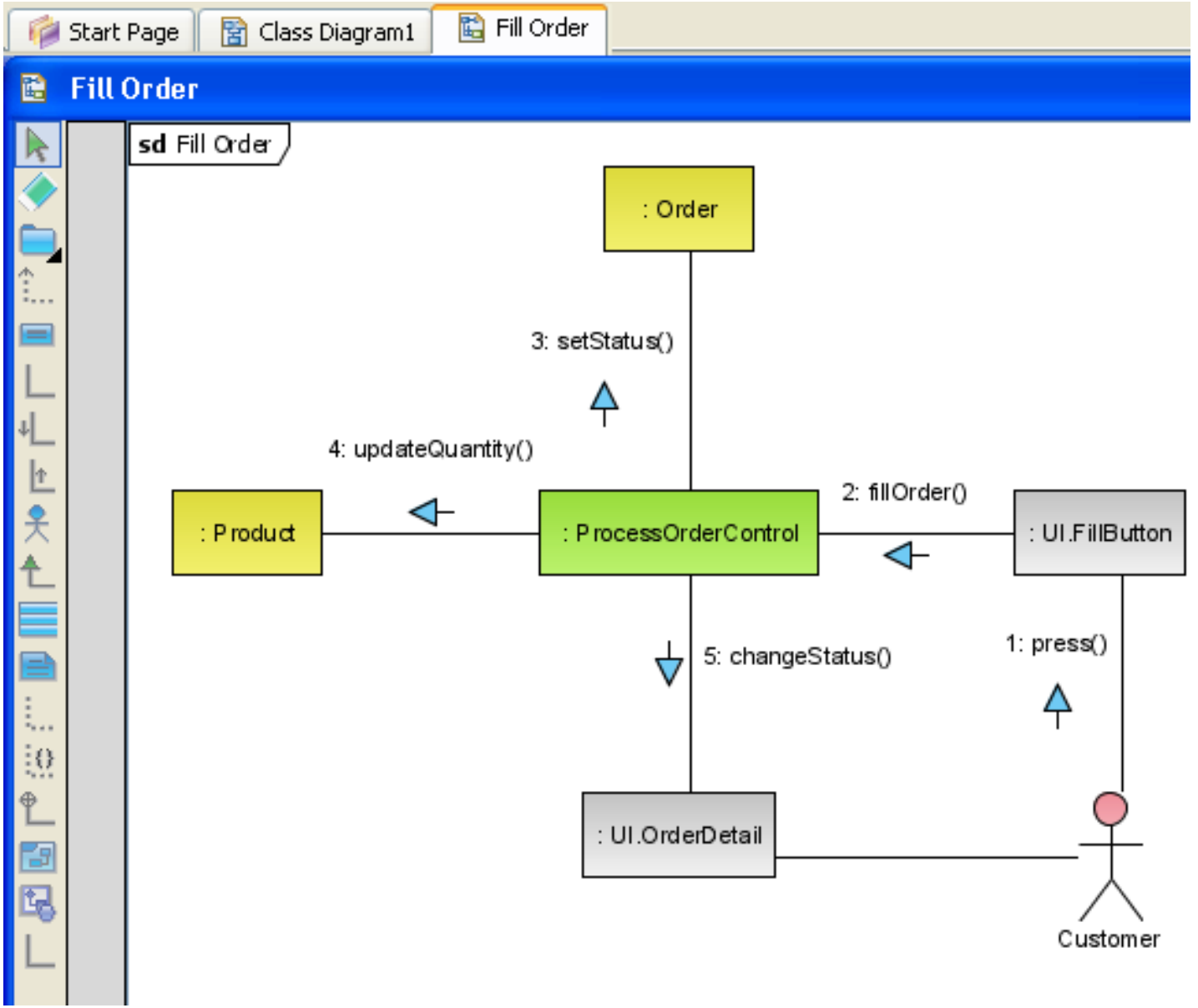
http://www.visual-paradigm.com/VPGallery/diagrams/index.html

Sprzedaż



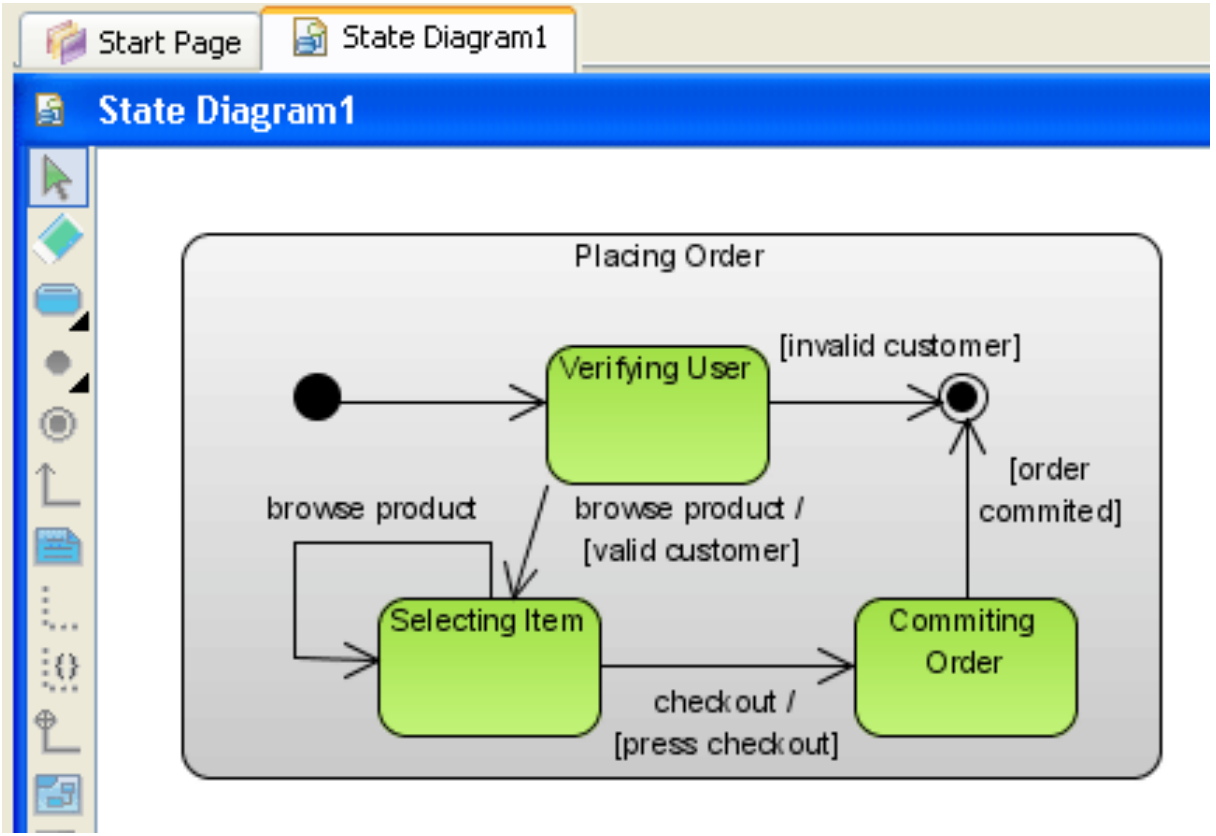
http://www.visual-paradigm.com/VPGallery/diagrams/index.html

Zamówienie



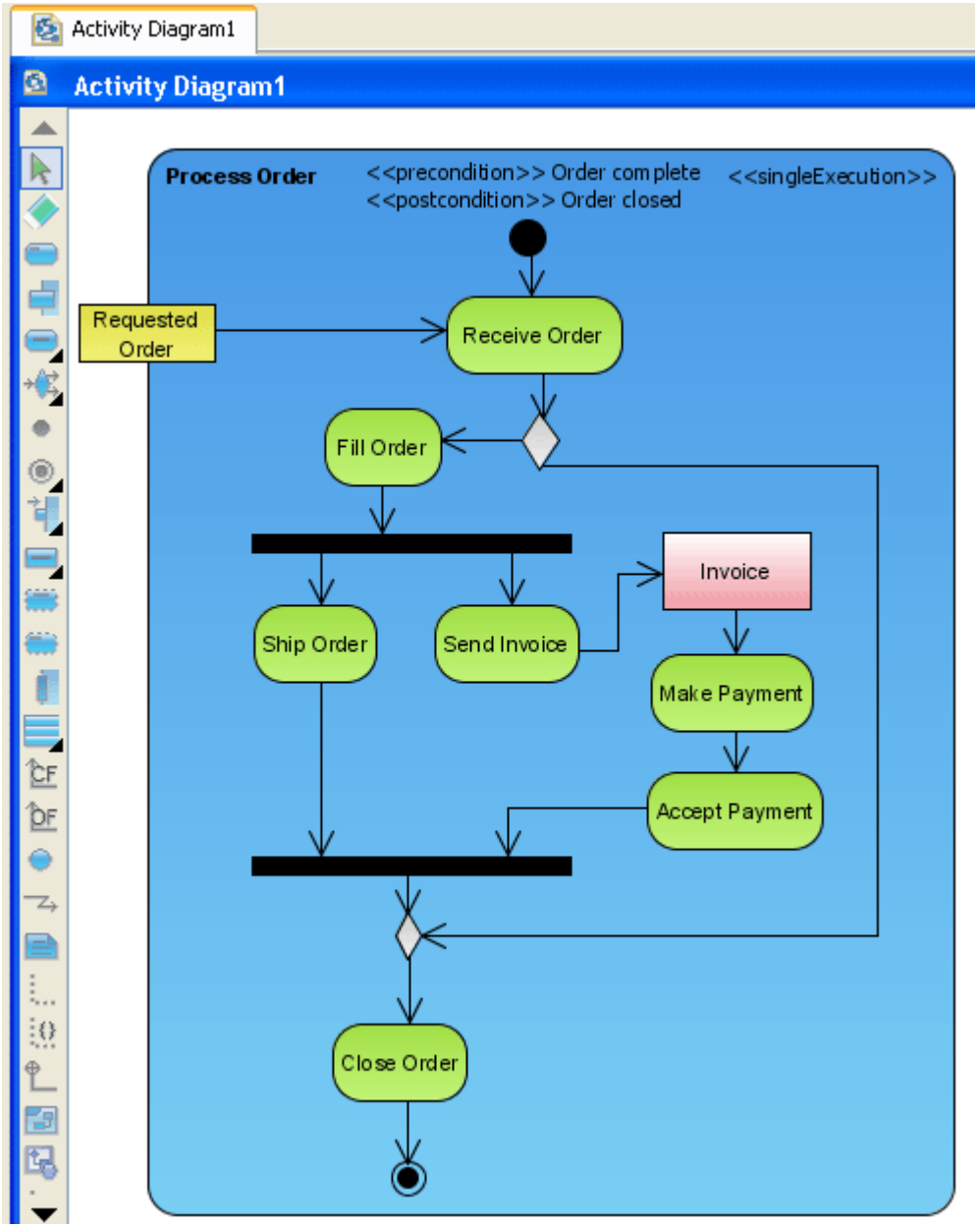
http://www.visual-paradigm.com/VPGallery/diagrams/index.html

Zamówienie



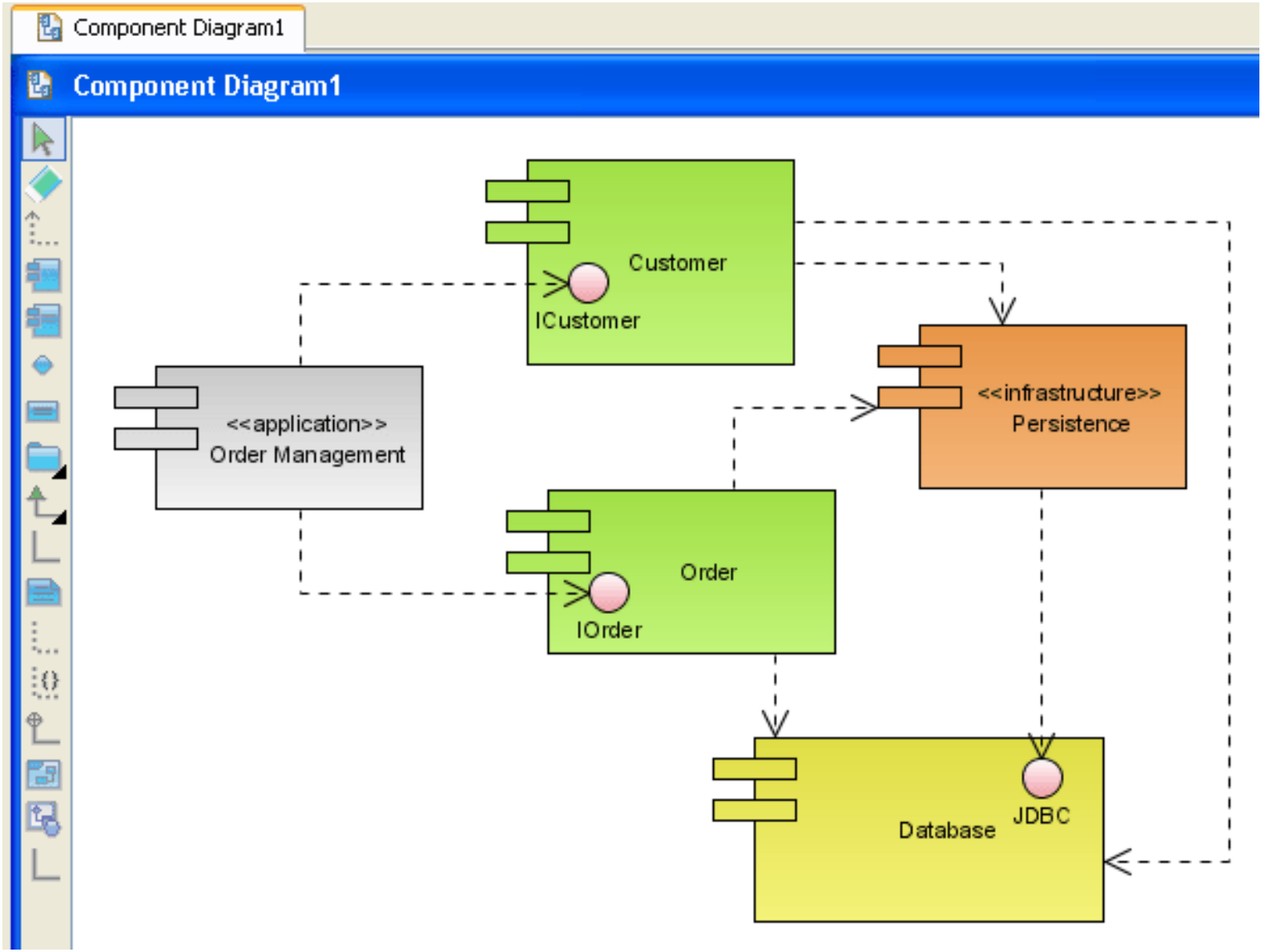
http://www.visual-paradigm.com/VPGallery/diagrams/index.html

Zamówienie



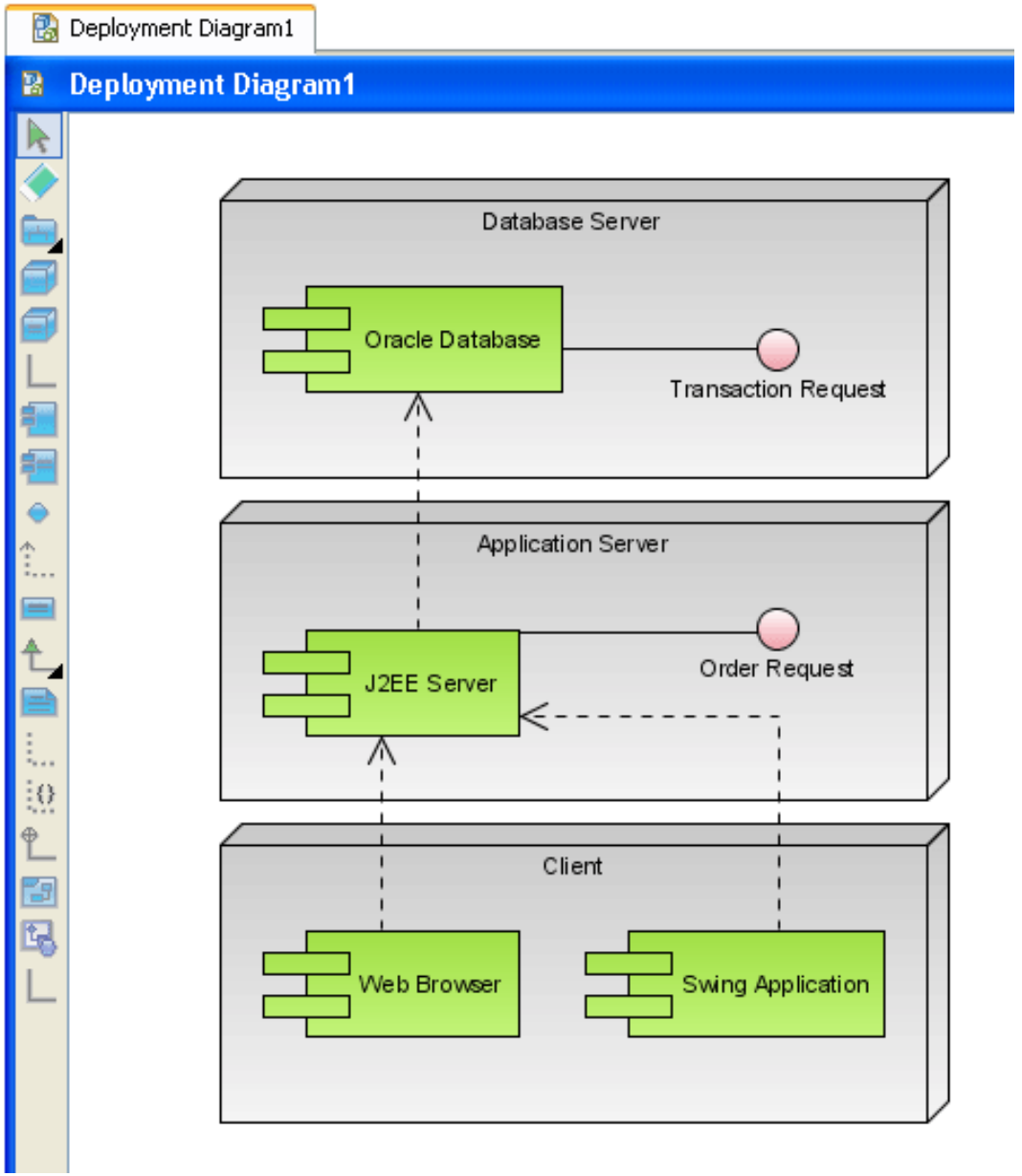
http://www.visual-paradigm.com/VPGallery/diagrams/index.html

Zamówienie



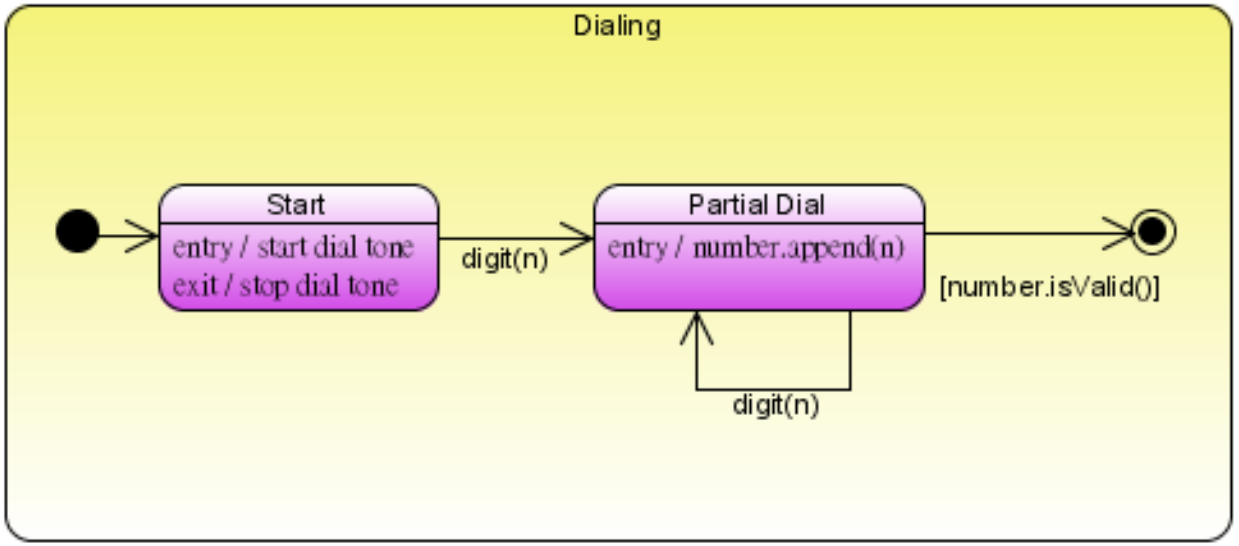
http://www.visual-paradigm.com/VPGallery/diagrams/index.html

Zamówienie



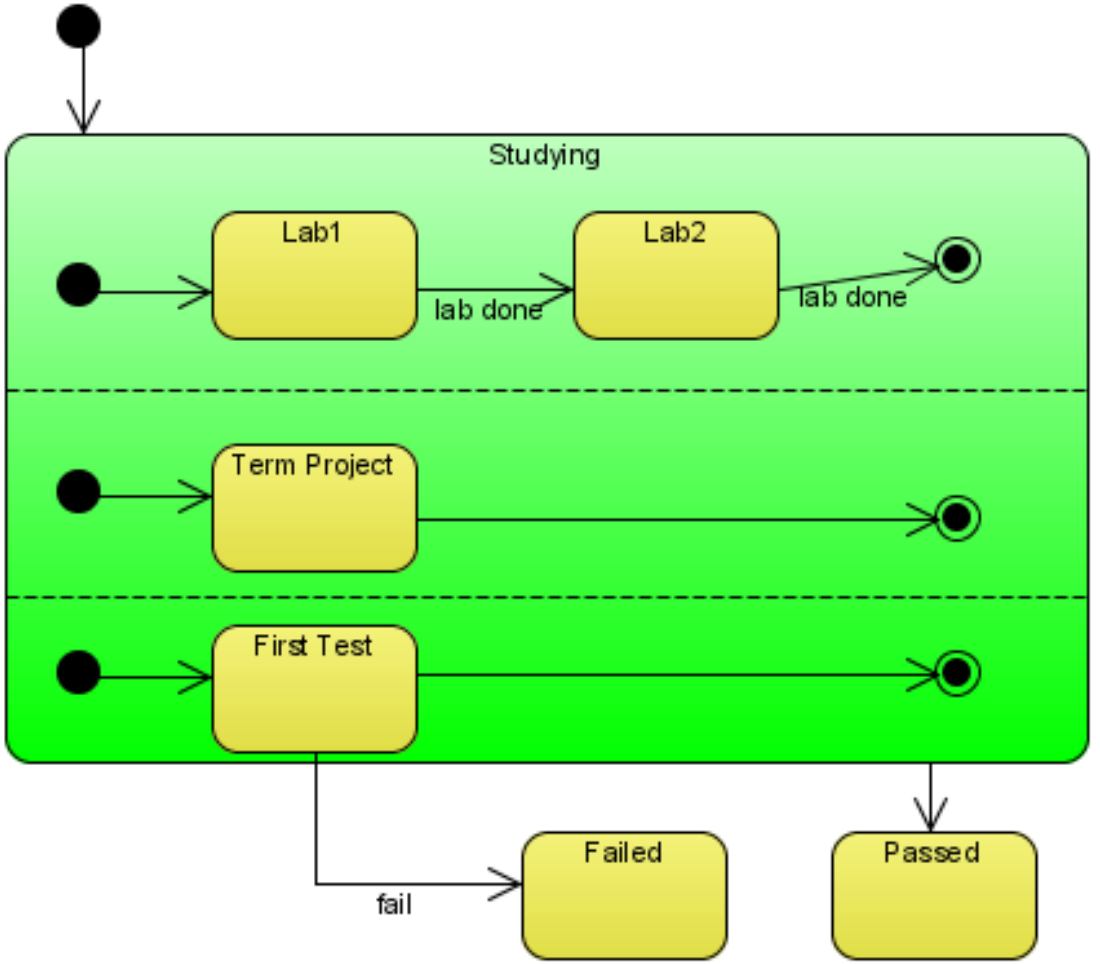
http://www.visual-paradigm.com/VPGallery/diagrams/index.html

Telefon



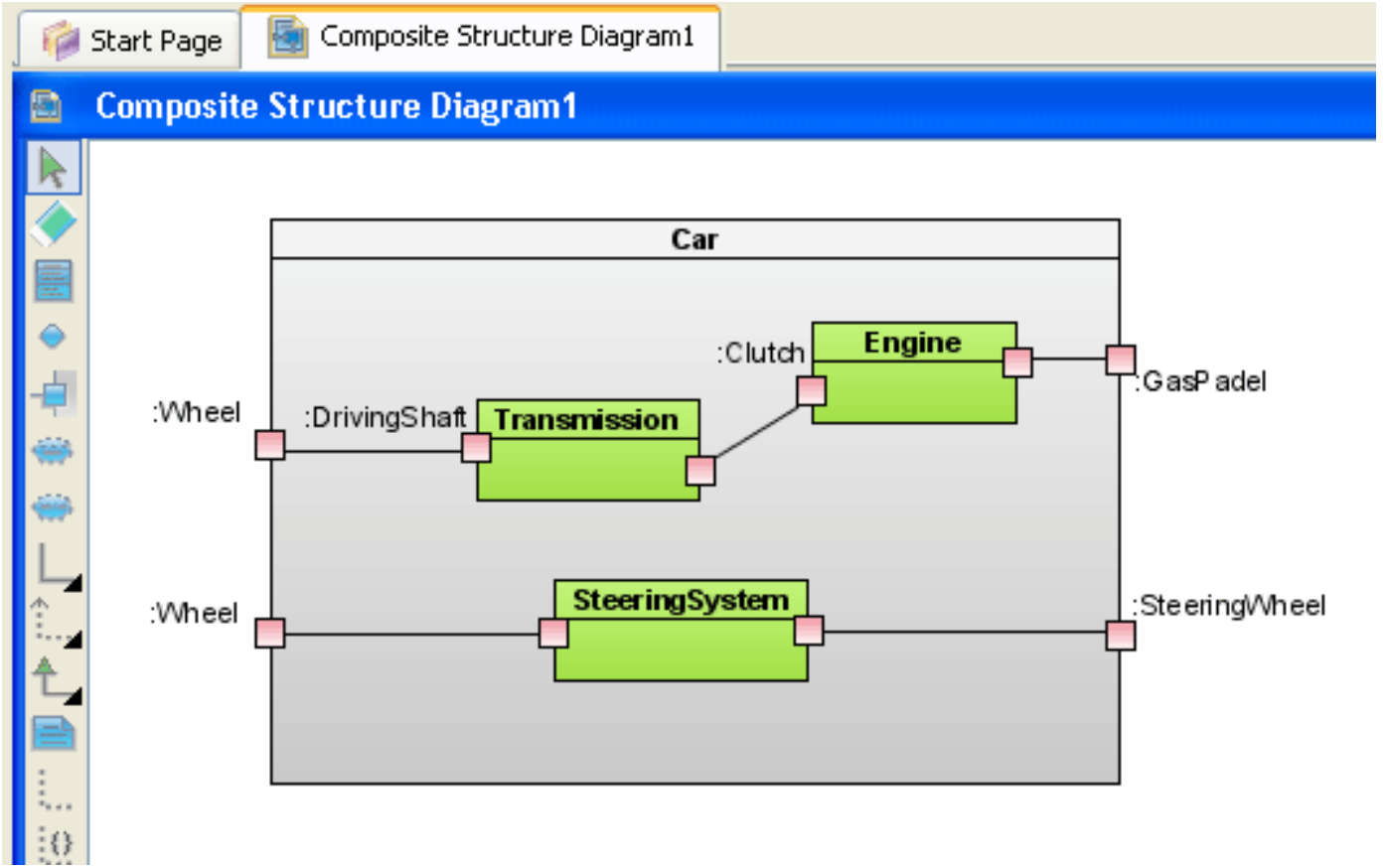
http://www.visual-paradigm.com/VPGallery/diagrams/index.html

Studia



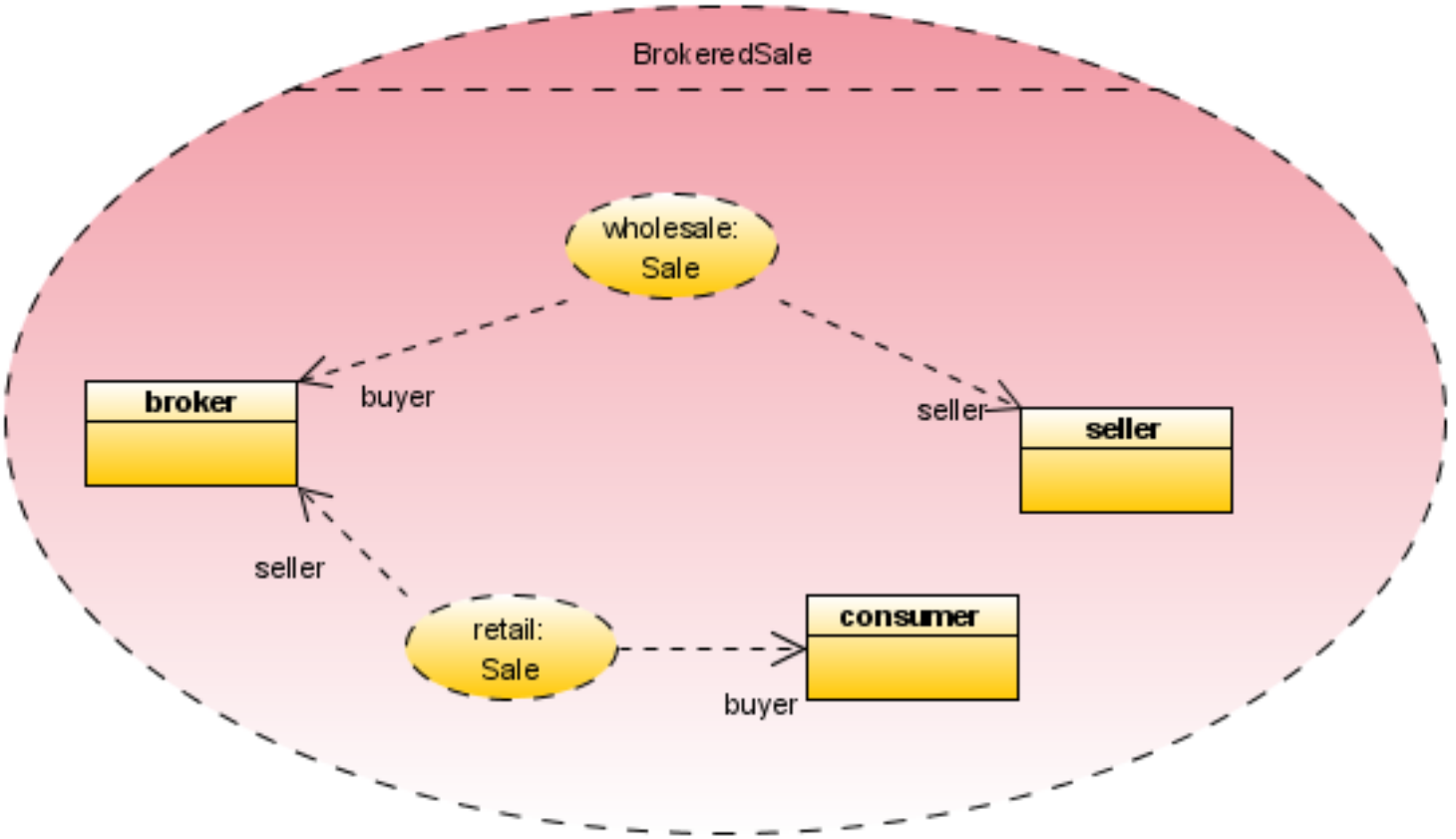
http://www.visual-paradigm.com/VPGallery/diagrams/index.html

Samochód



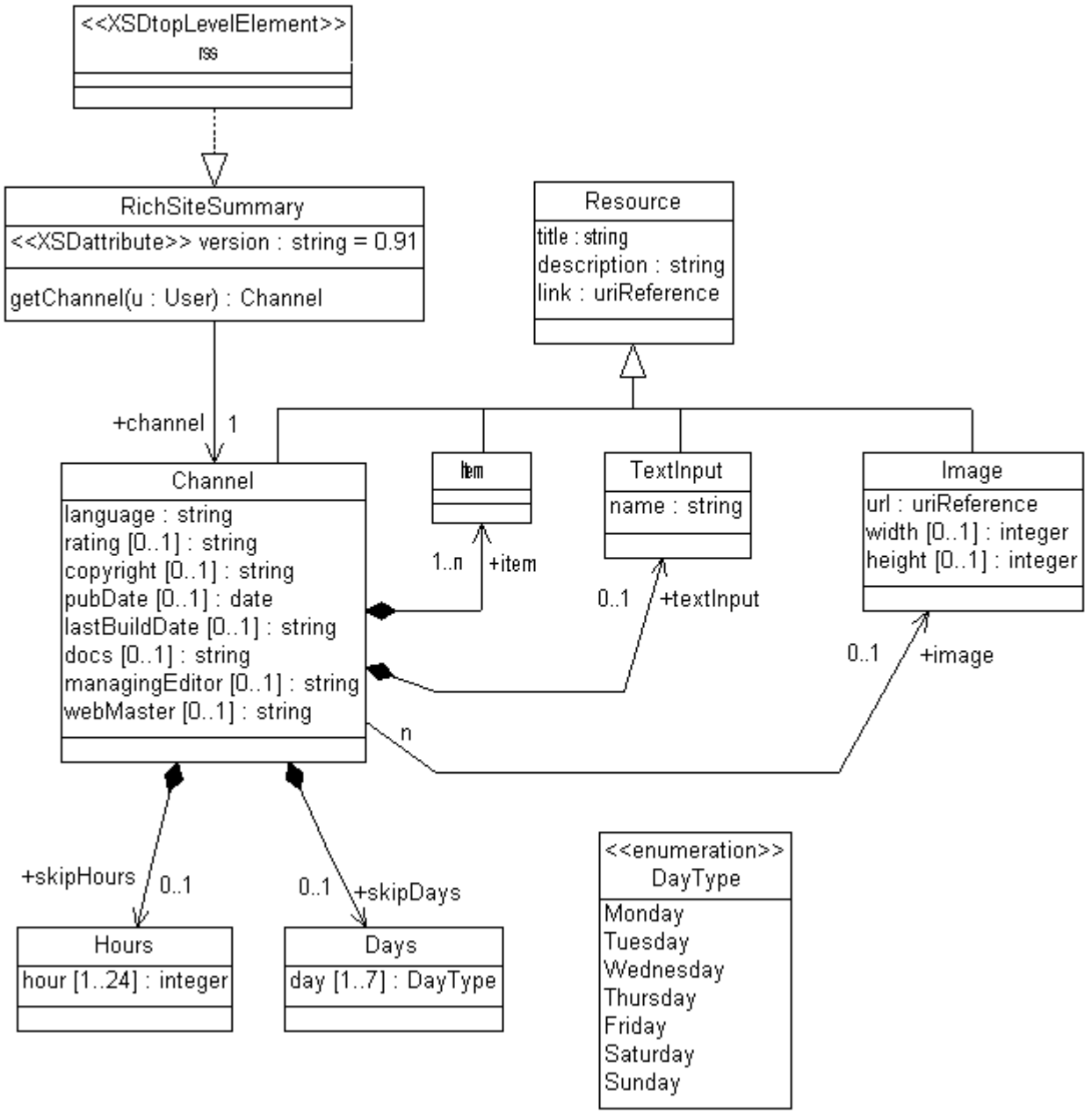
http://www.visual-paradigm.com/VPGallery/diagrams/index.html

Sprzedaż



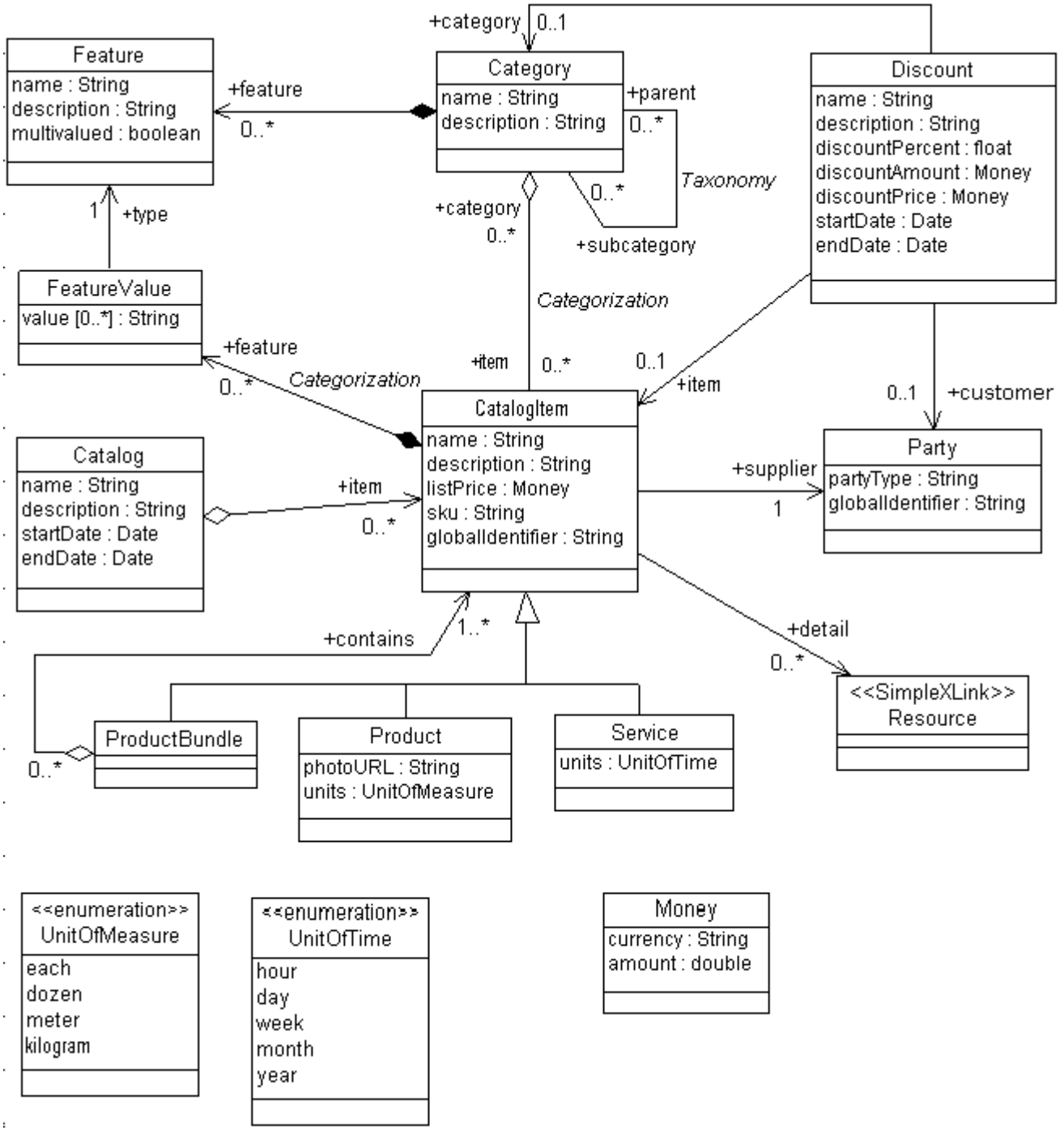
http://www.visual-paradigm.com/VPGallery/diagrams/index.html

RSS



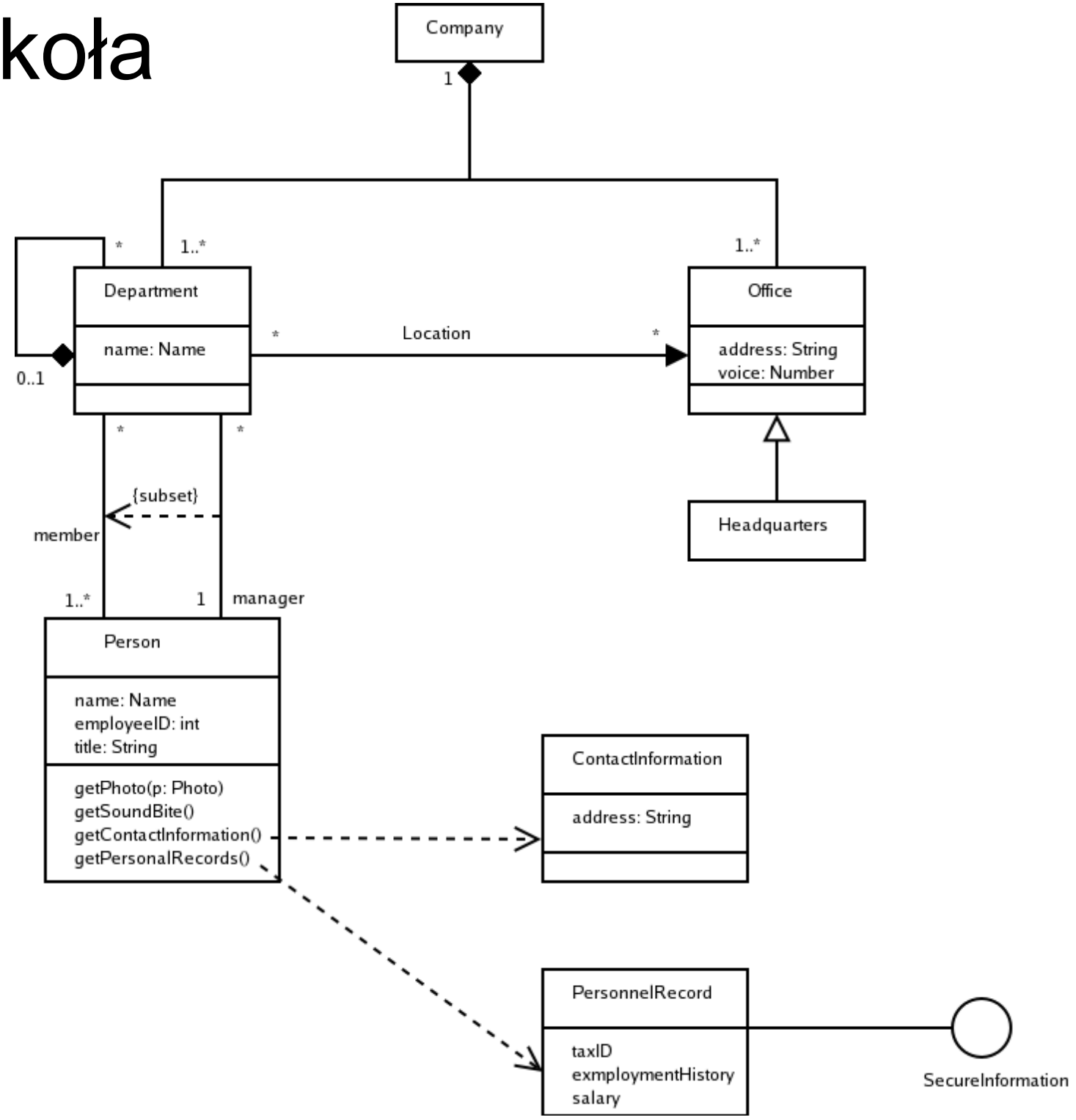
Modeling XML Applications with UML, by David Carlson, Ph.D. Published by: Addison-Wesley 2001, ISBN 0-201-70915-5

XML



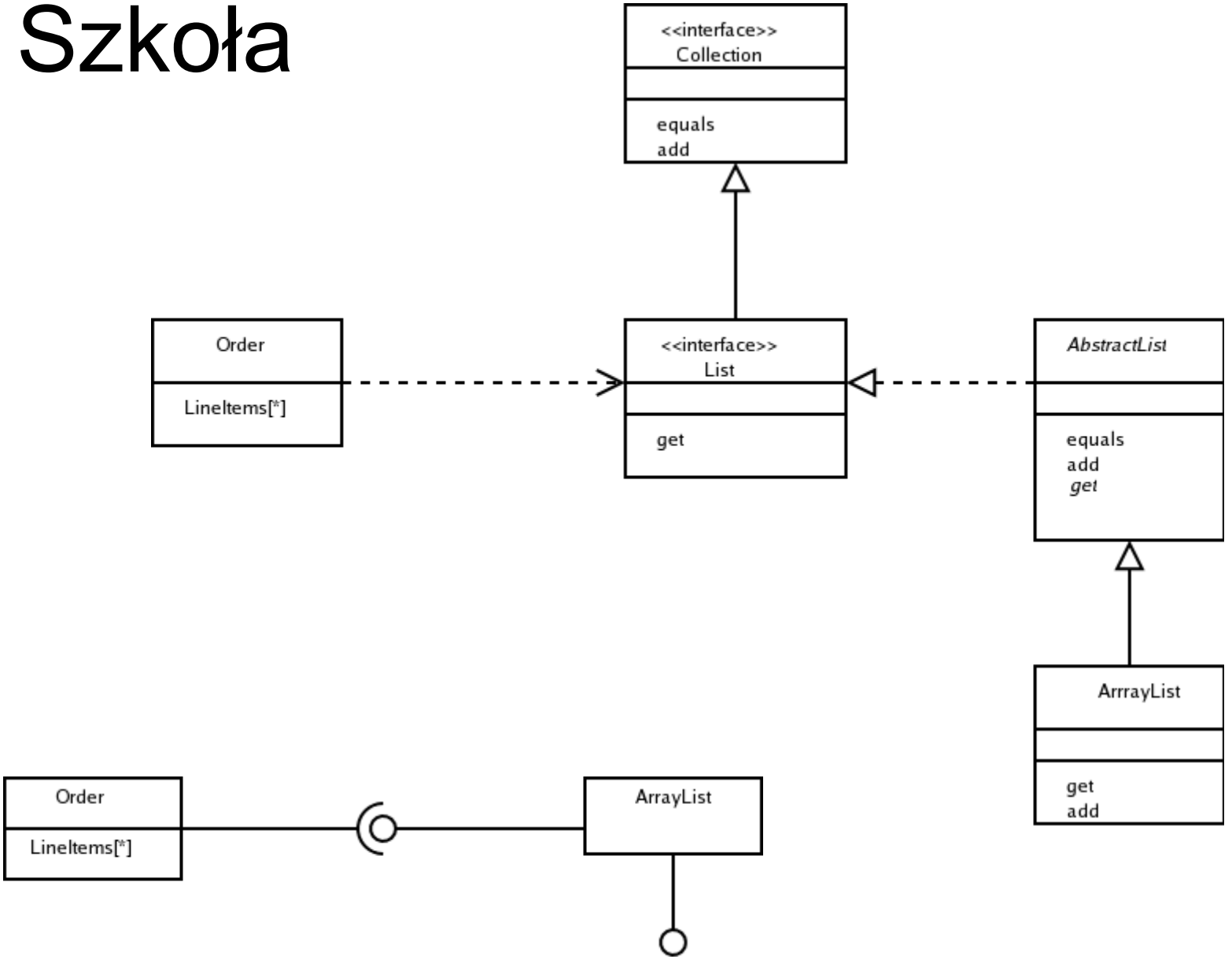
Modeling XML Applications with UML, by David Carlson, Ph.D. Published by: Addison-Wesley 2001, ISBN 0-201-70915-5

Szkoła



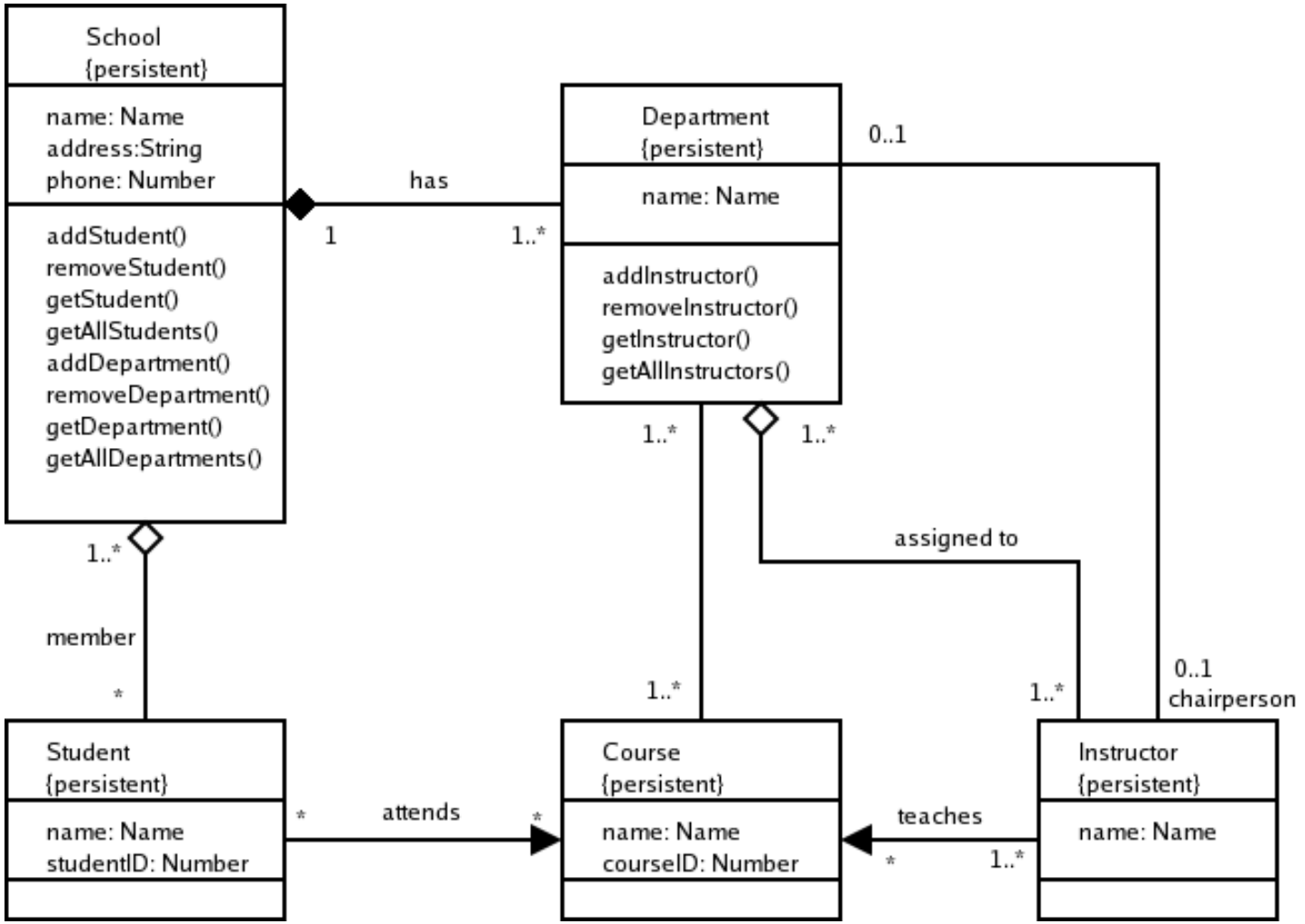
Booch, Rumbaugh and Jacobson 1999

Szkoła



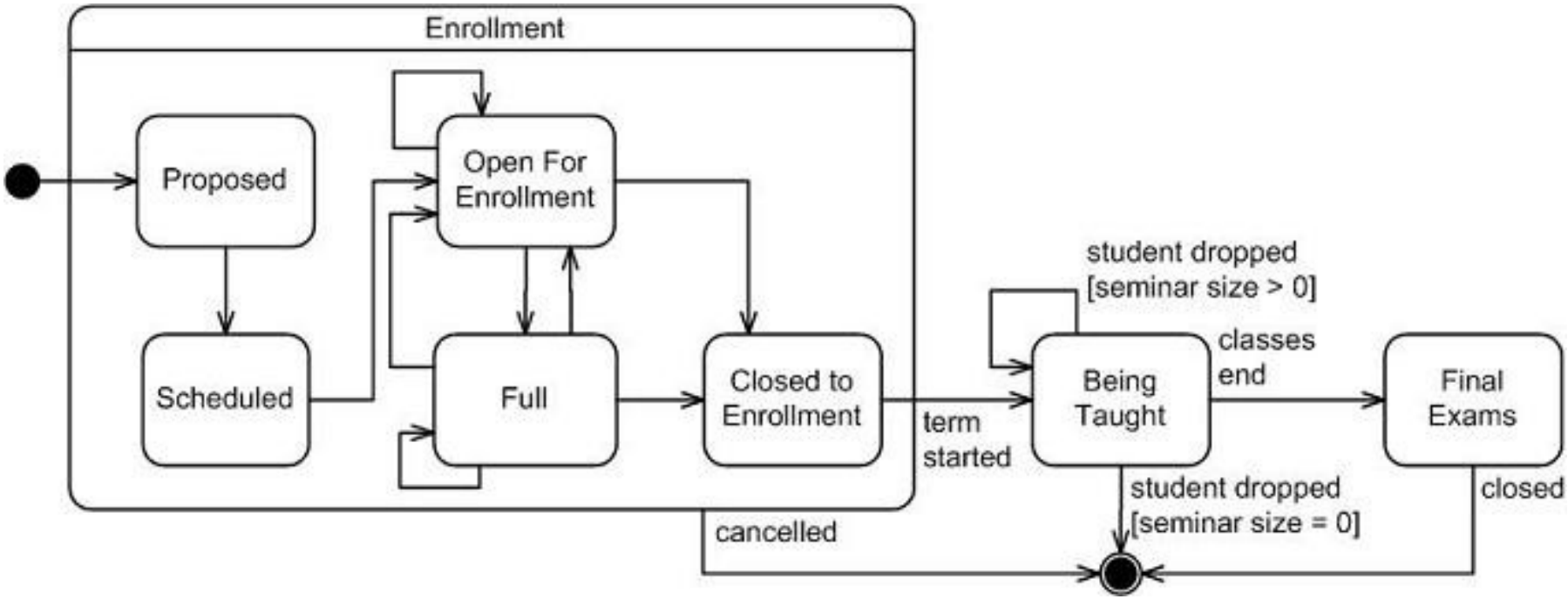
Fowler 2004

Szkoła



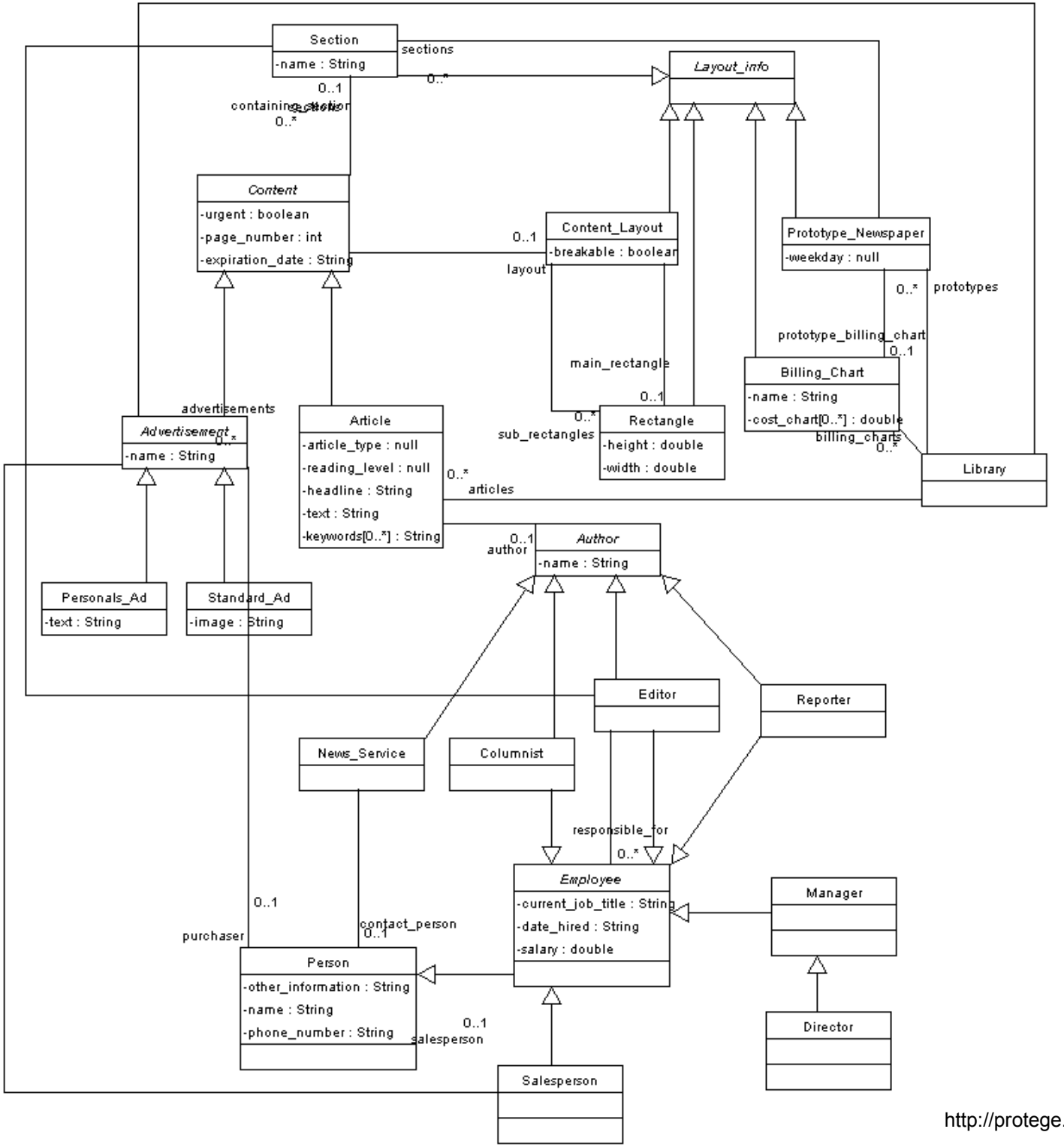
Fowler 2004

Seminarium

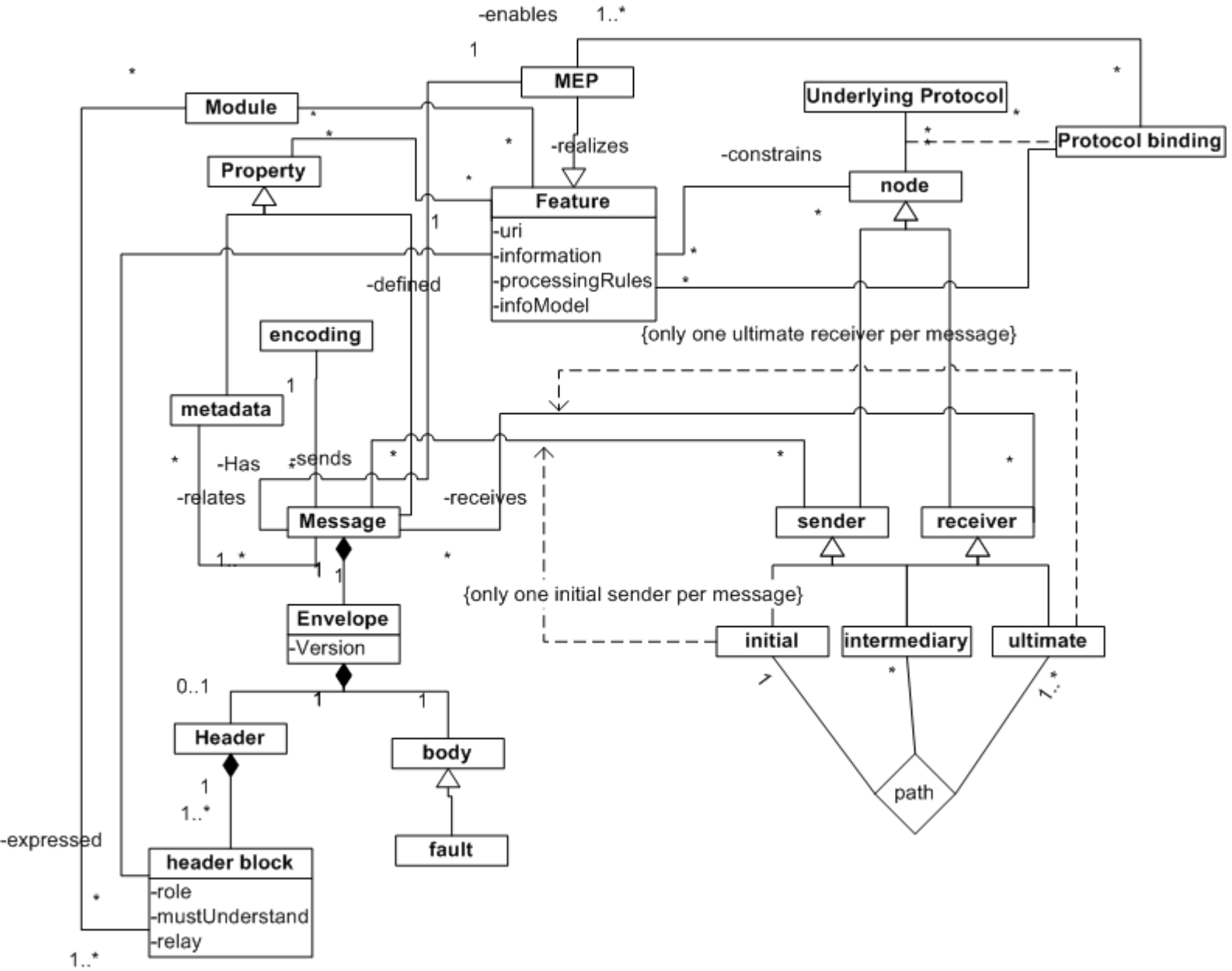


http://www.agilemodeling.com/artifacts/stateMachineDiagram.htm

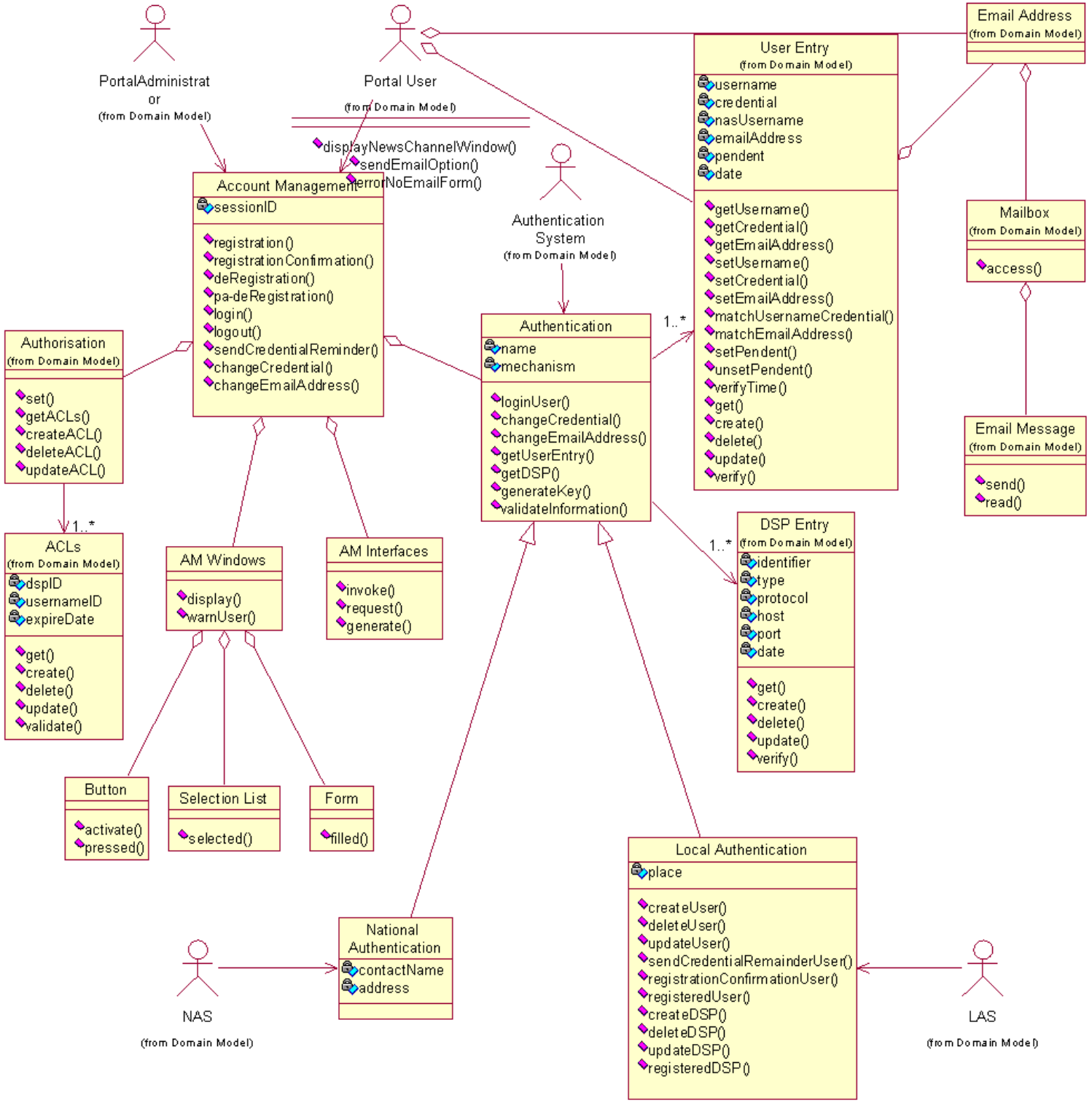
Gazeta



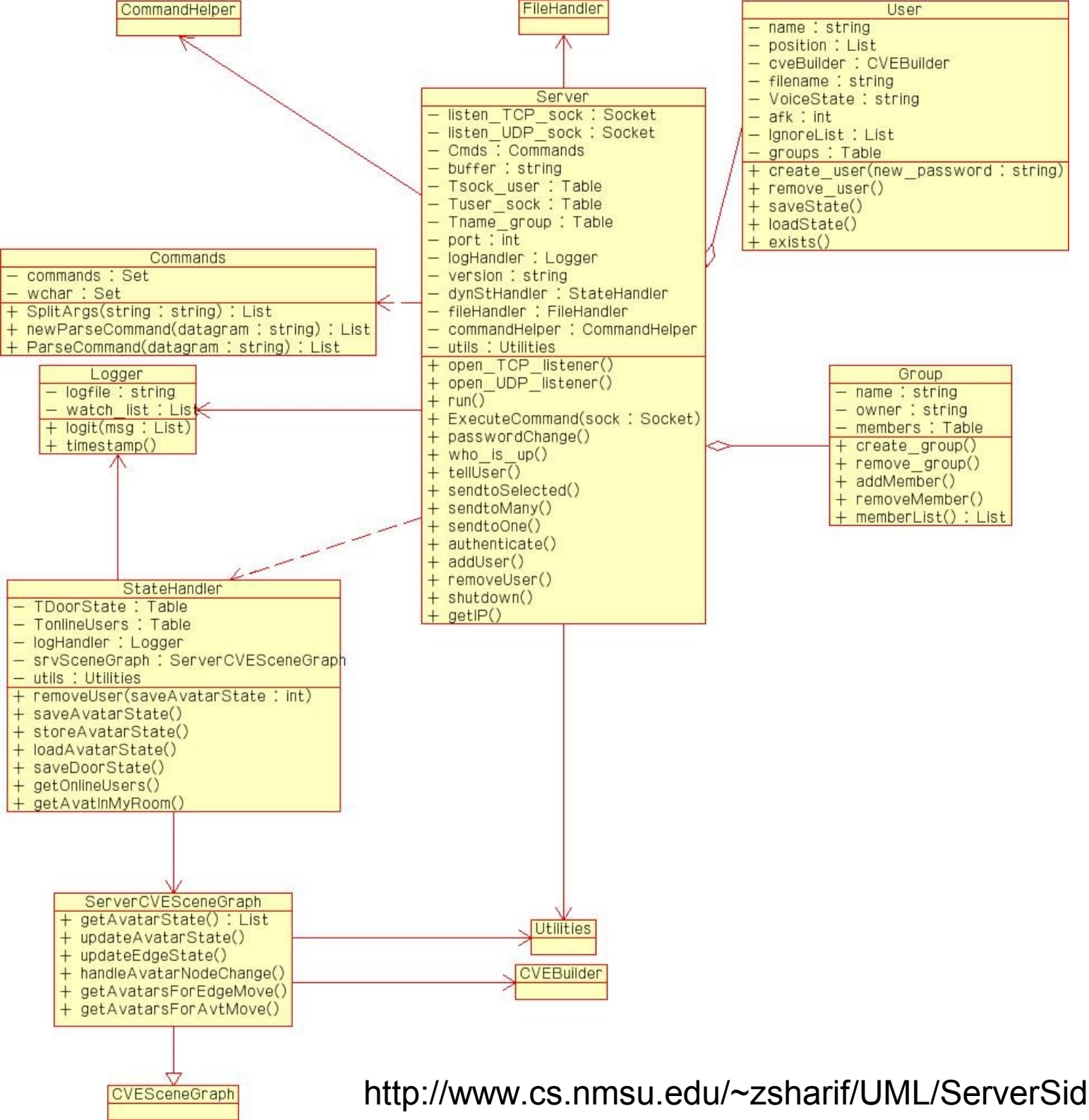
http://protege.stanford.edu/plugins/uml/images/newspaper-1.gif



http://www.novell.com/coolblogs/wp-content/uploads/2007/01/uml.png



http://www.ukoln.ac.uk/qa-focus/documents/case-studies/case-study-03/qa-uml



http://www.cs.nmsu.edu/~zsharif/UML/ServerSideUML_Details.jpeg