Model-Driven Engineering

Radosław Klimek

2015-22

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

Diagram sekwencji

Interakcje

- Diagramy interakcji są stosowane do modelowania dynamicznych aspektów systemu
- Wyróżnia się dwa podstawowe diagramy:
  - *diagramy przebiegu* (ang. *Sequence diagrams*) oraz
  - *diagramy kooperacji* (ang. *Collaboration diagrams*).

Diagram przebiegu inaczej **diagram sekwencji** modeluje kolejność komunikatów w czasie, **diagram kooperacji** modeluje organizację strukturalną obiektów wymieniających komunikaty.

Cele i przeznaczenie diagramu

- Służy do modelowania dynamicznych aspektów systemu
- Przedstawia współdziałanie kilku obiektów w celu realizacji konkretnego zadania z uwypukleniem perspektywy czasu
- Pozwala określić kolejność występowania komunikatów w czasie

Diagram sekwencji przedstawia obiekty (instancje klas) stanowiące składowe jakiegoś systemu oraz komunikaty wymieniane pomiędzy nimi w celu realizacji danego zadania. Diagram może, ale nie musi, zawierać również aktorów, oraz opisywać ich interakcję z systemem.

Zastosowanie, zakres

- Opis działania systemu
- Opis przypadków użycia
- Opis scenariusza przypadku użycia
- Opis operacji klasy

Układ diagramu

Diagram sekwencji ma dwa wymiary:

- wymiar poziomy – oś, na której umieszczono instancje klasyfikatorów biorące udział w interakcji, przedstawia role obiektów pomiędzy którymi trwa określona komunikacja,

- wymiar pionowy – oś czasu przedstawiająca ułożone chronologicznie komunikaty, te które są położone niżej realizowane są później.

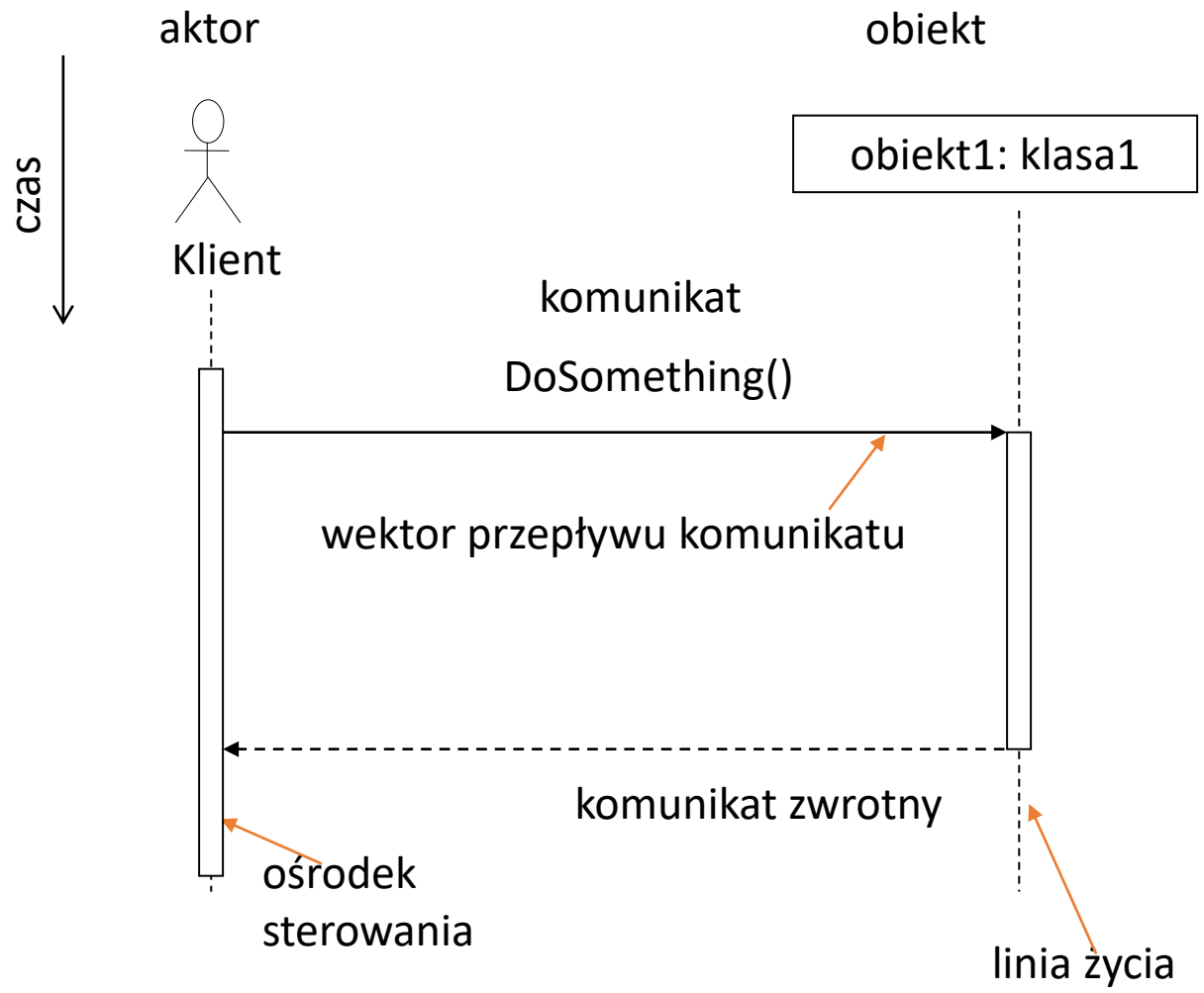
Rodzaje diagramów sekwencji

- **Konceptualny** – diagram o ogólnym zakresie i łatwych do zidentyfikowania interakcjach, stwarza możliwość ogólnego zidentyfikowania interakcji naszkicowania zakresu i zawartości interakcji
- **Implementacyjny** – diagram stanowiący opracowania specyfikacji programistycznej, prezentuje o wiele wyższy poziom precyzji oraz większy zakres interakcji
- **Wystąpieniowy** – przypadek diagramu implementacyjnego w odniesieniu do konkretnego scenariusza

Pojęcia podstawowe

- **Instancja klasyfikatora** – abstrakcyjna kategoria modelowania, która uogólnia kolekcję instancji o tych samych cechach
- **Linia życia** – powiązana z konkretną instancją klasyfikatora linia na diagramie sekwencji, specyfikowane przez pionowe linie przerywane symbolizujące czas istnienia obiektów.
- **Komunikat** – specyfikacja wymiany informacji między obiektami, zawierająca zlecenia wykonania określonej operacji
- **Ośrodek sterowania** – specyfikacja wykonywania czynności, operacji lub innej jednostki zachowania w ramach interakcji – w praktyce realizacja funkcjonalności, reprezentuje czas, w którym obiekt jest w stanie przyjmować i nadawać komunikaty (czas aktywności)., specyfikowany przez cienki prostokąt.Możliwe jest zagnieżdżanie specyfikowane przez inny prostokąt, przesunięty względem danego.

Przykład podstawowy



Aktor na diagramie

• W diagramie przebiegu aktor jest reprezentacją zewnętrznego użytkownika (lub ich grupy), którzy oddziaływają na system.

• Aktor, podobnie jak obiekty, ma oś czasu. Jeżeli aktor jest źródłem interakcji (a tak najczęściej jest) to usytuowany jest w brzegowej (skrajnej lewej lub prawej) części systemu.

• Dla aktora ustawić można parametry:

- nazwa
- kod
- komentarz

Obiekt na diagramie

• Obiekt na diagramie jest instancją klasy i tak jak aktor jest połączony z osią czasu. Jeżeli obiekt jest tworzony bądź niszczony, jest to zaznaczone na osi czasu.

• Parametry obiektów są takie jak dla aktora (nazwa, kod, komentarz), dodatkowo ustawić możemy pole Classifier, które określa nazwę klasy, której instancją jest obiekt.

• Jeżeli jeszcze nie istnieje dana klasa, możemy ją stworzyć na podstawie obiektu „Create Class”.

Linia życia

Linia życia to rola uczestnika interakcji, jaką pełni w czasie jej trwania.

Linia życia reprezentuje współuczestnika interakcji i czas jego istnienia podczas realizacji scenariusza. Linie życia reprezentują konkretne byty – obiekty, systemy i mogą przyjmować stereotypy, które świadczą o roli, jaką pełni dany obiekt w systemie.

Linia życia

W głowie linii życia mogą występować różne oznaczenia:

Name – nazwa instancji odnosząca się do określonych elementów;

Name: NameClass – nazwa instancji i nazwa klasy;

: NameClass – nazwa klasy;

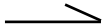
Komunikat

Komunikat jest to informacja przesyłana pomiędzy obiektami.

- Komunikat posiada nadawcę, odbiorcę i akcję, czyli treść informacji, która pojawia się nad symbolem strzałki. Tworząc komunikat określić można też tzw. aktywację (ośrodek sterowania) która reprezentuje czas wykonywania czynności przez obiekt. Aktywacja nie posiada własnej listy właściwości, jedynie graficzną reprezentację w postaci prostokąta.
- Możliwe jest tworzenie komunikatów pomiędzy obiektami, aktorami, a nawet tworzenie komunikatów rekurencyjnych, tj. takich gdzie nadawca i odbiorca to ten sam obiekt / aktor.

Rodzaje komunikatów

Komunikat synchroniczny – wywołujący czeka na akceptację wywołania przez wywoływanego, po czym następuje realizacja operacji, i przekazanie ewentualnego wyniku wywołującemu, po tym oba obiekty kontynuują swoje działania. Następuje więc ścisłe wiązanie obu przepływów sterowania. W komunikacji między procesami/wątkami, może to być (zdalne) wywołanie proceduralne.

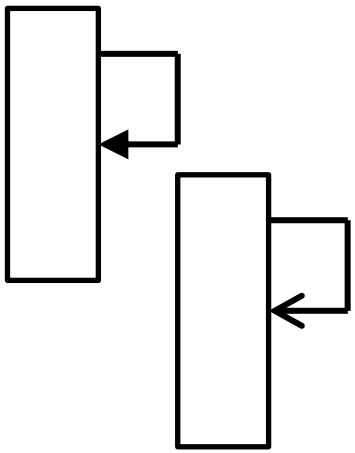


Komunikat asynchroniczny – asynchroniczny przepływ sterowania. Obiekt wysyłający nie czeka na odpowiedź odbiorcy, i może jednocześnie wykonywać inne operacje. Oba obiekty i ich przepływy sterowania nie są ze sobą zsynchronizowane. W komunikacji między procesami/wątkami może to być przekazywanie komunikatów.

Wywołanie proceduralne – oznacza wywołanie procedury lub inny zagnieżdżony przepływ sterowania. Może to być zatem zwykłe wywołanie procedury lub dotyczyć aktywnych obiektów, gdzie jeden wysyła sygnał i czeka na odpowiedź. Nadawca musi oczekiwać na odpowiedź, bądź koniec czasu aktywacji.

Powrót z wywołania procedury. Może być również traktowane jako koniec aktywacji. Zwykle jest to odpowiedź ba wywołanie procedury (return).

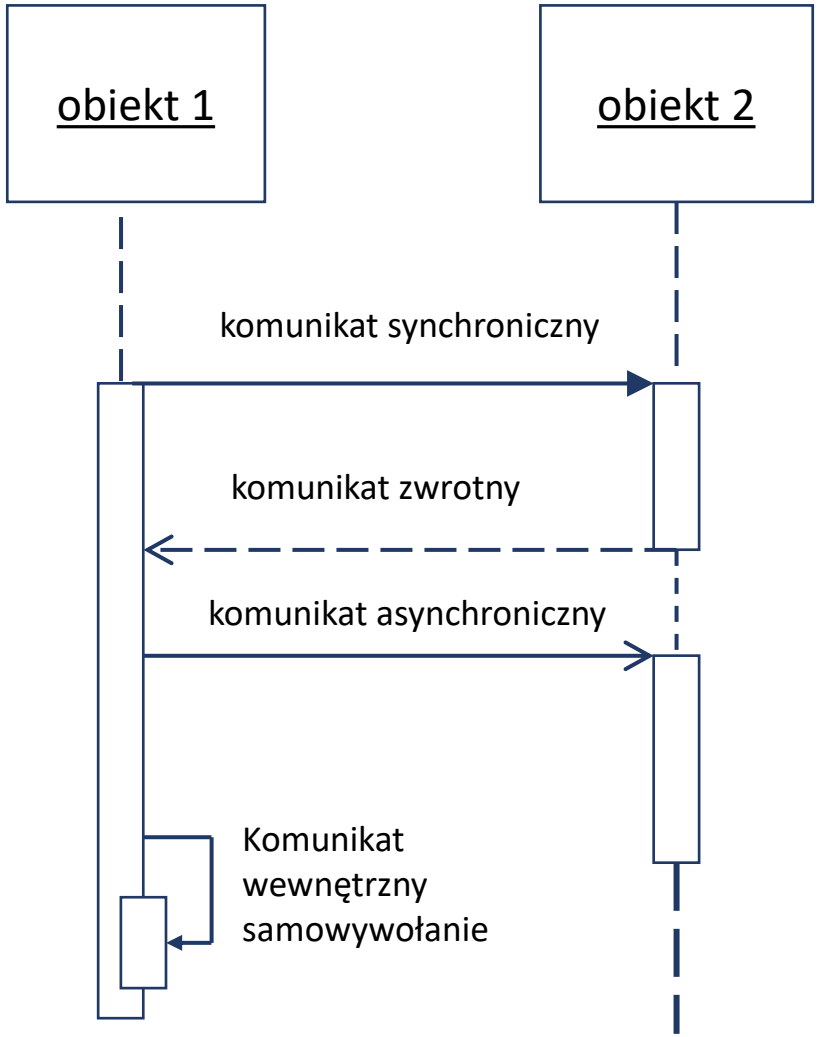
Inne rodzaje komunikatów



Samowywołanie (np. synchroniczne lub asynchroniczne) – jest to wiadomość wewnętrzna, którą obiekt wysyła sam do siebie. Tego typu wiadomości używa się na przykład w sytuacji, gdy obiekt wywołuje swoją własną metodę.

Komunikat niezdefiniowany – jeśli nie jest istotna zależność pomiędzy komunikatami, lub jej na razie nie znamy.

Komunikat – przykłady



Komunikat – nieformalna składnia, numery porządkowe

- **numer porządkowy**
  może składać się z kilku segmentów, każdy segment jest liczbą całkowitą określającą kolejność na danym poziomie zagłębienia, np.: 1.2.1 poprzedza 1.2.2 oraz poprzedza 1.3
  komunikaty współbieżne określane są przez dodanie litery za numerem porządkowym, np.. 1.2a i 1.2b

- **określenie komunikatu poprzedzającego**
  zamiast numeru porządkowego danego komunikatu można podać numery porządkowe komunikatów poprzedzających (lista numerów oddzielonych przecinkami i zakończona znakiem „/”).
  dany komunikat zostanie przesłany tylko wówczas, gdy wszystkie poprzedzające go komunikaty zostaną przesłane (synchronizacja)

Komunikat – nieformalna składnia, dozory i iteracje

- **warunek strzegący, dozoru**
  warunek dozoru, który musi być spełniony dla przesłania komunikatu.
  warunek dozoru podaje się w nawiasach kwadratowych, np. [n<10]

- **określenie iteracji**
  podanie klauzuli iteracji w nawiasach kwadratowych poprzedzonych znakiem „*” oznacza wielokrotne przesłanie komunikatu.
  klauzulę iteracji podaje się w określonym języku programowania lub w pseudokodzie, np.: *[i:=1..10]

Komunikat – nieformalna składnia, wynik i argumenty

- **zwracany wynik**

  wynik := nazwa operacji (lista argumentów)

- **lista argumentów**

  • lista wartości oddzielonych przecinkami, które są po kolei przypisywane do parametrów operacji.

  • wartości są wyrażeniami w określonym języku programowania lub w pseudokodzie.

  • w wyrażeniach mogą występować wartości zwrotne z poprzedzających komunikatów

Komunikat – przykłady

1.2a: DoSomething()

3, 5/7: DoSomething()

4: [action=new]NewDocument(SelectType())

Komunikat – bardziej formalnie

Etykieta specyfikuje wysyłany komunikat, jego argumenty i wartości zwracane, kolejność komunikatu w kontekście interakcji, łącznie z zagnieżdżeniem wywołania rozgałęzieniami, współbieżnością i synchronizacją.

```
[predessor] [guard_condition] [sequence_expression]

[return_value_list :=] message_name (argument_list)
```

Komunikat – bardziej formalnie

predessor	lista numerów w sekwencji oddzielonych przecinkami. Na końcu listy umieszczony jest ukośnik. Numery mogą być specyfikowane przez wyrażenia. Numery te odnoszą się do komunikatów, które dotrzeć zanim możliwe będzie wysłanie danego komunikatu.
guard_condition	jeśli specyfikowana jest ta fraza, to warunkiem wysłania komunikatu jest dodatkowo spełnienie tego warunku. Wyrażenie to (dozór) może zatem określać warunki synchronizacyjne wątków.

Komunikat – bardziej formalnie

`sequence_expression`

specyfikuje listę termów sekwencji oddzielonych kropkami, a zakończonych dwukropkiem (:). Każdy term określa poziom proceduralnego zagnieżdżenia w obrębie interakcji. W przypadku współbieżności zagnieżdżenie to nie występuje. Struktura termu: label [recurrence], gdzie label może być *nazwą* lub *liczbą*. Jeśli jest liczbą, wówczas określa porządek wywołania, kolejność względem poziomu wyższego. *Nazwa* specyfikuje natomiast współbieżne wykonanie wątku na danym poziomie zagnieżdżenia. recurrence - reprezentuje iterację lub rozgałęzienie. Postać recurrence:

*[iteration_clause] - iteracja lub
[condition_clause] - rozgałęzienie.

Iteracja oznacza sekwencję komunikatów na danym poziomie zgnieżdżenia. Może to być wyrażone w języku programowania lub pseudokodzie (*[i:=1..n]). Iteracja dotyczy sekwencji. Równoległość wykonania może być specyfikowana przez *||. Rozgałęzienie może być również wyrażone w języku docelowym lub pseudokodzie [x>y].

Komunikat – bardziej formalnie

`return_value_list`

lista nazw oddzielonych przecinkami specyfikująca wartości zwracane przez komunikat. Identyfikatory mogą występować
również w liście argumentów. Parametr opcjonalny, może
być usunięty wraz ze znakiem przypisania

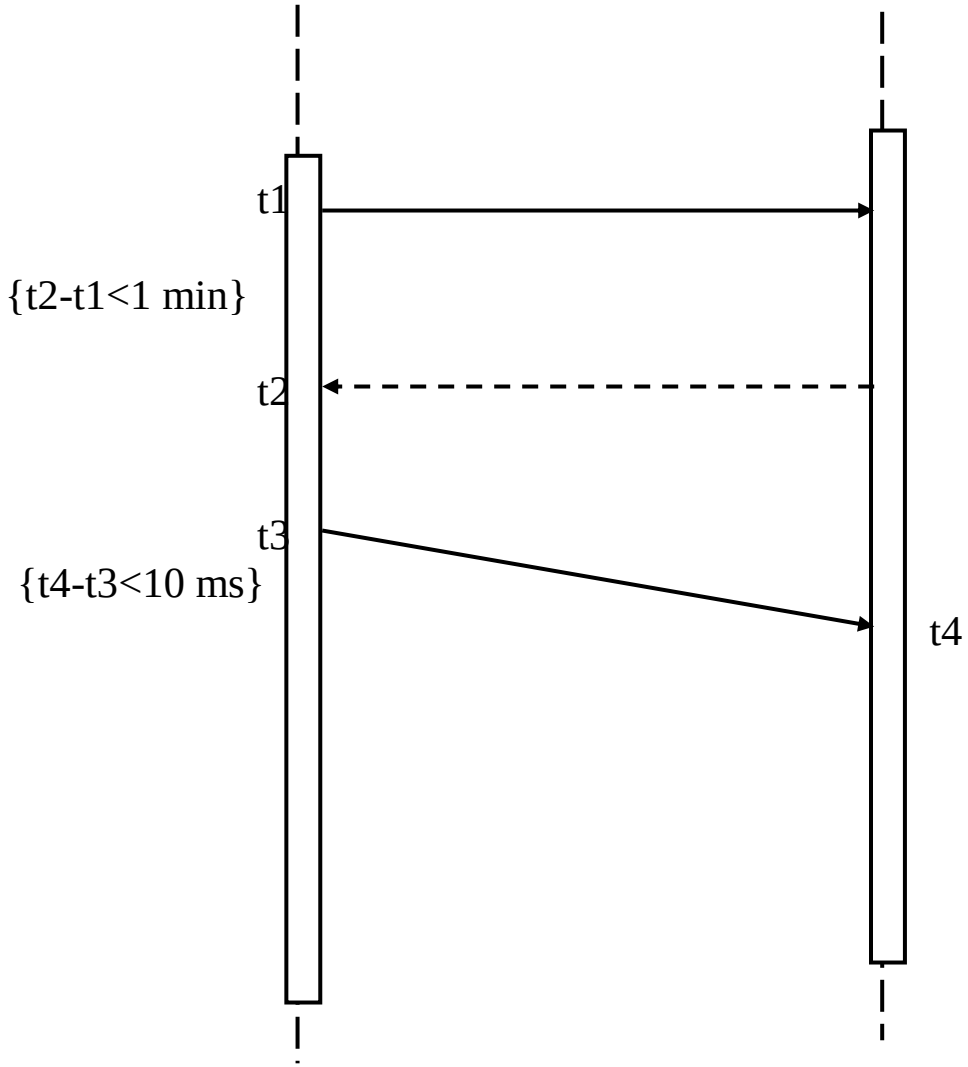
`message_name`

nazwa zdarzenia, które wystąpiło w obiekcie docelowym - często jest to żądanie operacji, która ma być wykonana.

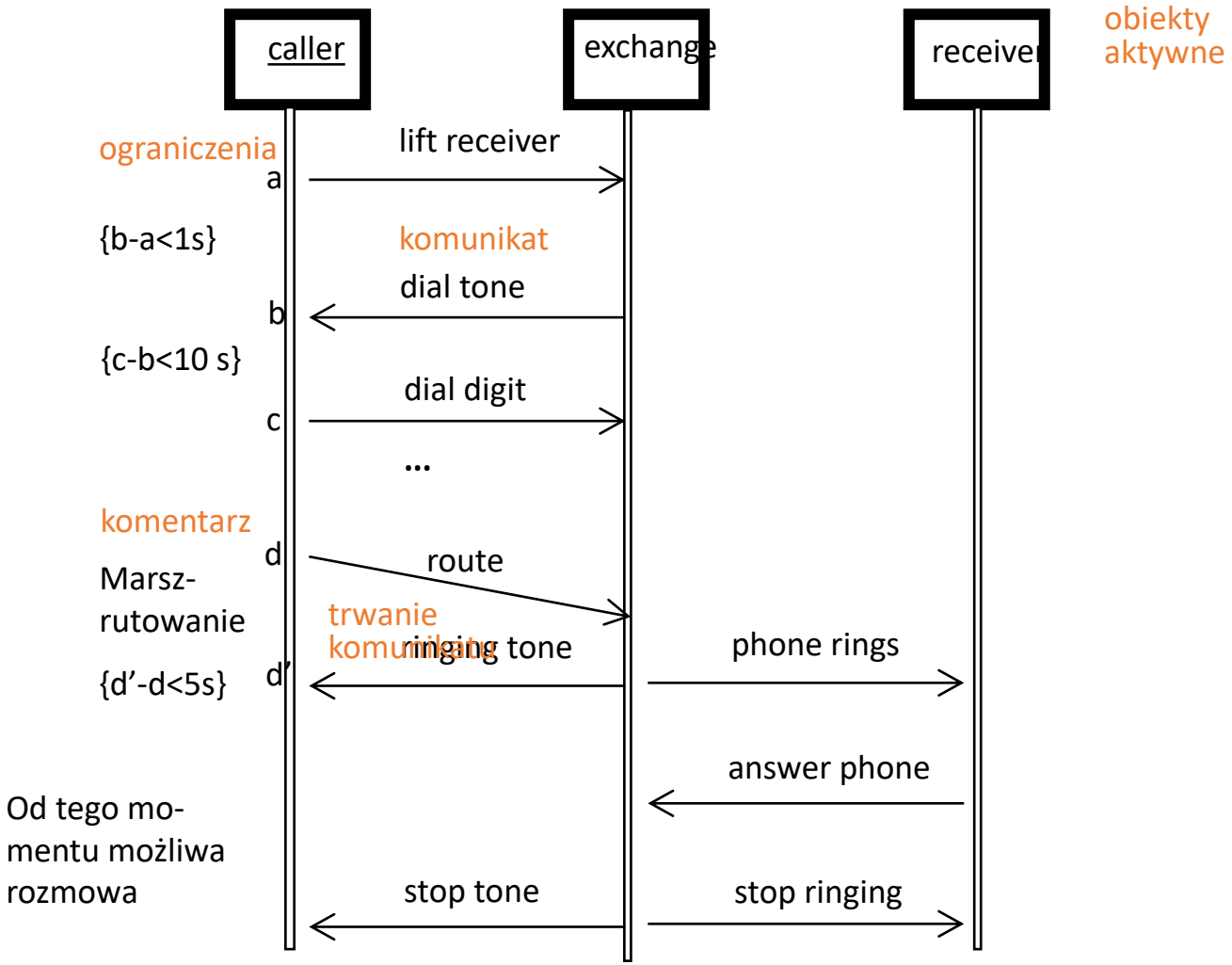
`argument`$_{\texttt{list}}$

lista argumentów oddzielonych przecinkami zamknięta w nawiasach okrągłych. Argumenty mogą być zdefiniowane wg konwencji języka programowania lub w pseudokodzie.

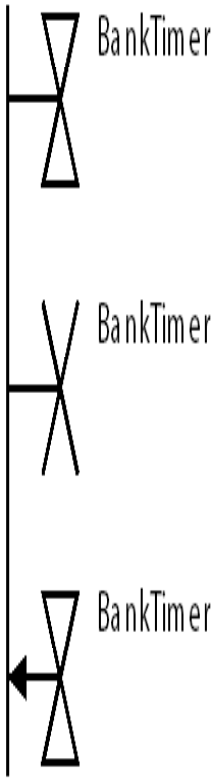
Dokładna specyfikacja czasu



Dokładna specyfikacja czasu – kolejny przykład



Modelowanie czasu – zegar



Trzy podstawowe rodzaje zdarzeń:

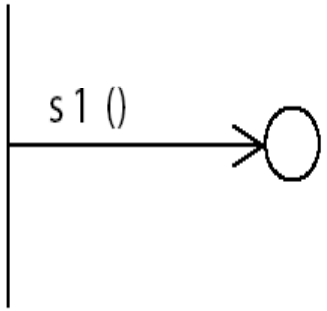
- ustawienie zegara;

- zerowanie zegara;

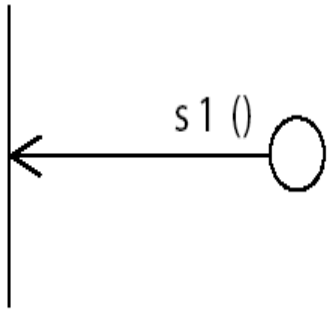
- przepełnienie (timeout) zegara.

Wiadomości niekompletne

Wiadomość niekompletna występuje wtedy, gdy nie zostało zdefiniowane jedno z dwóch związanych z nią zdarzeń.

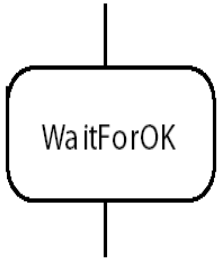


Wiadomość zagubiona –znane zdarzenie nadawcze (w obiekcie nadawcy).



Wiadomość odnaleziona –znane zdarzenie odbiorcze (w obiekcie odbiorcy).

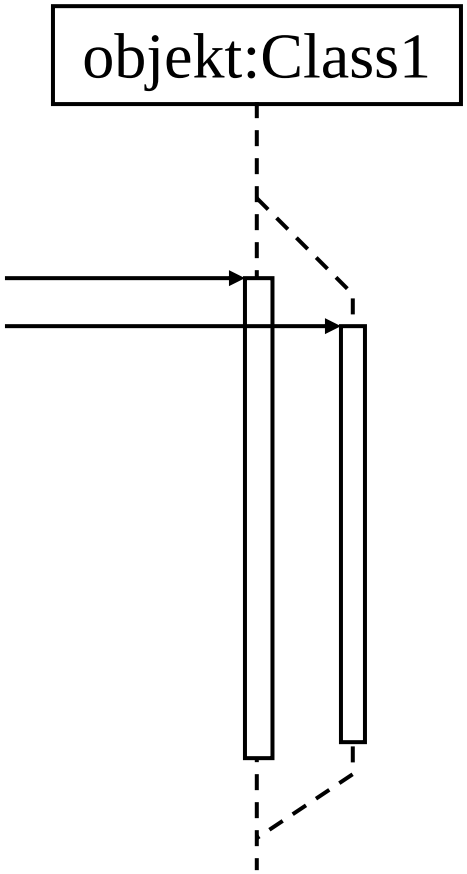
Stany



Stany – nazwy konkretnych stanów, na oznaczenie, że instancja musi się znaleźć w konkretnym stanie wyspecyfikowanym na linii życia.

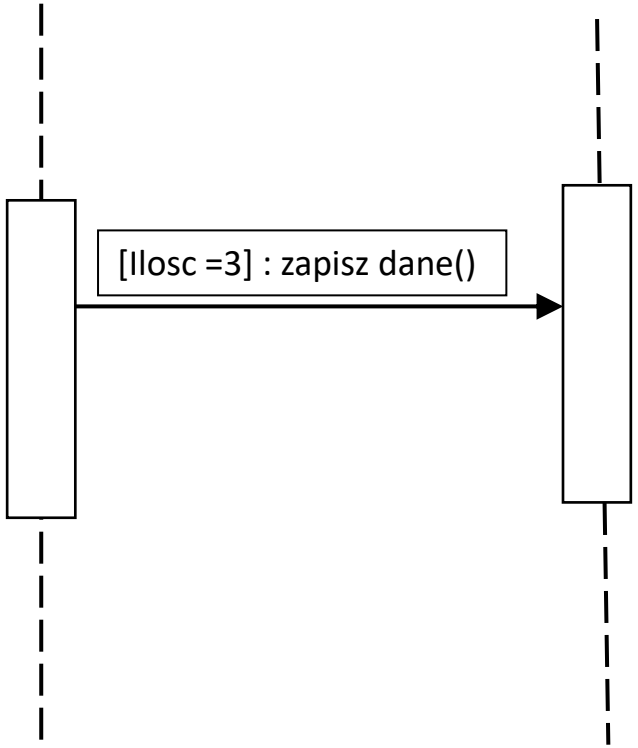
Niektóre pakiety dopuszczają specyfikowanie stanów

Współbieżne wątki sterowania



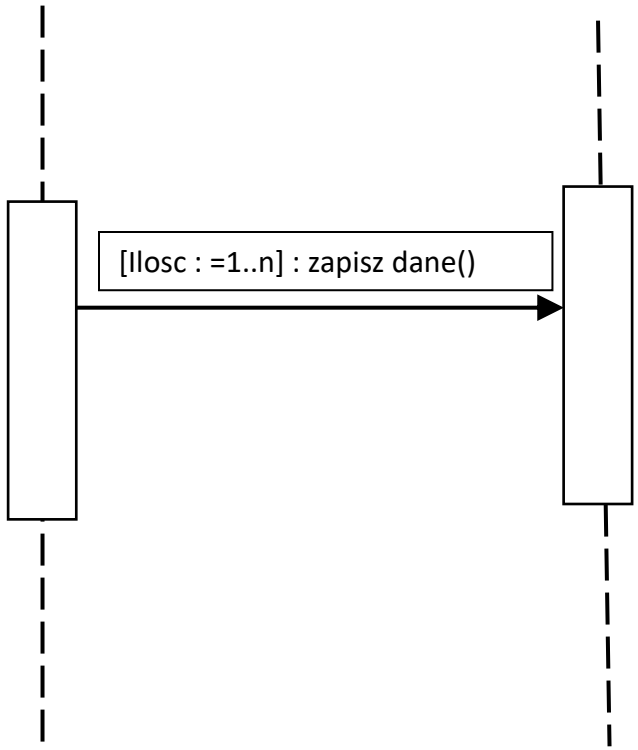
Komunikat – dozór

Dozór określa warunek logiczny jaki musi zostać spełniony aby operacja wskazywana przez komunikat została wykonana.



Komunikat – iteracje

Iteracja – wielokrotne, policzalne powtórzenie pojedynczego komunikatu



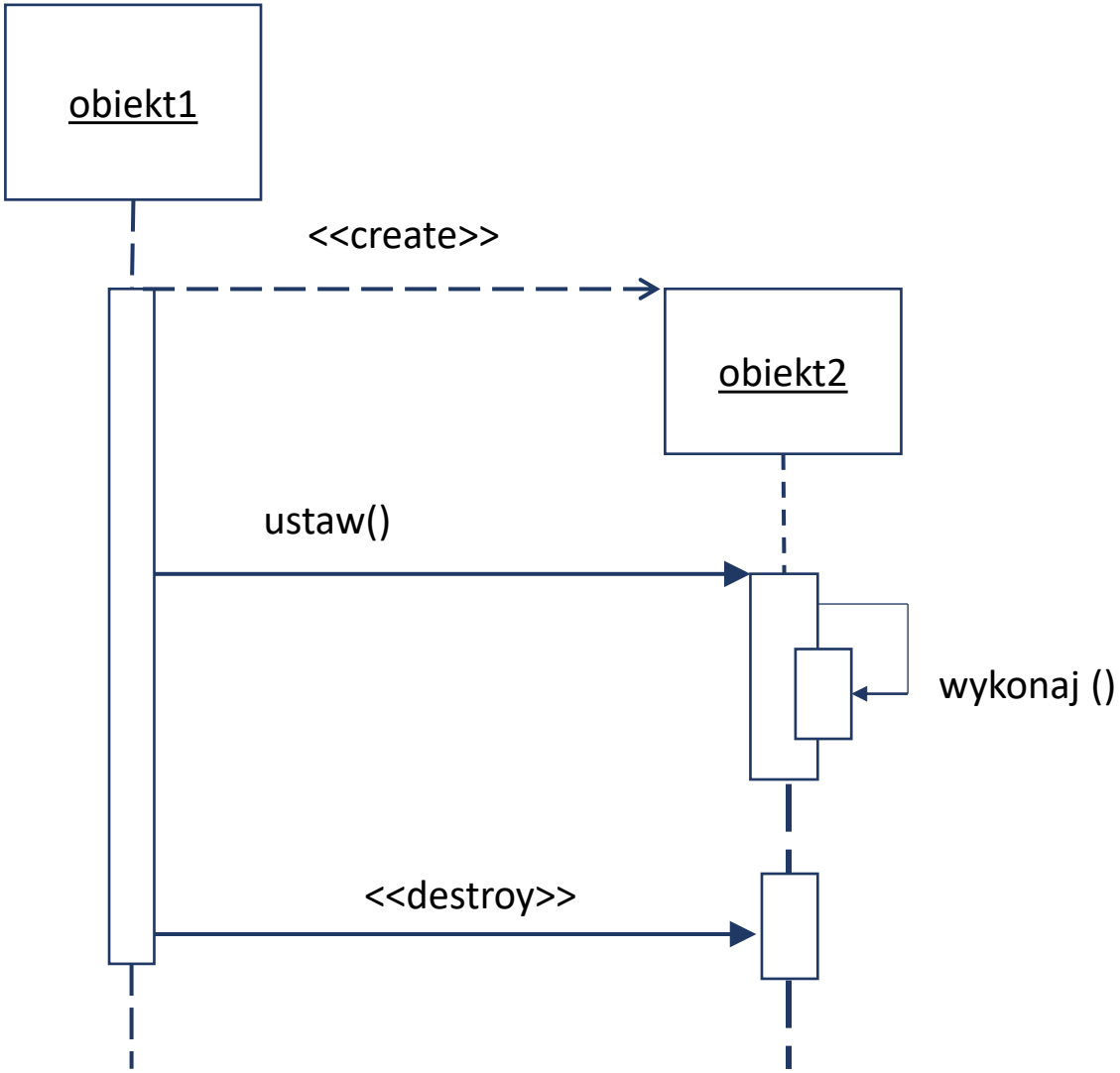
Tworzenie i niszczenie obiektów

Ostatnim rodzajem komunikatów są komunikaty, które tworzą obiekt lub kasują go.

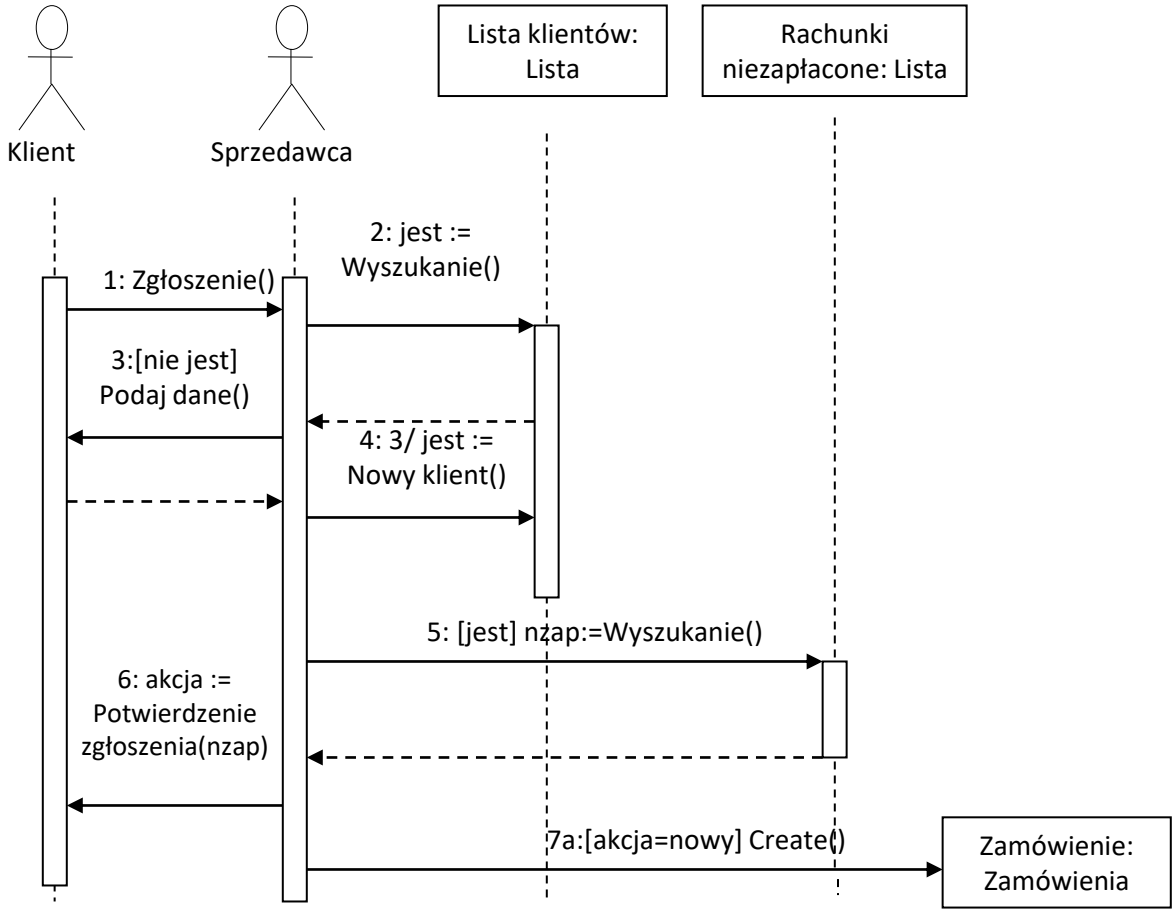
- Komunikat tworzący obiekt - stereotyp <<create>>, wygenerowany obiekt różni się od pozostałych (istniejących przez cały czas trwania scenariusza) tym, że jest umieszczony niżej względem nich.

- Komunikat niszczący – sterotyp <<destroy>> jest odpowiedzialny za zniszczenie obiektu, co jest zobrazowane znakiem X na końcu linii życia obiektu

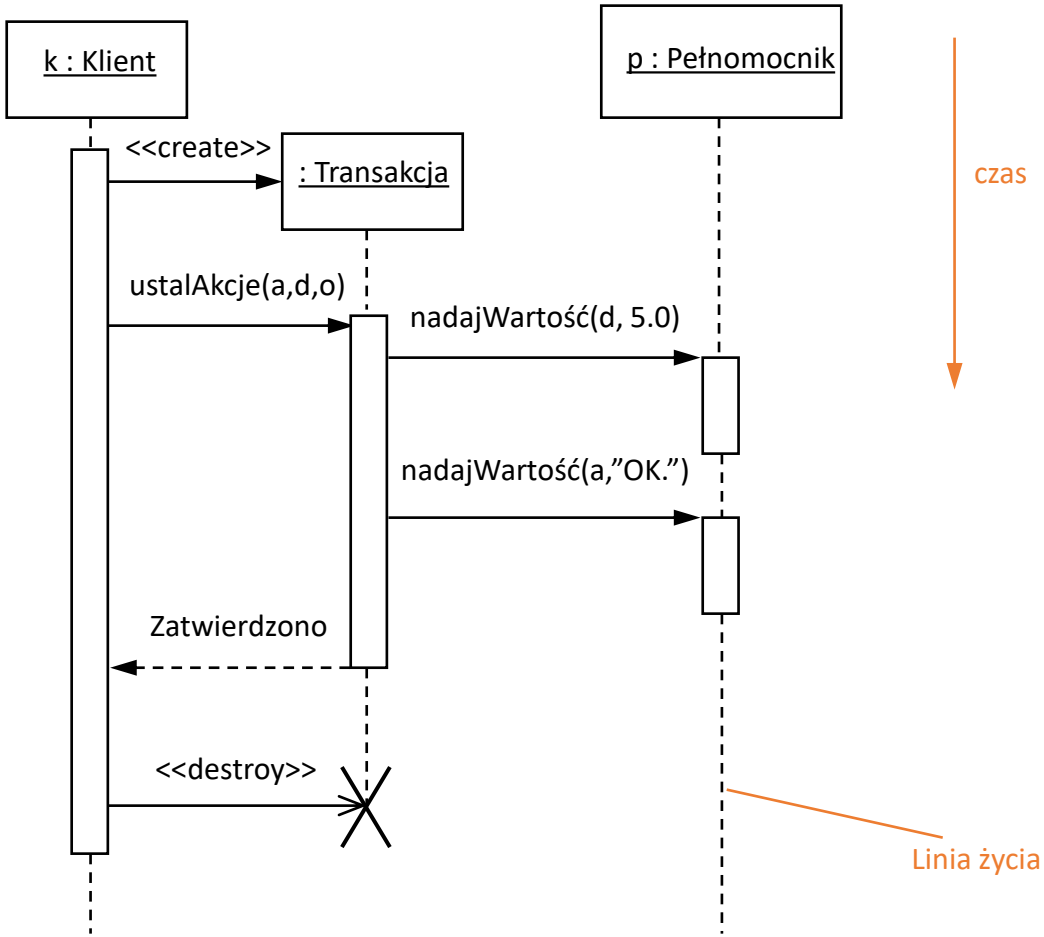
Tworzenie i niszczenie obiektów – przykład



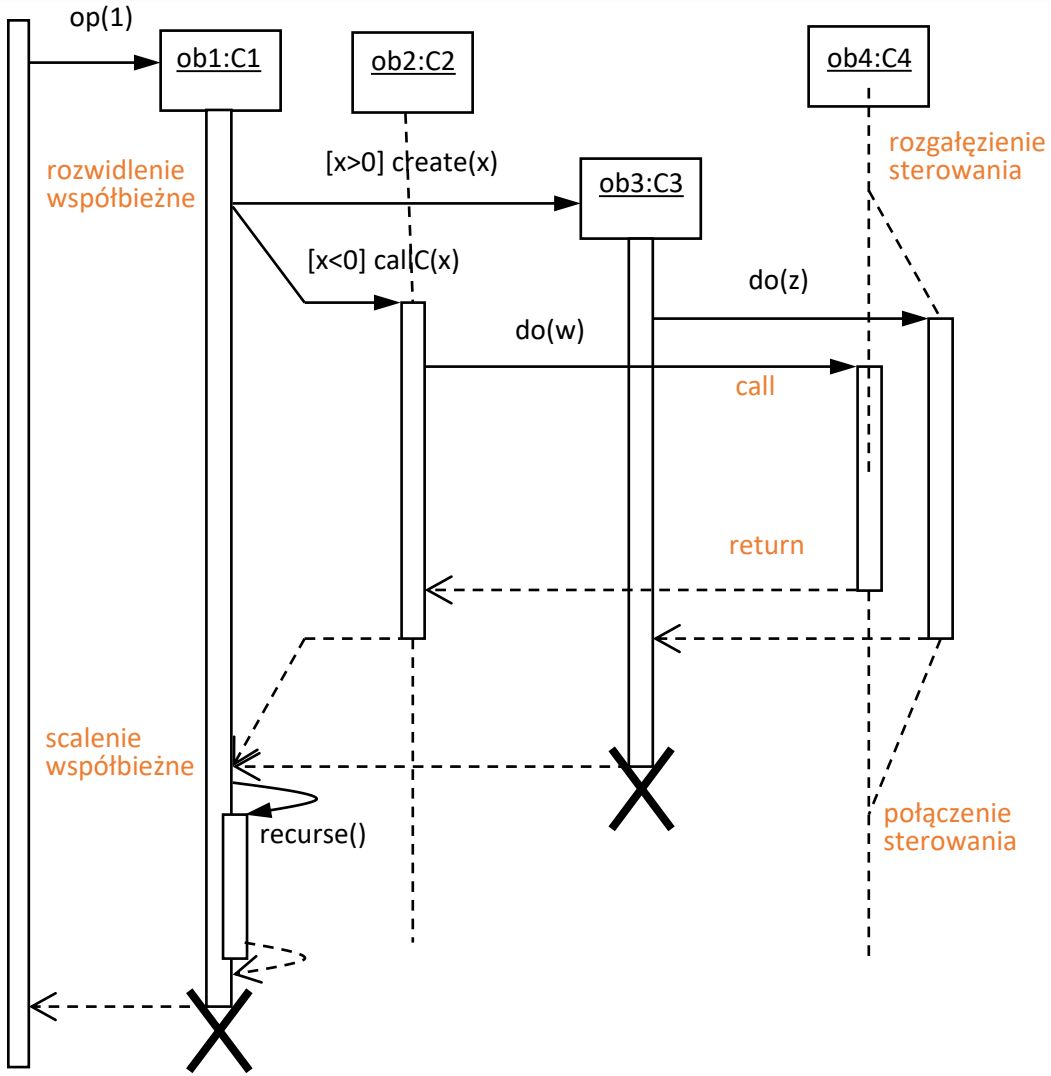
Przykład


Klient
Sprzedawca
Lista klientów: Lista
Rachunki niezapłacone: Lista
1: Zgłoszenie()
2: jest := Wyszukanie()
3:[nie jest] Podaj dane()
4: 3/ jest := Nowy klient()
5: [jest] nzap:=Wyszukanie()
6: akcja := Potwierdzenie zgłoszenia(nzap)
7a:[akcja=nowy] Create()
Zamówienie: Zamówienia

Przykład kolejny

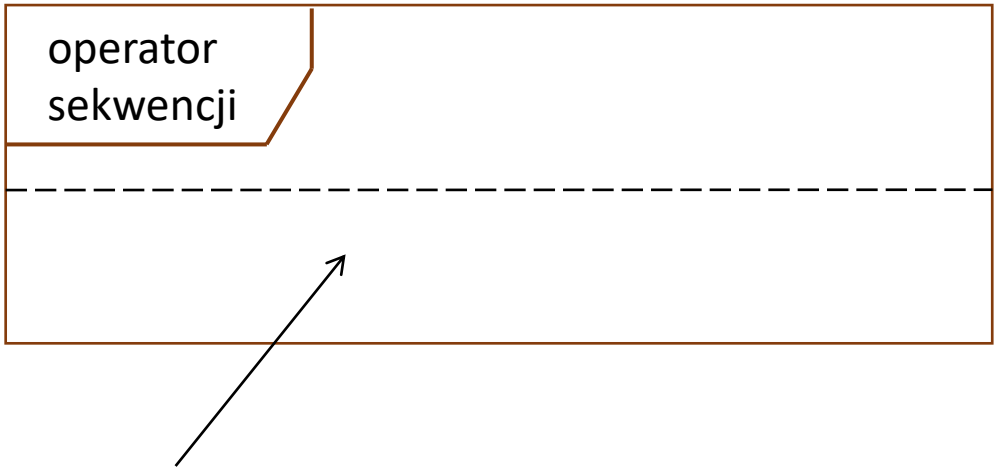


Przykład kolejny



Wyodrębniony fragment – blok

Blok – konceptualnie zamknięta część diagramu sekwencji, która rozszerza możliwości obejmowanego przez siebie obszaru diagramu sekwencji. Fragment może zawierać w sobie pętle, powtórzenia, scenariusze alternatywne lub wskazywać poziom abstrakcji modelowanego fragmentu systemu.



Rodzaj bloku jest określany poprzez umieszczenie odpowiedniego operatora sekwencji w lewym górnym rogu. Stanowi on sprecyzowanie funkcjonalności realizowanej przez fragment.

Blok – operatory

Alt – alternatywa
Opt – opcja
Break – przerwanie
Loop – iteracja
Par –współbieżność

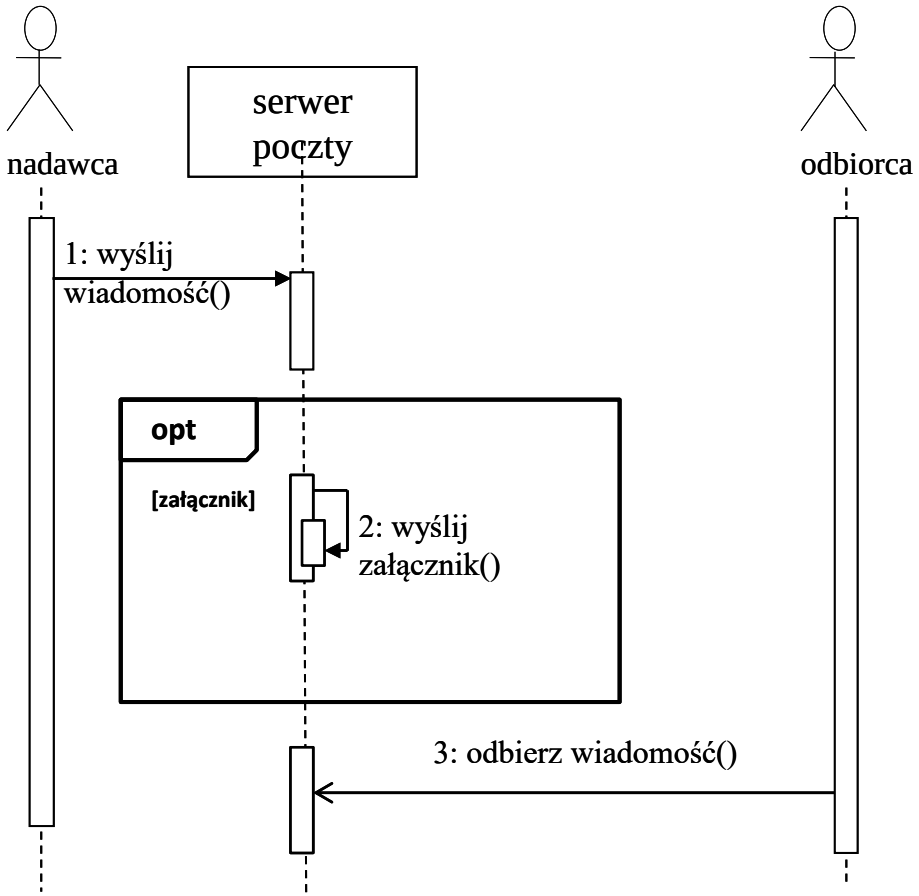
Neg
Critaical
Assert
Consider
Ignore
Strict
Seq

Operator – przykłady

alt – dzieli fragment interakcji zgodnie z warunkami logiki Boole'a na dwa alternatywne scenariusze; każda z alternatyw musi być opatrzona warunkiem dozoru, którego spełnienie gwarantuje wykonanie danej alternatywy.

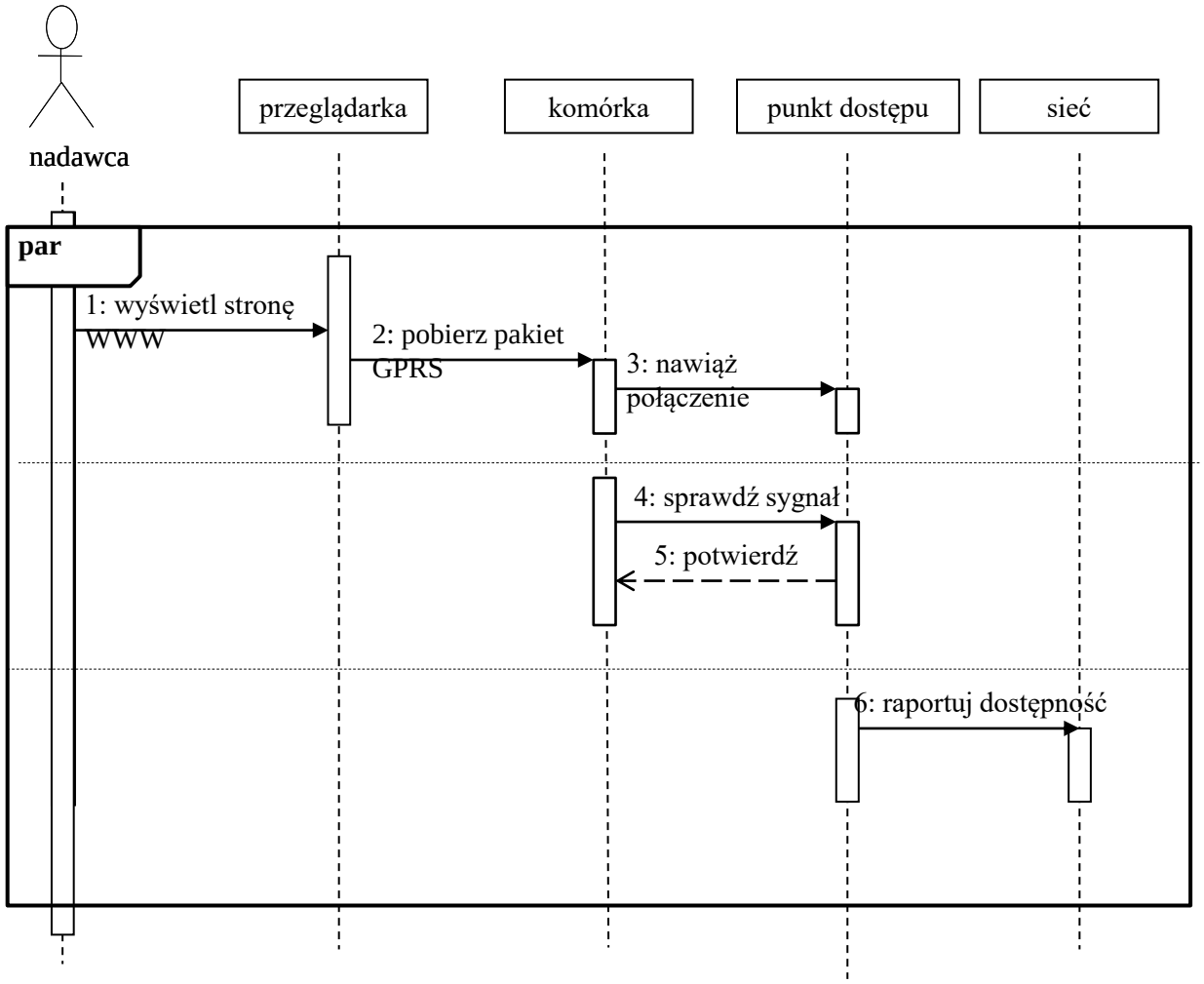
break – wskazuje fragment diagramu sekwencji, który realizowany jest po spełnieniu warunku dozoru; spełnienie warunku dozoru skutkuje wykonaniem sekwencji komunikatów zawartych we fragmencie, a następnie wyjście ze scenariusza; w przypadku, gdy warunek dozoru nie jest spełniony, komunikaty zawarte we fragmencie są pomijane.

Operator – opcjonalność



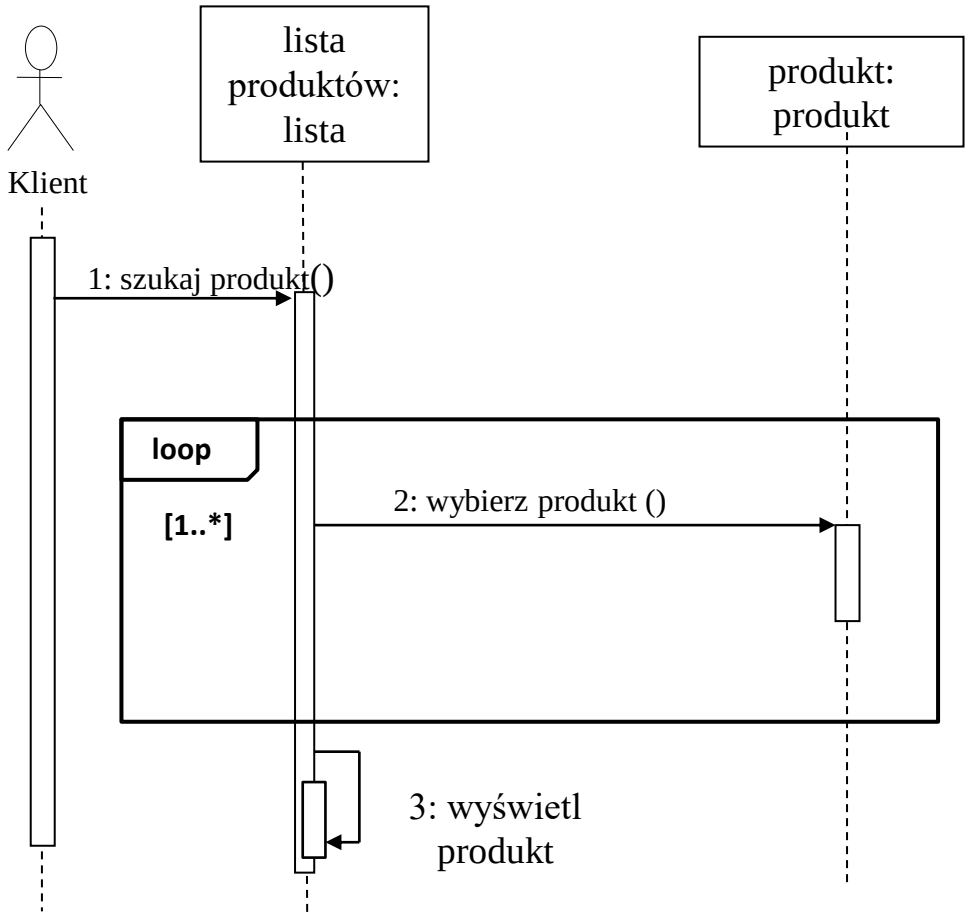
opt – wskazuje opcjonalny fragment interakcji, który jest wykonywany po spełnieniu warunku dozoru.

Operator – współbieżność



par – prezentuje równoległe wykonywanie przepływu wiadomości.

Operator – pętla



loop – powtórzenie fragmentu interakcji. Z reguły jawne określenie krotności wykonania.

Inne operatory

consider – wskazuje fragment z listą nazw komunikatów, które są wyselekcjonowane w tej części interakcji; oznacza to, że mimo innych komunikatów, które znajdują się w danej części interakcji, pokazane zostaną tylko te, które są wypisane w za słowem kluczowym consider.

critical – obszar krytyczny wskazuje, że dany fragment diagramu sekwencji nie może być przerwany przez inny proces. Wskazuje obszar o najwyższym priorytecie w realizacji interakcji. Może służyć do określania priorytetów fragmentów współbieżnych Instancje wchodzące w skład obszaru krytycznego nie mogą uczestniczyć w innych interakcjach.

Inne operatory

ignore – wskazuje, że w tym fragmencie interakcji znajdują się wiadomości, które zostały pominięte, gdyż ich widoczność nie zmienia zachowania systemu; zignorowane wiadomości są wylistowane po słowie ignore. Komunikaty te nie są istotne w trakcie wykonywania obejmowanego fragmentu. Ich wykonanie nie warunkuje wykonania całego fragmentu

assert – prezentuje fragment interakcji, który musi być wykonany zgodnie z założonymi warunkami i komunikatami. Wyodrębnia wykonanie sformalizowanego algorytmu, twierdzenia.

Inne operatory

neg – fragment prezentujący jedną lub więcej wiadomości, które są prawdopodobnie nieprawidłowe. Zawiera sekwencje czynności wykonywane w sytuacji zaistnienia nieprawidłowości. Podobne do przerwania ale nie zawiera obsługi wyjątku ani nie anuluje dalszych etapów

seq – wskazuje słabo uszczegółowiony fragment sekwencji, tzn. taki, który jest ogólny. Komunikaty w wyodrębnionym fragmencie odnoszące się do różnych linii życia mogą być wykonane w innym porządku niż wskazany na diagramie.

strict – prezentuje szczegółową, całkowitą komunikację pomiędzy obiektami. Komunikaty w wyodrębnionym fragmencie muszą być wykonane w określonym na diagramie porządku

Konstrukcja diagramów sekwencji

1. Określ otoczenie interakcji (system, podsystem, operacja, klasa, scenariusz przypadku użycia, kooperacja).
2. Zidentyfikuj obiekty biorące udział w interakcji i rozmieść je od lewej do prawej.
3. Narysuj linie życia dla każdego obiektu. Specyfikuj również linie życia obiektów tworzonych/niszczonych w trakcie interakcji.
4. Komunikat inicjujący interakcję umieść u góry diagramu. Stosuj zasadę kolejności według miejsca na linii życia.
5. Specyfikację zagnieżdżenia komunikatów lub obliczenia specyfikuj przez ośrodek sterowania.
6. Stosuj znaczniki czasowe do specyfikacji ograniczeń czasowych.
7. Jeśli chcesz wyspecyfikować warunki ograniczające przepływ sterowania - zapisz odpowiednio warunki wstępne i końcowe związane z każdym komunikatem.

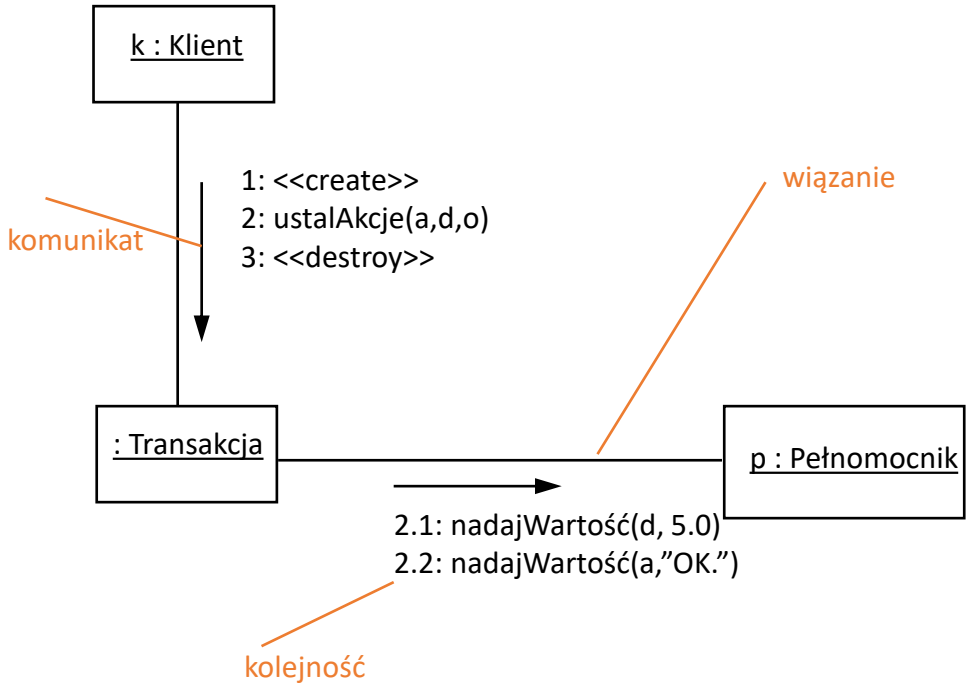
Diagram kooperacji

Diagramy przebiegu i kooperacji są równoważne, gdyż specyfikują tę samą informację w metamodelu UML.

Zastosowania diagramów interakcji

1. Modelowanie przepływu sterowania z uwzględnieniem kolejności komunikatów w czasie (diagramy przebiegu).
2. Modelowanie przepływu sterowania z uwzględnieniem organizacji strukturalnej obiektów (diagramy kooperacji).

Diagram kooperacji – przykład



Konstrukcja diagramów kooperacji

1. Określ otoczenie interakcji (system, podsystem, operacja, klasa, scenariusz przypadku użycia, kooperacja).
2. Zidentyfikuj obiekty biorące udział w interakcji i umieść je jako wierzchołki w diagramie kooperacji. W środkowej części umieść najważniejsze obiekty, w miejscach otaczających mniej ważne.
3. Określ własności początkowe każdego obiektu. Jeśli wartości atrybutów, metki, stan lub rola ulegają zmianie podczas interakcji, umieść na diagramie kopię tego obiektu, uwzględniając te nowe własności.
4. Określ wiązania między obiektami oraz przekazywane komunikaty.
5. Zaczynając od komunikatu inicjującego interakcję skojarz wszystkie komunikaty z odpowiednimi wiązaniami, nadając im stosowne numery porządkowe.
6. Dodaj znaczniki czasu i określ ograniczenia czasowe.
7. Jeśli przepływ sterowania ma być bardziej formalny - zapisz warunki wstępne i końcowe związane z każdym komunikatem.