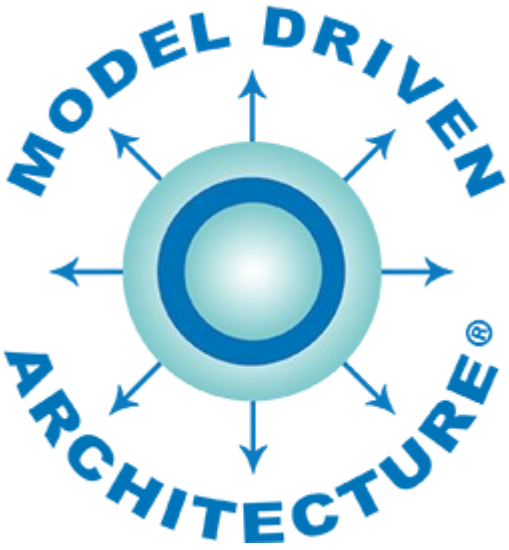


Architektura Sterowana Modelem

Model Driven Architecture

prezentacja 3

Język UML – diagramy zachowań

wersja 1.0

dr inż. Paweł Głuchowski

Wydział Informatyki i Telekomunikacji, Politechnika Wrocławska

Treść prezentacji

1. Język UML (i SysML)
2. Diagram wymagań
3. Diagram przypadków użycia
4. Diagram czynności (aktywności)
5. Diagram stanów
6. Diagram komunikacji
7. Diagram sekwencji
8. Diagram przeglądu interakcji
9. Diagram czasowy

1

Język UML (i SysML)

UML

- **Unified Modeling Language (UML)**
  – graficzny język modelowania i analizowania systemów.

SysML

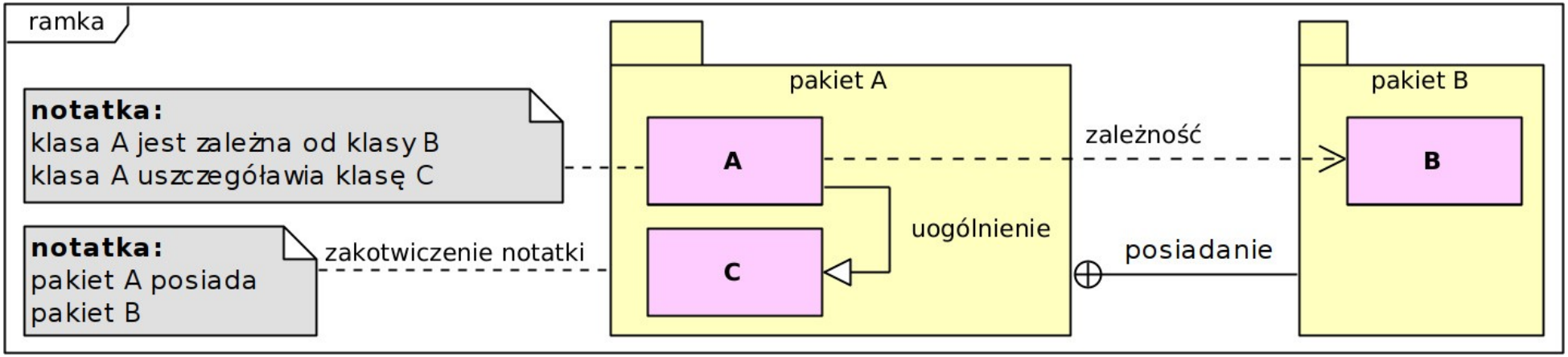
- **Systems Modeling Language (SysML)**
  – profil, rozszerzenie języka UML.

UML i SysML

- Służą do specyfikacji, analizowania, projektowania i weryfikowania złożonych systemów, w tym systemu informatycznego (SI):
  - definicja struktury i zachowania SI i jego części na różnych poziomach abstrakcji i na różnych etapach cyklu życia oprogramowania: pół-/formalna, standaryzowana, graficzna i precyzyjna;
  - wsparcie pracy zespołowej nad SI;
  - dekompozycja SI (jego struktury i zachowania) – pokonanie jego złożoności;
  - tworzenie i stosowanie wzorców projektowych;
  - projektowanie testów oprogramowania.
- Posiadają elastycznie określoną składnię (reprezentację graficzną) i semantykę (jej zastosowanie).
  - W elemencie diagramu ma znaczenie: jego kształt, położenie, sposób powiązania z innym elementem.
  - W elemencie diagramu NIE ma znaczenia: jego kolor (z wyjątkiem koloru grotów kilku związków – biały / czarny).

Elementy UML stosowane na różnych diagramach

- **notatka** /note/ – słowne wyjaśnienie diagramu lub elementu, do którego jest zakotwiczona;
- **ramka** /frame/ – semantyczne grupowanie elementów;
- **pakiet** /package/ – funkcjonalne grupowanie elementów;
- **relacja zależności** /dependency/ – wskazujący element jest zależny od wskazanego elementu;
- **relacja uogólnienia** /generalization/ – wskazujący element jest uszczegółowieniem wskazanego elementu;
- **relacja posiadania** /containment/ – element z grotem ⊕ na końcu relacji zawiera element z drugiego jej końca.



2

Diagram wymagań

Diagram wymagań /requirement diagram/

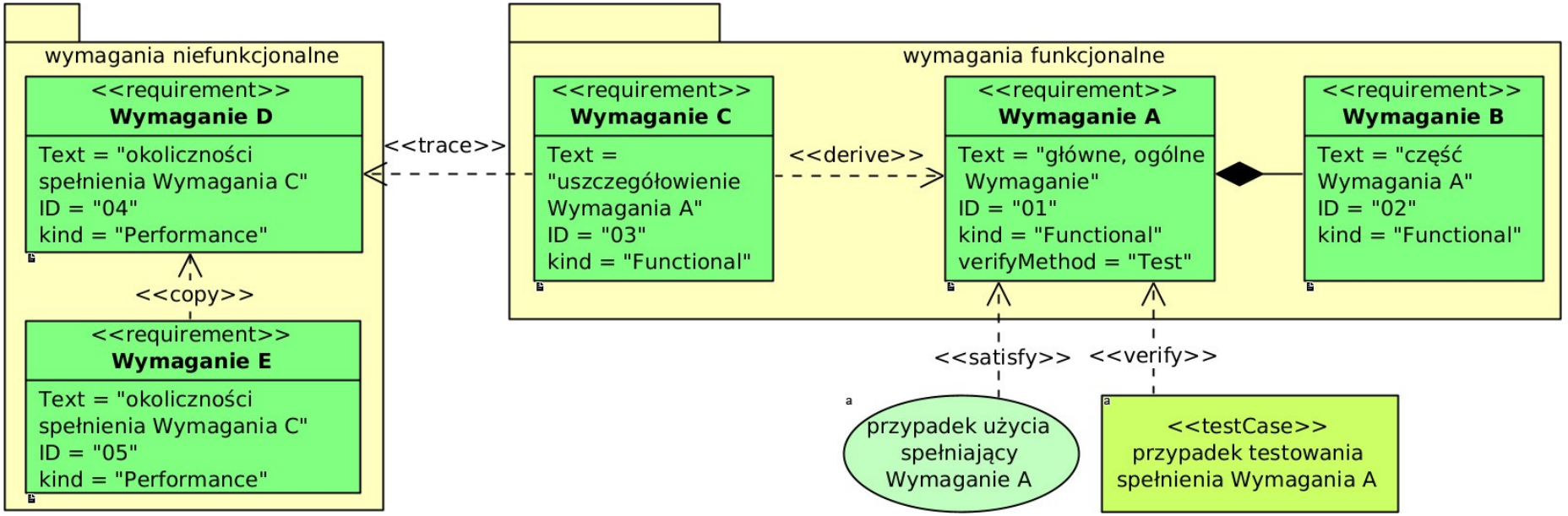
- Modeluje wymagania funkcjonalne i niefunkcjonalne stawiane systemowi:
  - warunki lub cele do spełnienia lub spełniania,
  - relacje między nimi.
- **Wymaganie** /requirement/ to NIE proces realizacji jakiegoś celu, ale sam cel.
- Opracowany na podstawie słownej specyfikacji wymagań.
- W języku zrozumiałym przez klienta – użytkownika SI.

Wymaganie funkcjonalne

- Opisuje zadanie, które system musi wykonać lub wykonywać
  - co system ma osiągnąć lub w jakim być stanie.

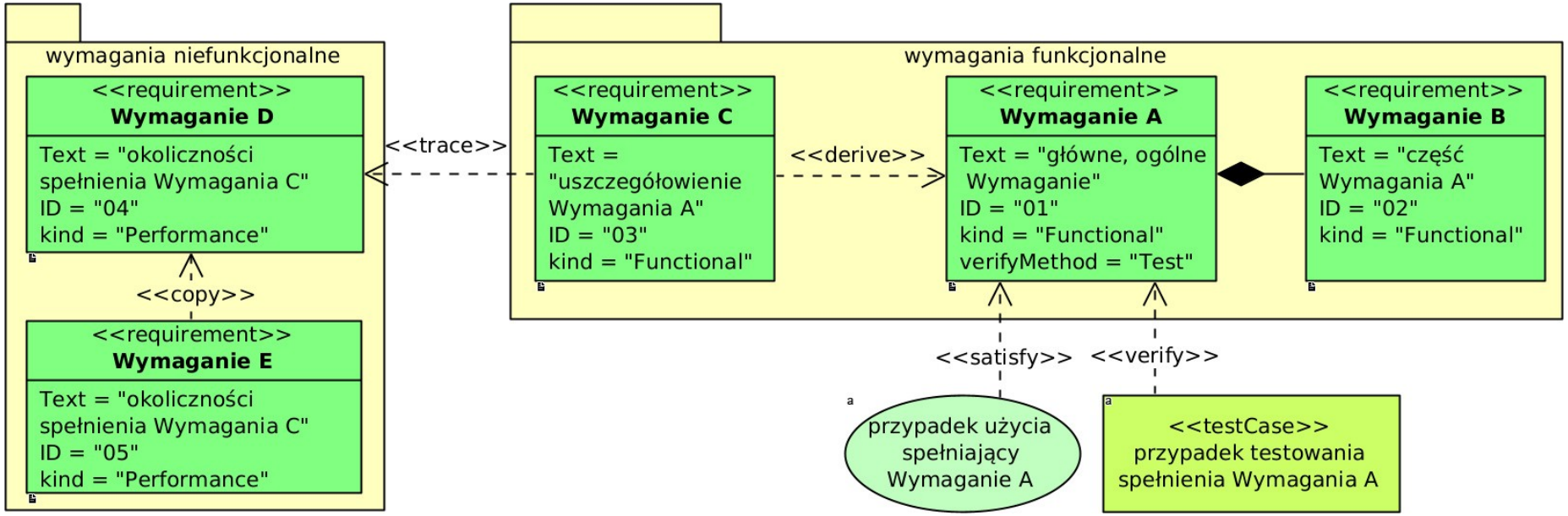
Wymaganie niefunkcjonalne

- Opisuje jakościowe kryteria efektywności zadań systemowych
  - jak lub w jakich warunkach system ma to osiągnąć
    (przy jakich ograniczeniach sprzętowych, organizacyjnych, prawnych…).



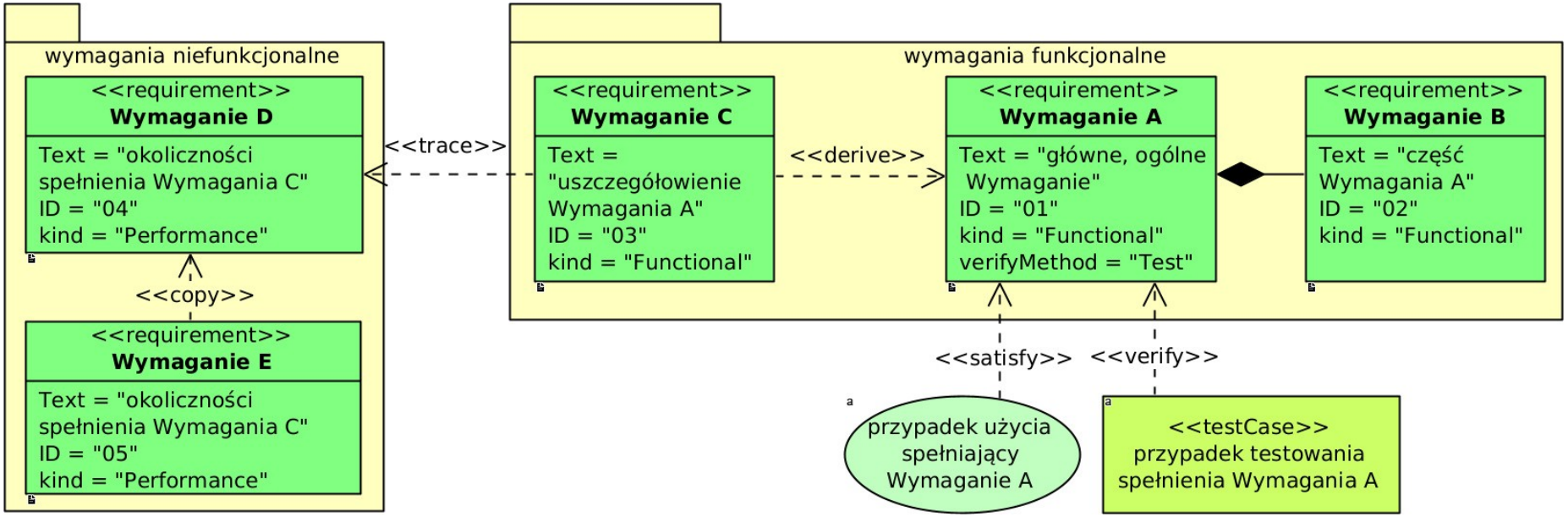
Przypadek testowania *«testCase»*

- Opisuje sposób sprawdzenia, czy wymaganie zostało spełnione.
- Modeluje: test realizacji przypadku użycia, test systemu (np. funkcjonalny), test metody klasy (jednostkowy) i inne.
- Zakłada stan początkowy przedmiotu testu.
- Zawiera scenariusz testowania w postaci tekstowej (pseudokod) lub graficznej (powiązany z nim diagram czynności).
- Zakłada uzyskanie określonego artefaktu lub stanu końcowego.



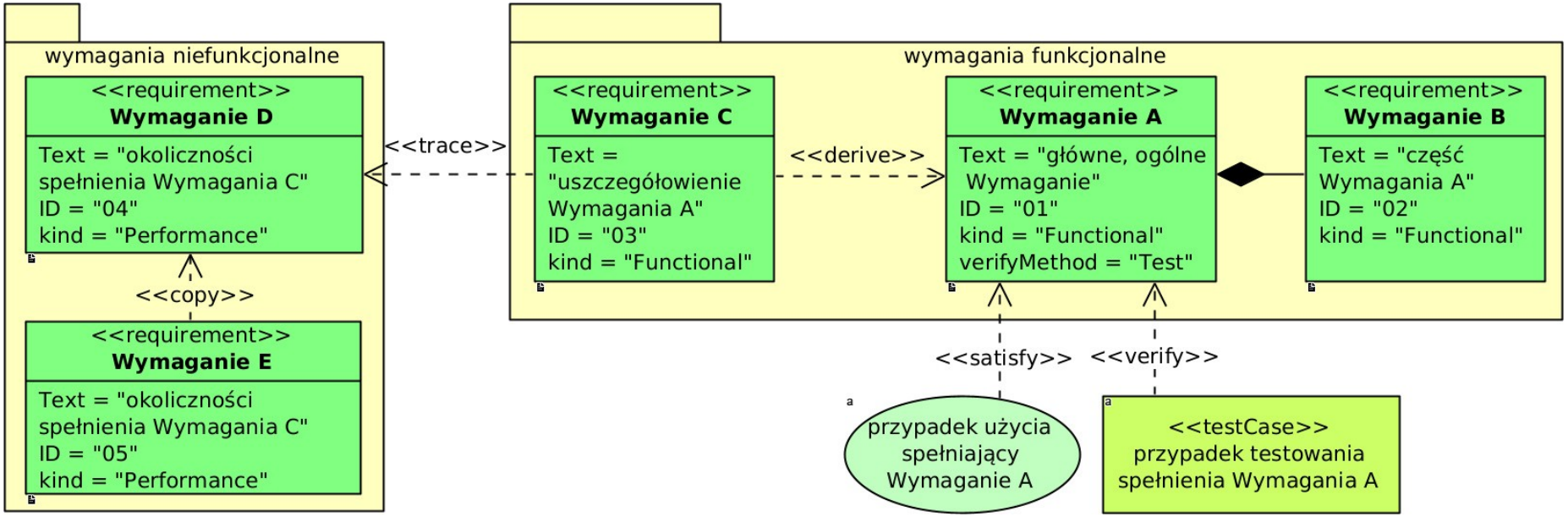
Relacje powstałe w analizie wymagań

- **Relacja kompozycji** /composition/ – łączy wymagania w związek całość-część (grot ♦ jest przy „całości”).
- **Relacja wyprowadzenia *«derive»*** – wyprowadzenie wskazywanego wymagania ze wskazującego (uszczegóławianego) wymagania.
- **Relacja śladu *«trace»*** – słabsze uzależnienie (uwarunkowanie) wskazującego wymagania przez wskazane wymaganie, będące:
  - ograniczeniem spełnienia wskazującego wymagania,
  - wcześniej spełnionym wymaganiem dla wskazującego wymagania.



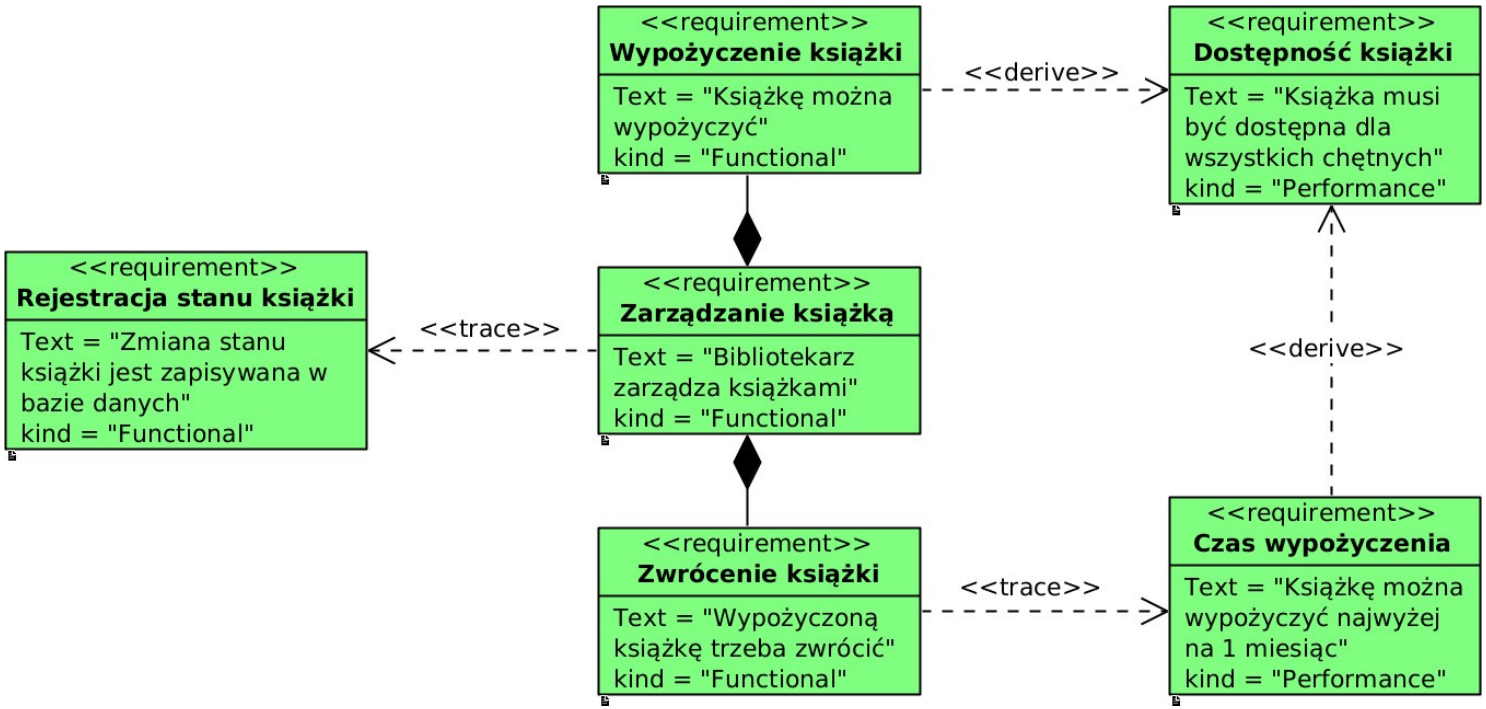
Relacje powstałe w analizie wymagań

- **Relacja kopii *«copy»*** – utworzenie wskazującego wymagania jako kopii (aliasu) wskazanego wymagania.
  - Kopia ma inną nazwę, ale ten sam opis.
  - Kopię można zmienić tylko przez zmianę oryginału.
- **Relacja spełnienia *«satisfy»*** – spełnienie wskazywanego wymagania przez wskazujący przypadek użycia.
- **Relacja sprawdzenia *«verify»*** – sprawdzenie spełnienia wskazywanego wymagania przez wskazujący przypadek testowania.



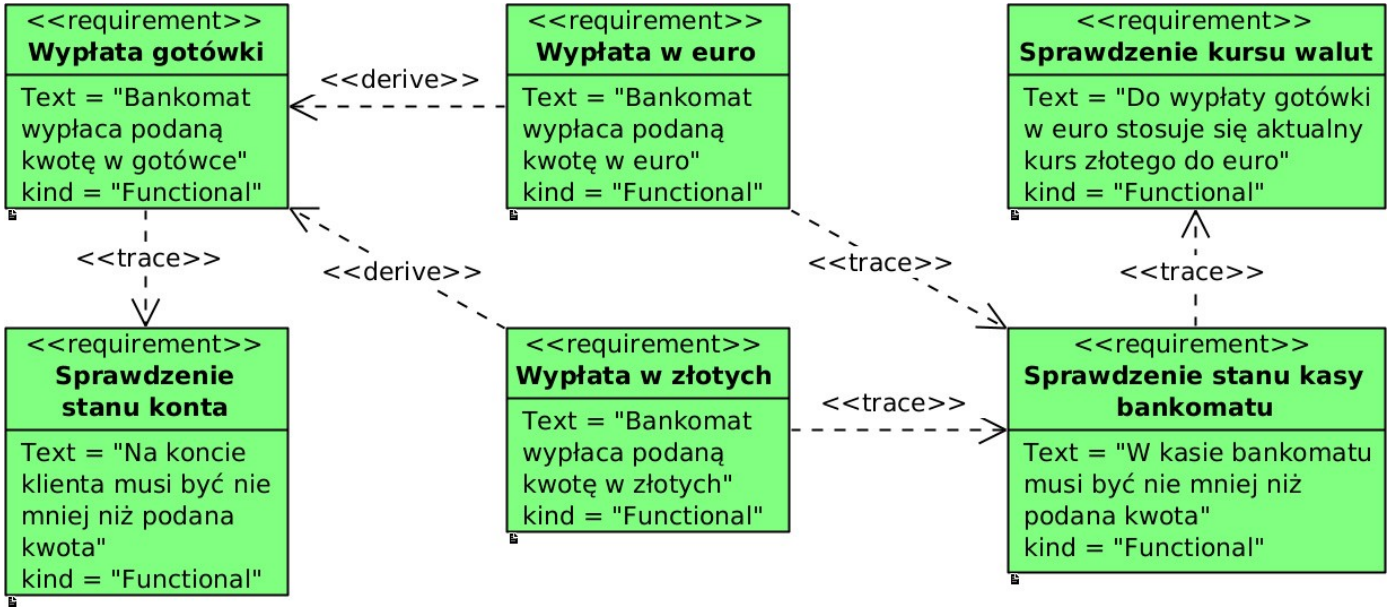
Przykład relacji kompozycji, śladu i wyprowadzenia

- Zarządzanie książką to (kompozycja) jej wypożyczenie lub zwrócenie.
- Zarządzanie książką prowadzi do (*«trace»*) rejestracji jej stanu.
- Wypożyczenie książki wynika z (*«derive»*) jej dostępności.
- Zwrócenie książki wymaga (*«trace»*) zachowania czasu jej wypożyczenia.
- Czas wypożyczenia książki wynika z («derive») zachowania jej dostępności.



Przykład relacji śladu i wyprowadzenia

- Wersje wypłaty gotówki (*«derive»*) to: wypłata w euro i wypłata w złotych.
- Wypłata gotówki wymaga (*«trace»*) sprawdzenia stanu konta.
- Wypłata w euro i wypłata w złotych wymaga (*«trace»*) sprawdzenia stanu kasy bankomatu, które wymaga (*«trace»*) sprawdzenia kursu walut.



3

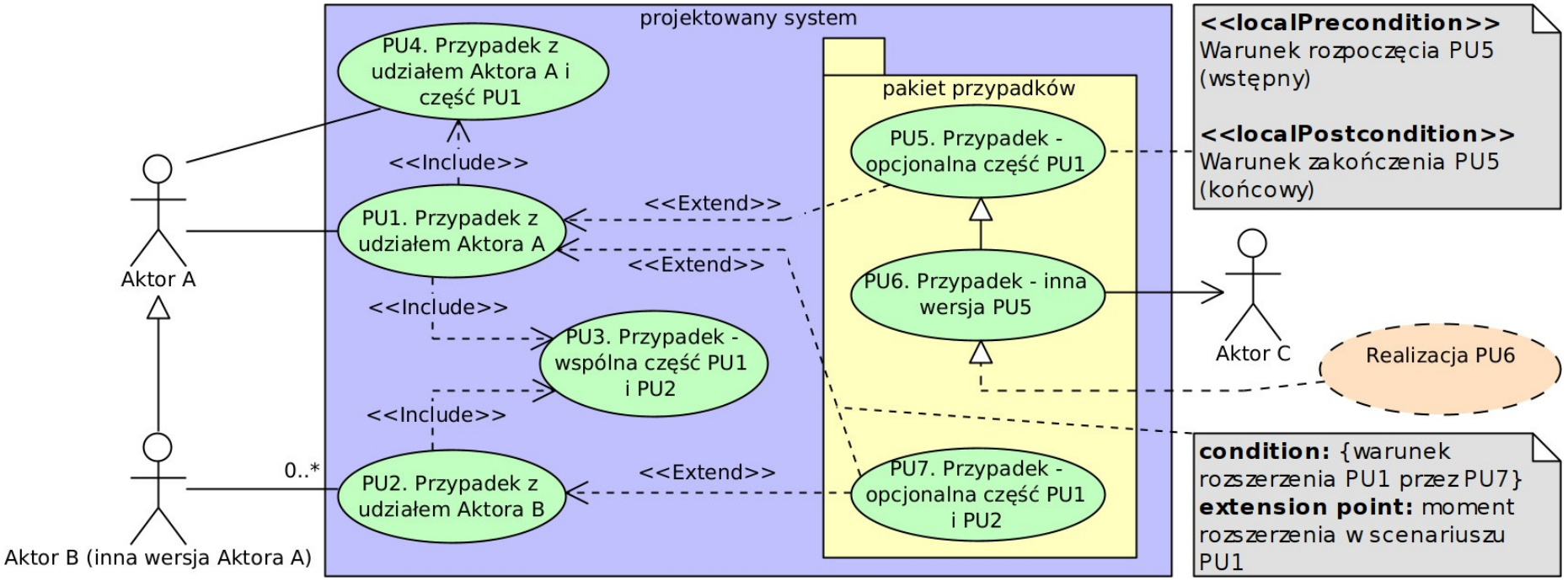
Diagram przypadków użycia

Diagram przypadków użycia /use case diagram/

- Modeluje przypadki użycia i aktorów systemu oraz jego podział na podsystemy.
- Modeluje związki między przypadkami użycia i między przypadkami użycia a aktorami.
- NIE modeluje następstwa (kolejności) realizacji przypadków użycia.
- Opracowany na podstawie tekstowego opisu specyfikacji wymagań oraz diagramów wymagań.
- W języku zrozumiałym przez klienta – użytkownika SI.
- Modeluje zewnętrzną strukturę systemu z jej funkcjonalnością i ogólną koncepcją architektury.

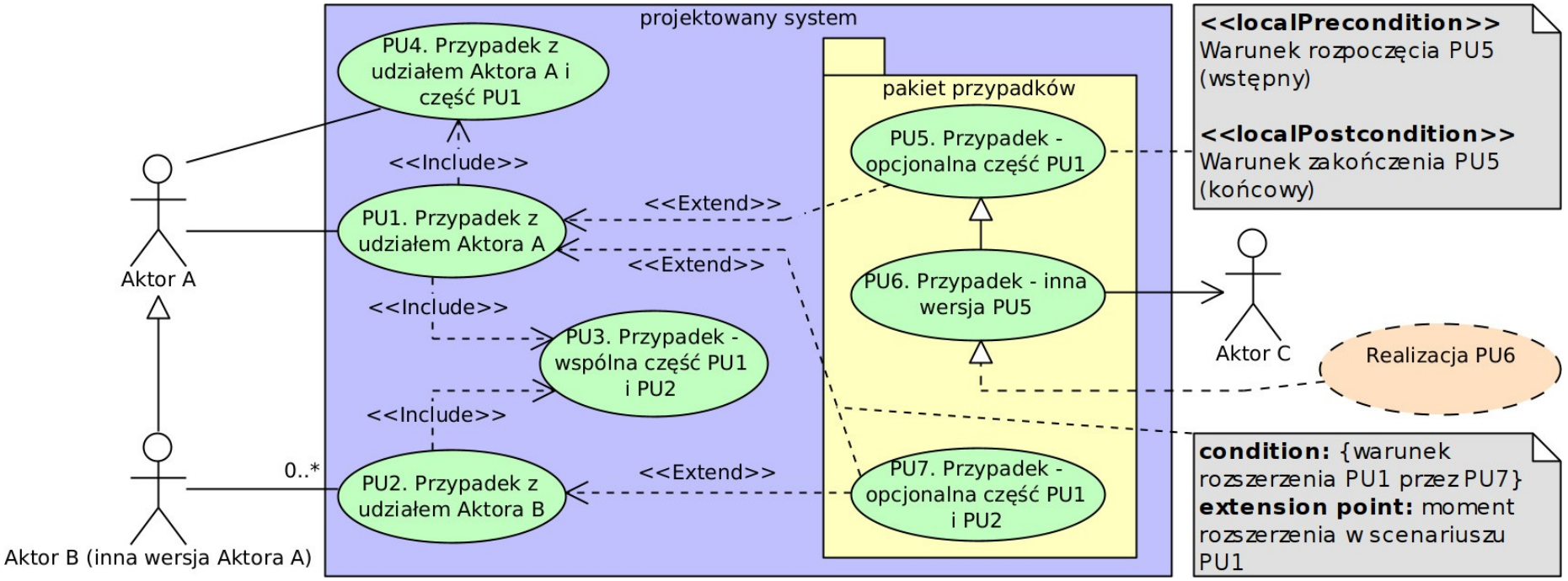
Przypadek użycia /use case/

- Proces biznesowy spełniający wymagania funkcjonalne.
- Zdefiniowany ogólnie, a NIE jako składowa czynność procesu biznesowego.
- Modeluje działania i oczekiwania aktorów w stosunku do SI.
- Przypadki użycia można grupowane w pakiety.



Przypadek użycia

- Może mieć **warunki wstępne** (*«localPrecondition»*) – spełnienie którego wymagania lub wykonanie którego przypadku użycia pozwala rozpocząć wykonanie tego przypadku użycia.
- Może mieć **warunki końcowe** (*«localPostcondition»*) – spełnienie którego wymagania lub wykonanie którego przypadku użycia pozwala zakończyć wykonanie tego przypadku użycia.



Aktor /actor/

- Użytkownik SI (człowiek lub inny system).
  - Aktor-system może (nie musi) być modelowany innym „obrazkiem”.
- Ma **bezpośredni lub pośredni udział** w wykonaniu przypadku użycia.
- Ma **pasywny lub aktywny (inicjujący) udział** w wykonaniu przypadku użycia.

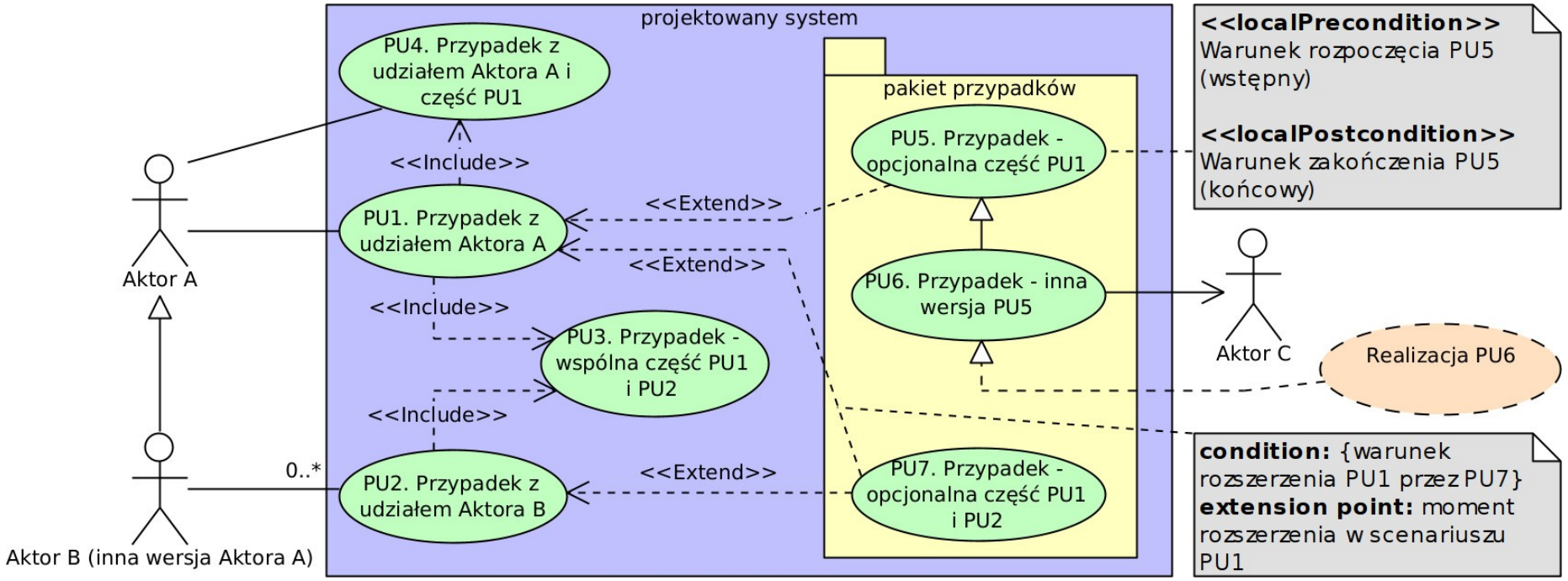
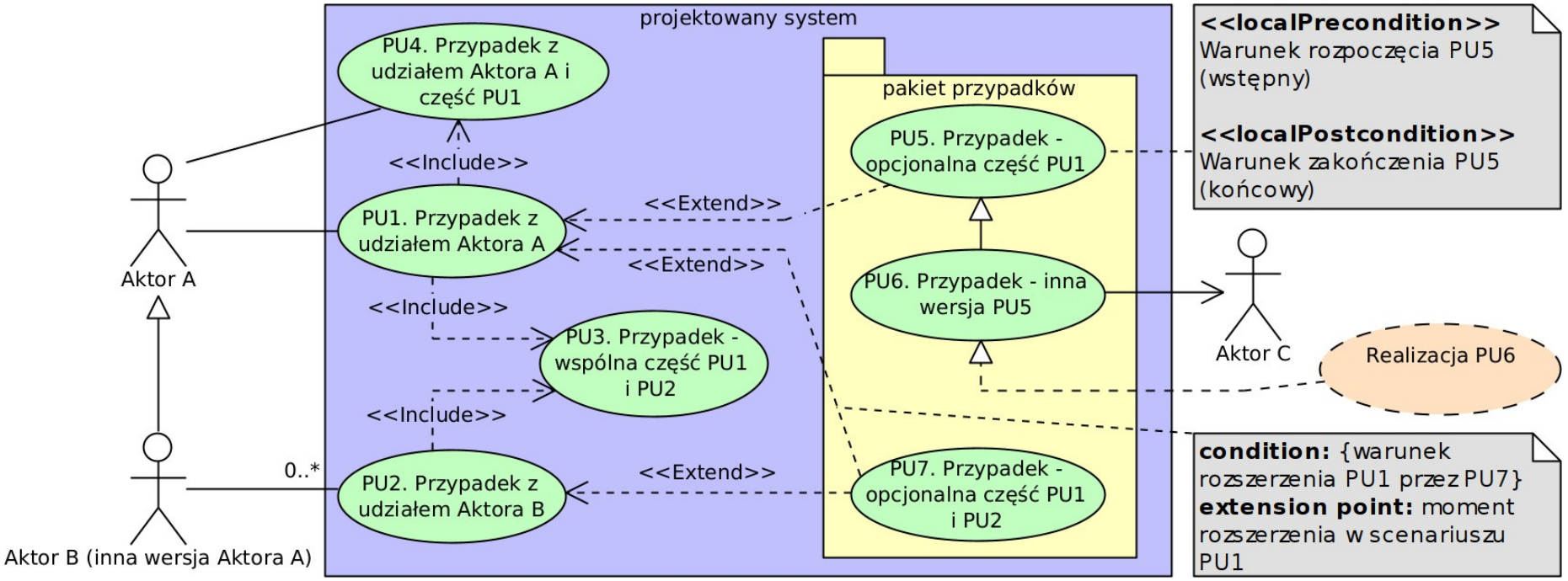


Diagram przypadków użycia

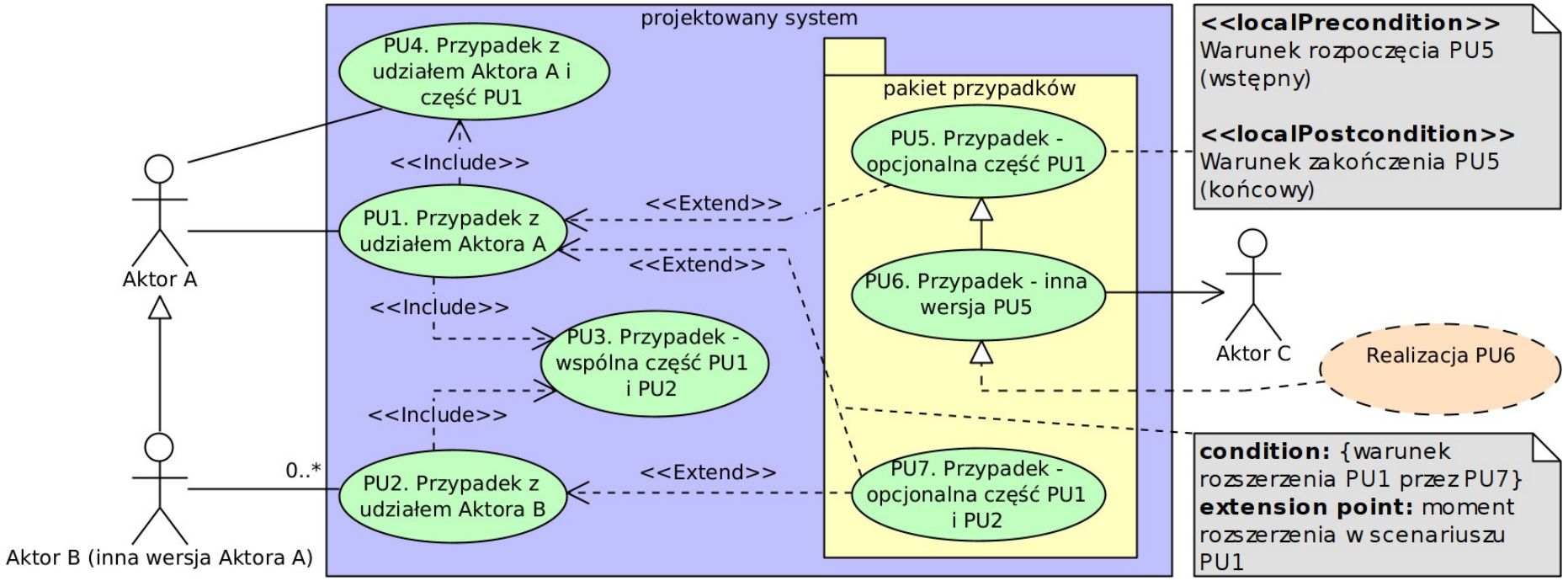
System /system/

- Modelowany SI lub jego podsystem (część innego systemu).
  - Podsystem ma stereotyp *«subsystem»* nad swoją nazwą.
- NIE zawiera innego systemu i NIE zawiera aktorów.
  - Modułową budowę SI można pokazać na diagramie wdrożenia.
- Obejmuje przypadki użycia modelowanego SI i ich pakiety.



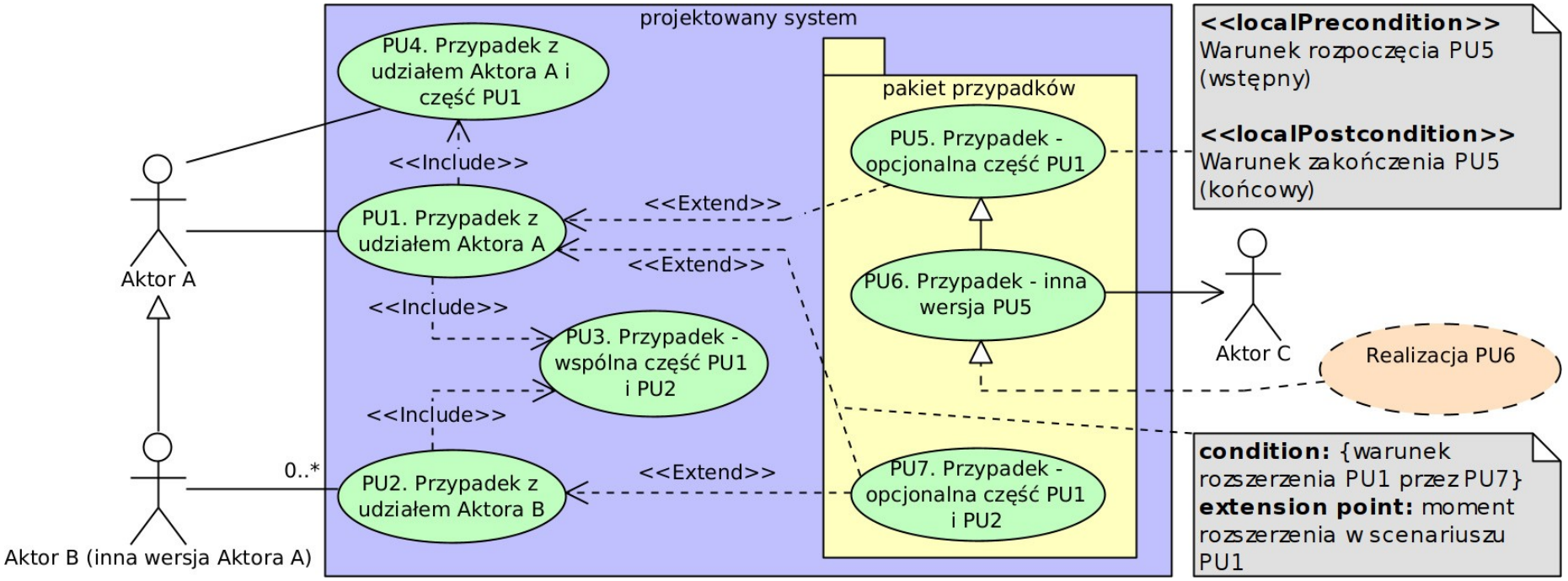
Relacje powstałe w analizie przypadków użycia

- **Relacja uogólnienia** /generalization/ (ciągła linia z grotem Δ).
  - Łączy przypadki użycia lub aktorów w związek wersja podstawowa (ogólna)-wersja pochodna (szczególna).
  - Grot Δ wskazuje na „wersję podstawową”.



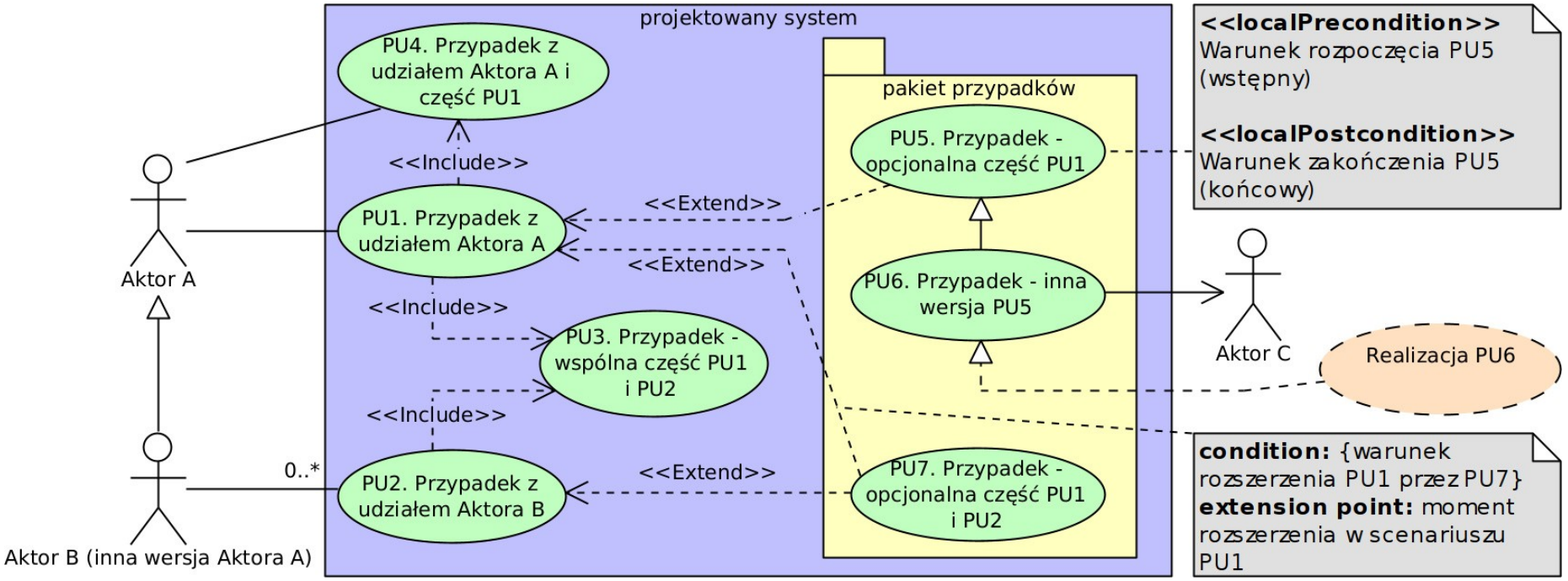
Relacje powstałe w analizie przypadków użycia

- **Relacja asocjacji** /association/ (ciągła linia z otwartym grotem lub bez).
  - Łączy aktora z przypadkiem użycia, gdy aktor inicjuje wykonanie przypadku lub przypadek bezpośrednio komunikuje się z aktorem.
  - Łączy aktorów ze sobą, gdy komunikują się ze sobą poza SI.
  - Pokazuje kierunek komunikacji, gdy jest skierowana.
  - Może wyrażać stosunek ilościowy między połączonymi elementami.



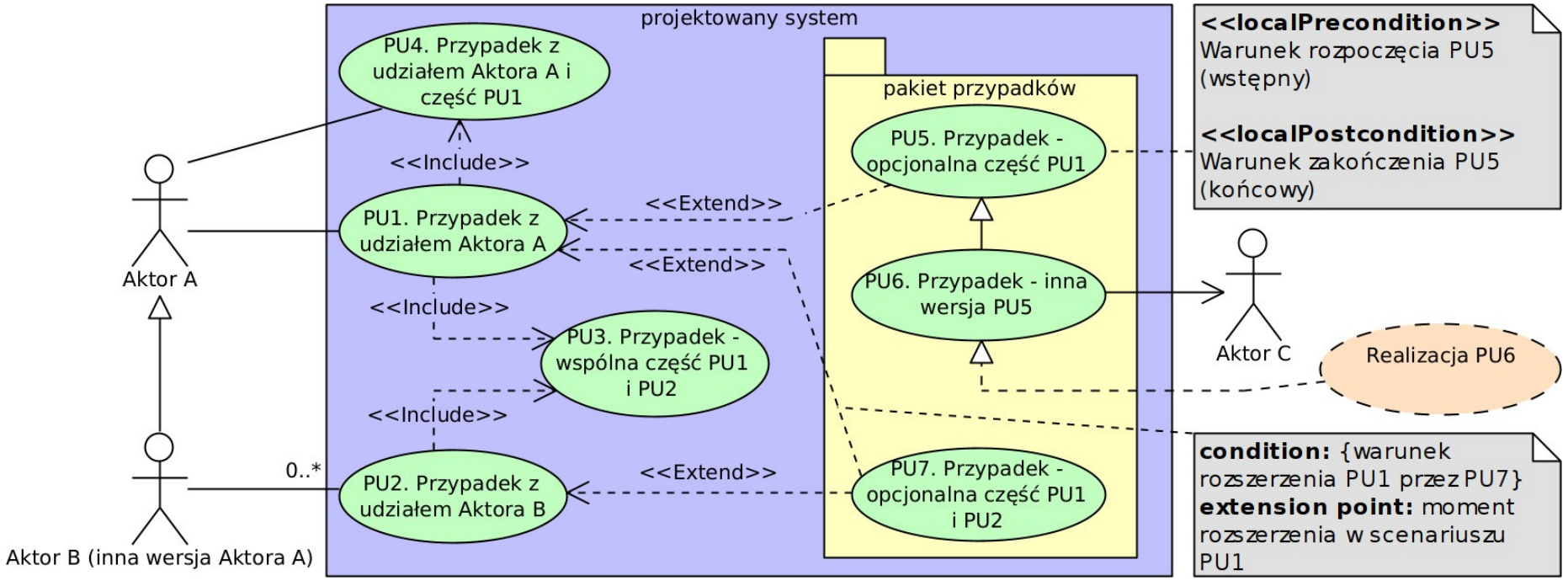
Relacje powstałe w analizie przypadków użycia

- **Relacja zawierania *«include».***
  - Łączy przypadki użycia w związek zależna całość–obowiązkowa część.
  - Wskazuje na część.
  - Wskazany przypadek użycia obowiązkowo zachodzi w ramach wskazującego przypadku.



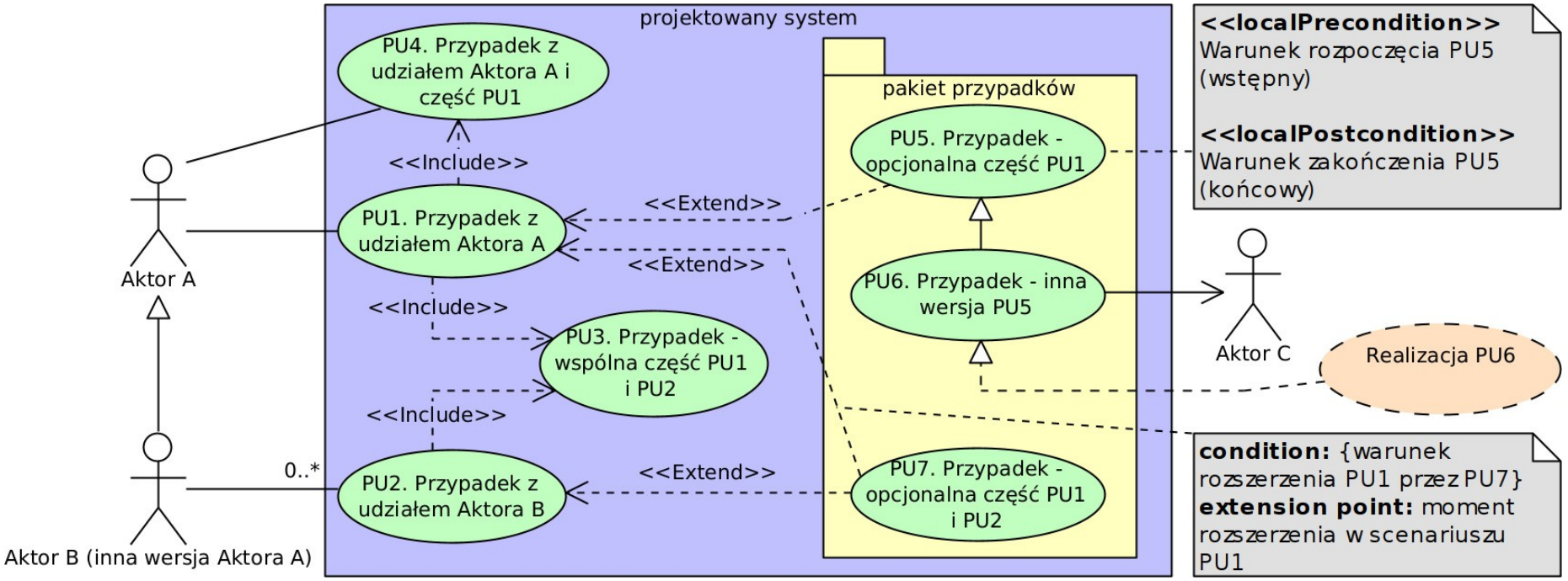
Relacje powstałe w analizie przypadków użycia

- **Relacja rozszerzania *«extend».***
  - Łączy przypadki użycia w związek niezależna całość–opcjonalna część.
  - Wskazuje na całość.
  - Wskazujący przypadek użycia opcjonalnie lub alternatywnie z innym zachodzi w ramach wskazanego przypadku.
  - Warunek i moment rozszerzenia można podać w notatce i w scenariuszu realizacji rozszerzanego przypadku.



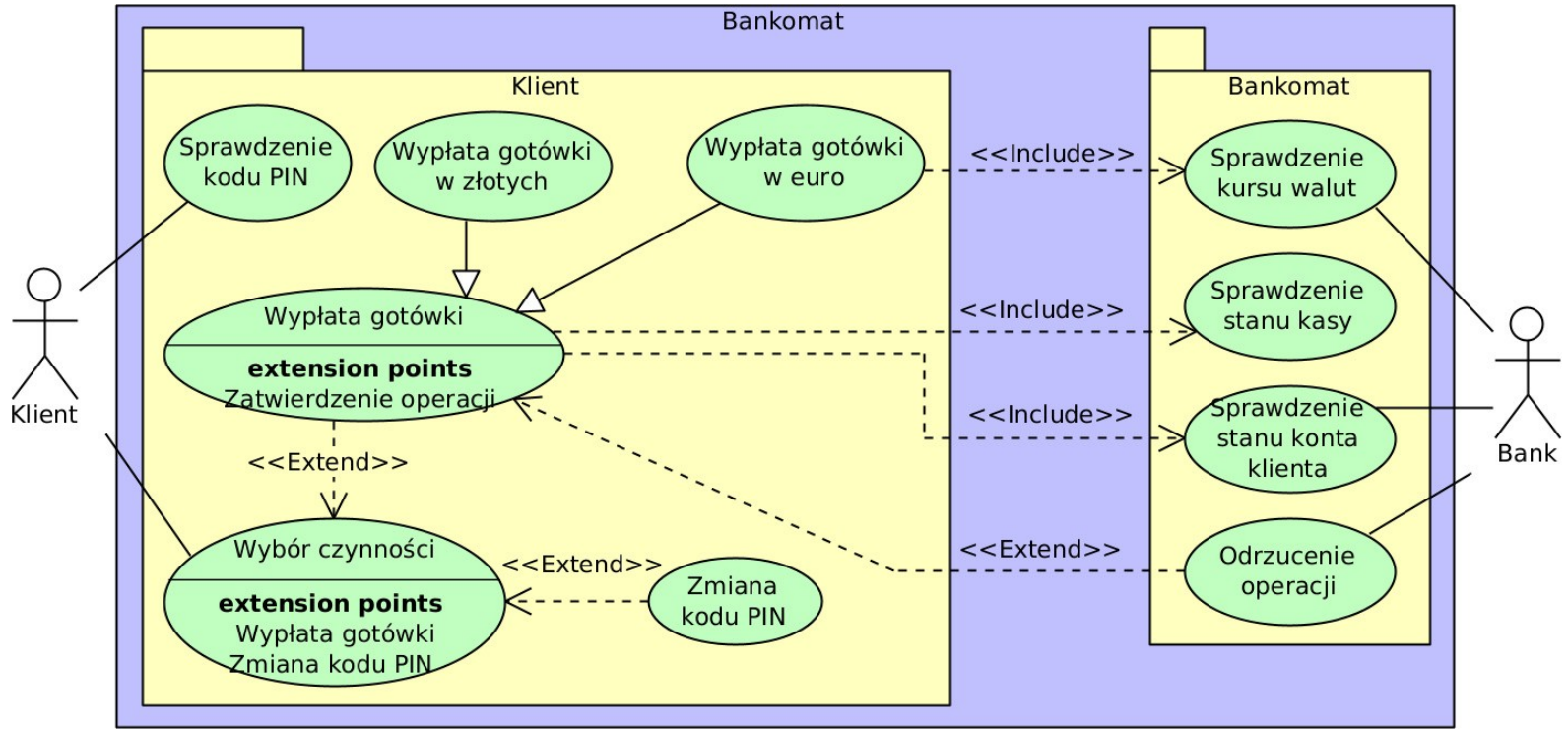
Relacje powstałe w analizie przypadków użycia

- **Relacja realizacji** /realization/ (przerywana linia z grotem Δ).
  - Łączy przypadek użycia ze współdziałaniem w celu jego realizacji.
  - Współdziałanie można pokazać np. na diagramie klas, gdzie będzie zawierać klasy uczestniczące w realizacji zadań tego przypadku użycia.
  - Grot Δ wskazuje na przypadek użycia.



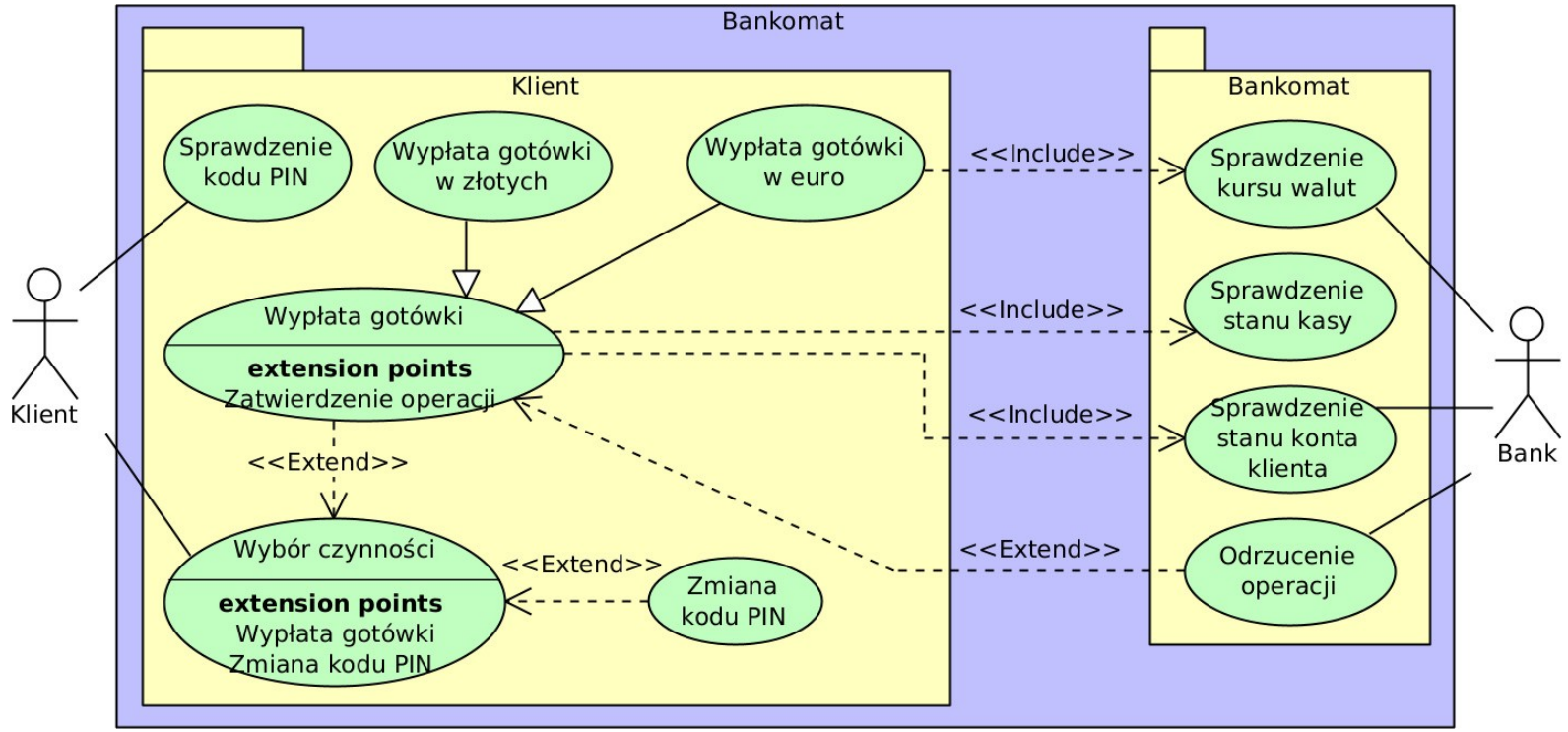
Przykład dekompozycji przypadków użycia

- Modelowany jest system **Bankomat**.
- Pakiety **Klient** i **Bankomat** grupują przypadki użycia na podstawie udziału w nich aktorów.
- **Klient** jest powiązany tylko z przypadkami użycia, które inicjuje.
- **Bank** jest powiązany tylko z przypadkami użycia, które inicjują z nim komunikację (wymianę informacji).



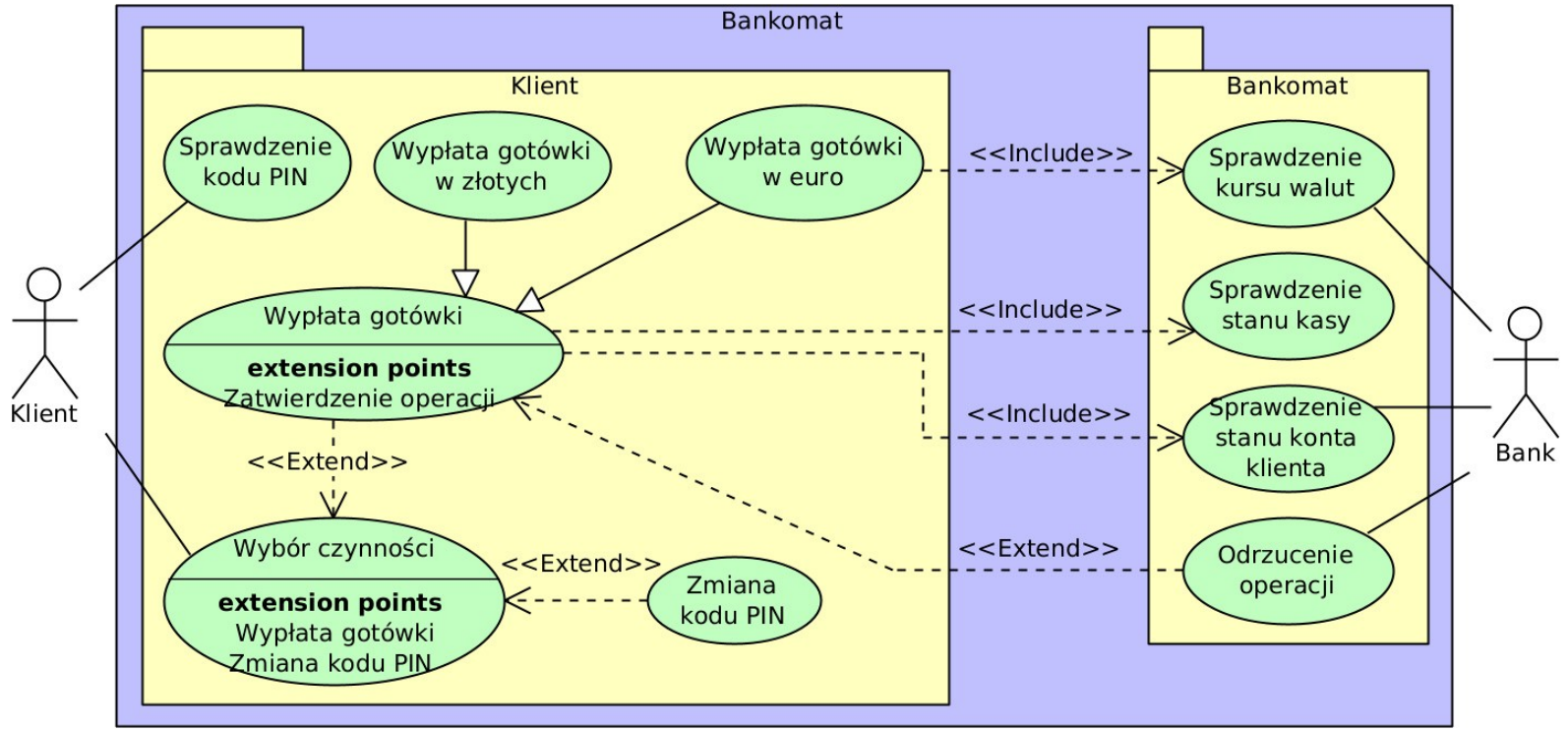
Przykład dekompozycji przypadków użycia

- Klient inicjuje PU **Sprawdzenie kodu PIN**.
- Następnie (diagram nie może pokazać następstwa) **Klient** inicjuje PU **Wybór czynności**, w którym może (*«extend»*) przejść do wykonania PU **Zmiana kodu PIN** lub PU **Wypłata gotówki**.
- Wykonanie PU **Wypłata gotówki** to (uogólnienie) wykonanie PU **Wypłata gotówki w złotych** lub PU **Wypłata gotówki w euro**.



Przykład dekompozycji przypadków użycia

- Wykonanie PU **Wypłata gotówki w euro** zawiera (*«include»*) wykonanie PU **Sprawdzenie kursu walut** z udziałem **Banku**.
- Wykonanie PU **Wypłata gotówki** zawiera (*«include»*) wykonanie PU **Sprawdzenie stanu kasy** i PU **Sprawdzenie stanu konta klienta** z udziałem **Banku**.
- Wykonanie PU **Wypłata gotówki** może zawierać (*«extend»*) wykonanie PU **Odrzucenie operacji** z udziałem **Banku**.



Dokumentacja przypadku użycia

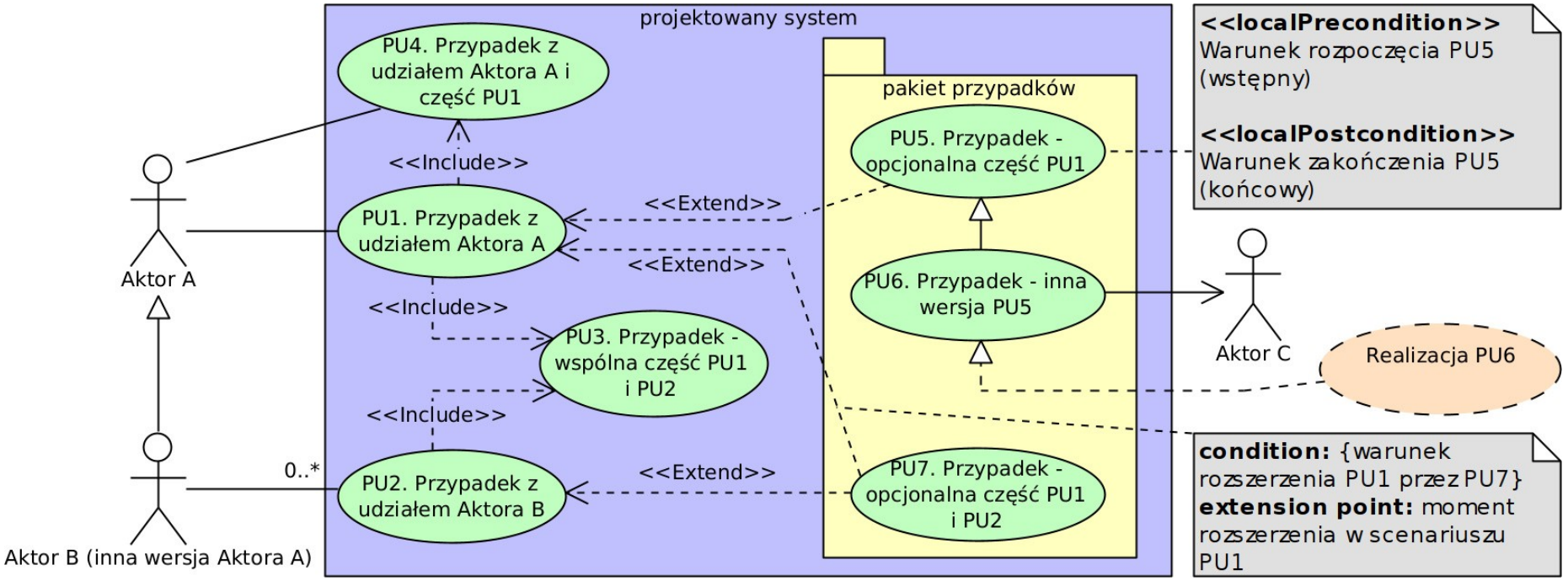
- Typowy skład słownego opisu przypadku użycia (PU):
  - **numer PU**;
  - **nazwa PU** – krótka, w formie wykonywanej operacji, a NIE obiektu;
  - **cel PU** – dłuższe wyjaśnienie (jeśli nazwa nie jest dość precyzyjna);
  - **warunki wstępne PU** – wymagania i przypadki użycia pozwalające rozpocząć wykonanie PU;
  - **warunki końcowe PU** – wymagania i przypadki użycia pozwalające zakończyć wykonanie PU (po czym poznać, że wykonał się prawidłowo);
  - **przypadki testowania** – testy sprawdzające wykonanie PU;
  - **scenariusz PU** – przepływ zdarzeń, które składają się na wykonanie PU (algorytm wykonania PU).
- Scenariusz jest najważniejszy i obowiązkowy!

Scenariusz realizacji przypadku użycia

- Przepływ zdarzeń, które składają się na wykonanie przypadku użycia w modelowanym SI.
- Uwzględnia:
  - aktorów,
  - artefakty (przepływ obiektów),
  - zdarzenia – czynności wykonywane w przez aktorów w modelowanym SI lub przez modelowany SI (przepływ sterowania).
- Może być tekstowy (pseudokod) lub graficzny (diagram czynności).
- Oprócz głównego scenariusza mogą być alternatywne scenariusze.
- Pokazuje, w którym momencie następuje rozszerzenie PU przez inny przypadek użycia, będący z nim w relacji *«include».*
- Pokazuje, w którym momencie i pod jakim warunkiem lub alternatywnie z jakim innym przypadkiem użycia następuje rozszerzenie PU przez inny przypadek użycia, będący z nim w relacji *«extend».*

Przykład dokumentacji przypadku użycia

- **Opis sytuacji:**
  - PU1 zawiera w sobie PU3 i PU4 (relacje «include»).
  - PU1 jest rozszerzany przez PU5 (lub PU6) lub alternatywnie przez PU7 (relacje «extend»).
  - PU2 zawiera w sobie PU3 (relacja «include»).
  - PU2 może być rozszerzany przez PU7 (relacja «extend»).



Przykład dokumentacji przypadku użycia

- **Dokumentacja PU1:**
  - **numer:** PU1
  - **nazwa:** Przypadek z udziałem aktora A
  - **cel:** Aktor A bierze udział w wykonaniu PU1, aby otrzymać pewną informację.
  - **warunki wstępne:** W roli Aktora A nie występuje Aktor B.
  - **warunki końcowe:** Wykonano PU5/PU6 i Aktor A dostał informację lub PU7.
  - **scenariusz**:

    (1) Aktor A inicjuje wykonanie PU1.

    (2) System przygotowuje wstępną treść informacji.

    (3) Wykonanie PU3 w celu uzupełnienia tej informacji.

    (4) Wykonanie PU4 w celu sprawdzenia tej informacji.

    (5) System przedstawia Aktorowi A tę informację do zatwierdzenia.

    (6) Wykonanie PU5, jeśli Aktor A chce zatwierdzić tę informację, lub wykonanie PU7, jeśli Aktor A chce ją odrzucić. Zamiast PU5 można wykonać PU6, jeśli Aktor C ma wiedzieć o zatwierdzeniu tej informacji.

Przykład dokumentacji przypadku użycia

- **Dokumentacja PU2:**
  - **numer:** PU2
  - **nazwa:** Przypadek z udziałem aktora B
  - **cel:** Aktor B bierze udział w wykonaniu PU2, aby odrzucić błędną informację.
  - **warunki wstępne:** Aktor A wykonał PU1 razem z PU5 lub wykonał PU6 i utworzył informację.
  - **warunki końcowe:** Dla błędnej informacji wykonano PU7 i odrzucono ją.
  - **scenariusz PU**:

    (1) Aktor B inicjuje wykonanie PU2.

    (2) System przedstawia treść informacji.

    (3) Wykonanie PU4 w celu sprawdzenia tej informacji.

    (4) System przedstawia Aktorowi B tę informację do zatwierdzenia.

    (5) Wykonanie PU7, jeśli ta informacja jest błędna i Aktor B chce ją odrzucić.

- Co robi system i co robią aktorzy w ramach wykonania PU3–PU7?
  To pokazują ICH dokumentacje.

4

Diagram czynności (aktywności)

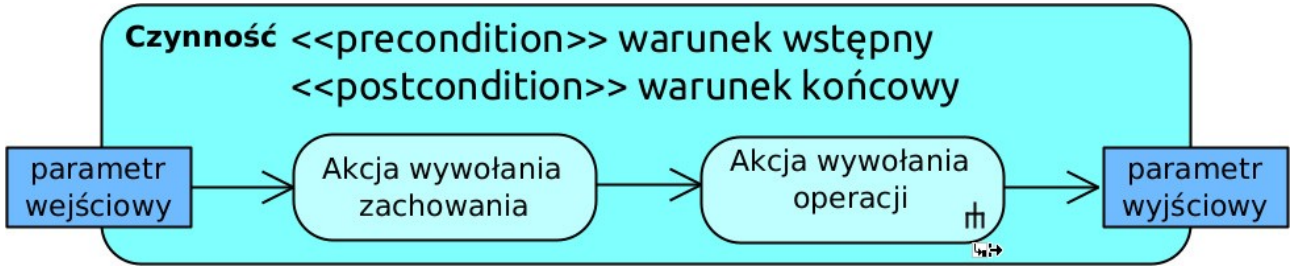
Diagram czynności /activity diagram/
(niepoprawnie ale często zwany diagramem aktywności)

- Składa się z wykonywanych operacji: czynności i akcji.
- Modeluje algorytm i dynamikę systemu lub jego części:
  - **przepływy sterowania** – bezpośrednie przejścia między operacjami;
  - **przepływy obiektów** – przejścia między operacjami z przekazaniem obiektów między nimi.
- Na podstawie słownej specyfikacji wymagań oraz diagramów, np. przypadków użycia, można wykonać diagram czynności modelujący:
  - wykonanie przypadku użycia (scenariusz),
  - wykonanie przypadku testowania,
  - przebieg procesu systemowego,
  - algorytm operacji klasy.

Diagram czynności (aktywności)

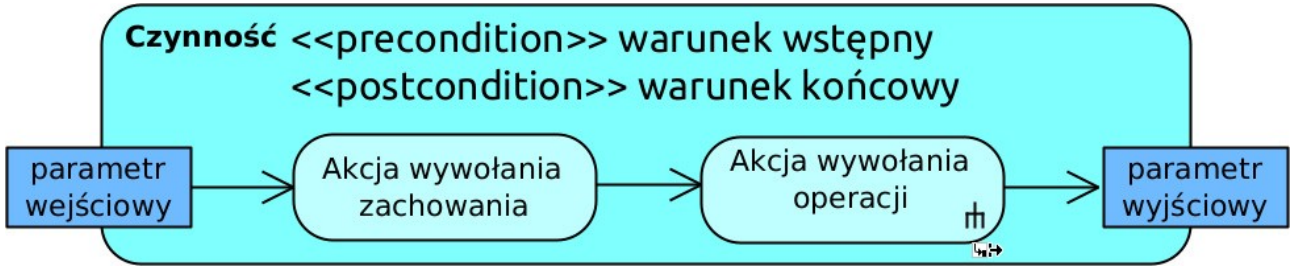
Czynność /activity/

- **Złożona procedura realizacji określonego zachowania**, np.:
  - w procesie biznesowym: złożone zachowanie systemu,
  - w algorytmie programu: operacja klasy.
- Może mieć **warunki wstępne** ***«precondition»*** – co musi być spełnione zanim czynność się zacznie.
- Może mieć **warunki końcowe** ***«postcondition»*** – co staje się spełnione po zakończeniu czynności.
- Może mieć **parametry wejściowe** /input parameter node/ – obiekty danych do przetwarzania przez czynność (zwykle umieszczane na lewej krawędzi czynności).
- Może mieć **parametry wyjściowe** /output parameter node/ – obiekty danych wytworzonych przez czynność (zwykle umieszczane na prawej krawędzi czynności).



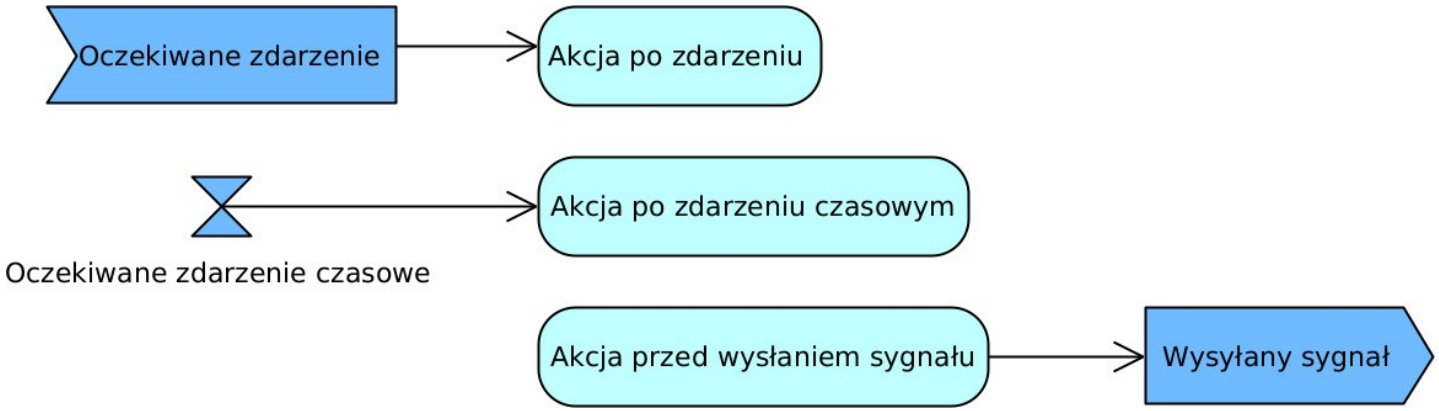
Akcja /action/

- Prosta i niepodzielna **część czynności**.
  - Czynność może zawierać akcje.
  - Akcja nie może zawierać czynności i akcji (jest atomowa – nie podlega dekompozycji).
- **Akcja wywołania zachowania** – przypisane jej zachowanie wykonuje sama,
- **Akcja wywołania operacji** (oznaczona „widłami”) – przypisane jej zachowanie przekazuje do wykonania czynność o tej samej nazwie.
  - Tę czynność przedstawia osobny diagram.



Akcja

- **Akcja oczekiwania na zdarzenie** /accept event action/
(prostokąt z trójkątnym wcięciem)
  - oczekuje na zajście zdarzenia, którego opis lub nazwę zawiera.
- **Akcja oczekiwania na czasowe zdarzenie** /accept time event action/
(klepsydra)
  - oczekuje na zajście zdarzenia czasowego, którego opis lub nazwę zawiera: kiedy lub jak często ono zachodzi.
- **Akcja wysłania sygnału** /send signal action/
(prostokąt z trójkątnym uwypukleniem)
  - tworzy i wysyła sygnał powodujący zdarzenie (np. wykonanie czynności lub stanu).



Przykład sekwencji czynności i akcji

- Przepływ sterowania fragmentu procesu wypłaty pieniędzy z bankomatu:
  - czynność **Wprowadź kwotę do wypłaty**,
  - akcja wysłania sygnału **Żądanie wypłaty**,
  - akcja oczekiwania na zdarzenie **Zgoda banku**,
  - czynność **Zabierz pieniądze**.

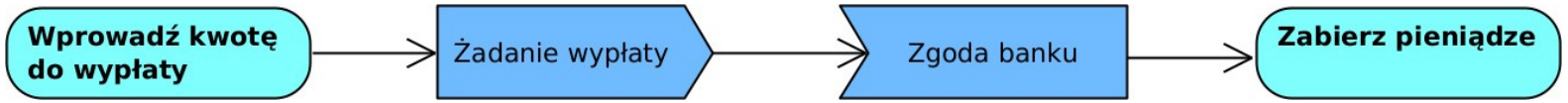


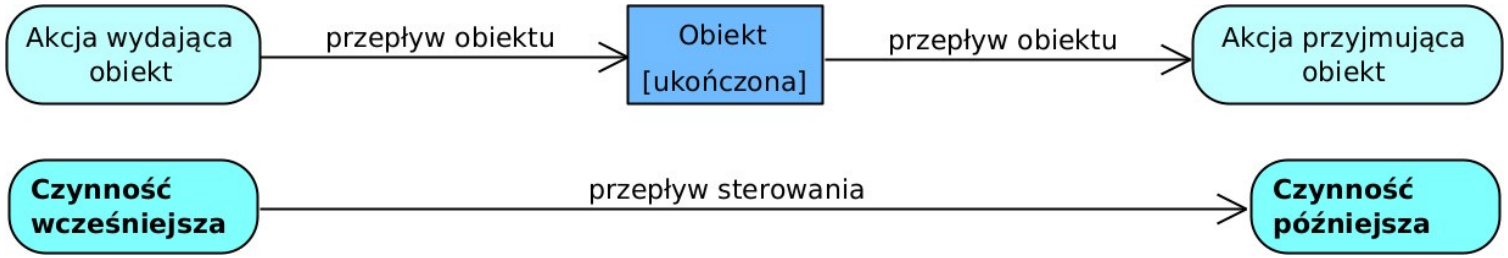
Diagram czynności (aktywności)

Obiekt /object/

- **Artefakt** (np. przetwarzana informacja, instancja klasy), który może być przedmiotem czynności i akcji jako ich dane wejściowe lub dane wyjściowe.
- Jest przekazywany między akcjami i czynnościami (przepływ obiektów).
- Bieżący **stan obiektu** podaje się w nawiasach [ ].
- Może być kontenerem obiektów.

Przepływ

- **Przepływ sterowania** /control flow/ – bezpośrednie przejście między dwiema czynnościami,dwiema akcjami lub czynnością a akcją.
- **Przepływ obiektu** /object flow/ – przejście między operacjami z przekazaniem obiektu między nimi.



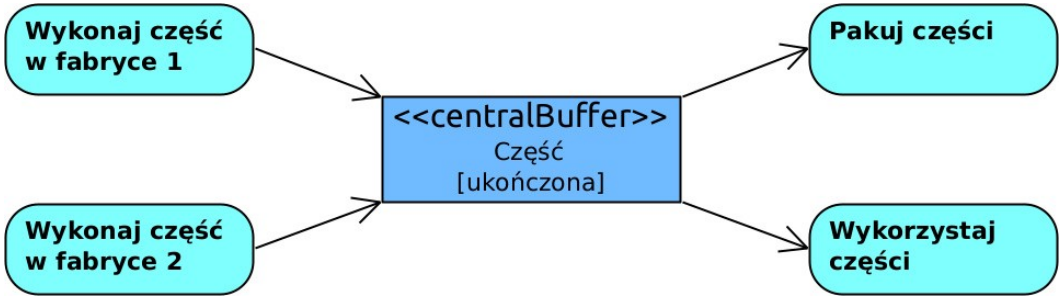
Kontener

- **Repozytorium**, które przyjmuje, przechowuje i wydaje obiekty (np. kolekcja, baza danych).
  - Umieszczenie obiektu w kontenerze – przez wchodzący do niego przepływ obiektów.
  - Pobranie obiektu z kontenera – przez wychodzący z niego przepływ obiektów, opcjonalnie z parametrem *weight* i liczbą obiektów do pobrania, np.:
    - *{weight = all}* – pobranie wszystkich obiektów,
    - *{weight = *}* – pobranie dowolnej liczby obiektów.
- **Bufor centralny** (obiekt ze stereotypem ***«centralBuffer»***).
  - Wydawany obiekt jest usuwany jest z kontenera.
- **Magazyn danych** (obiekt ze stereotypem ***«datastore»***).
  - Wydawany obiekt NIE jest usuwany z kontenera.
  - Wydawana jest jego kopia lub referencja do niego.

Przykład bufora centralnego

na podst. Unified Modeling Language (UML)

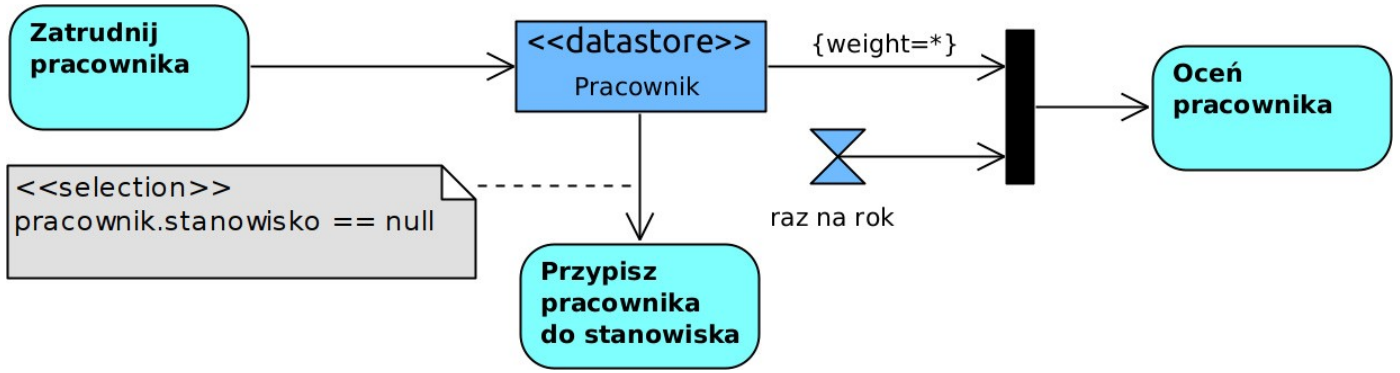
- Dwie fabryki wytwarzają obiekty **Część**.
- Czynności **Wykonaj część w fabryce 1** i **Wykonaj część w fabryce 2** przekazują swój obiekt **Część** w stanie **ukończona** do bufora **Część**.
  - Bufor gromadzi te obiekty.
- Czynności **Pakuj części** i **Wykorzystaj części** pobierają z tego bufora obiekt **Część**:
  - każdy obiekt **Część** trafia tylko do jednej z tych czynności (losowo),
  - tylko niepobrany obiekt **Część** pozostaje w buforze.



Przykład magazynu danych

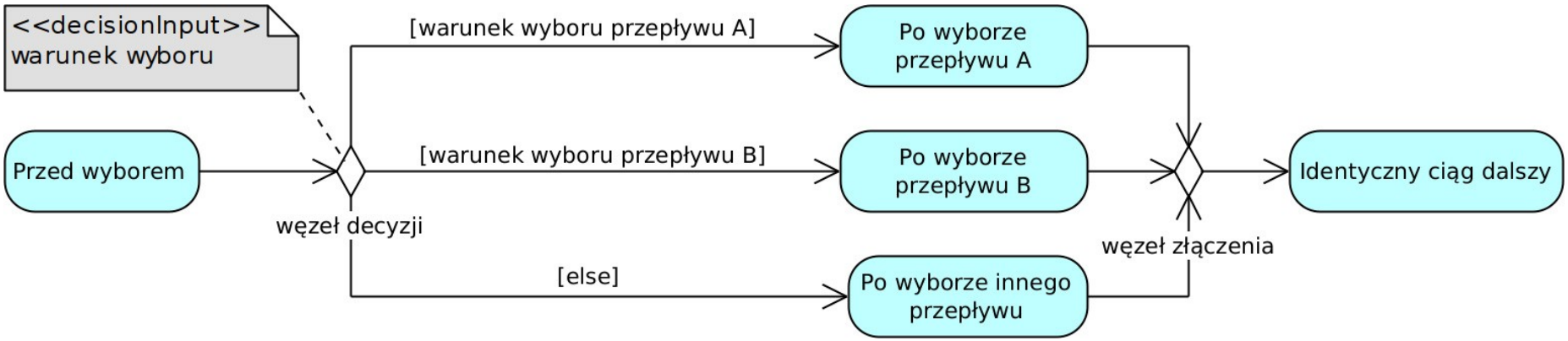
na podst. Unified Modeling Language (UML)

- Czynność **Zatrudnij pracownika** umieszcza obiekty **Pracownik** w magazynie **Pracownik**.
  - Magazyn gromadzi te obiekty.
- Czynność **Przypisz pracownika do stanowiska** pobiera z magazynu obiekt **Pracownik**, jeśli ten obiekt nie ma przypisanego stanowiska («selection»).
  - pobrany obiekt **Pracownik** pozostaje w buforze.
- Czynność **Oceń pracownika** pobiera z magazynu dowolną liczbę (*{weight=*}*) obiektów **Pracownik** raz na rok (zdarzenie czasowe).
  - pobrany obiekt **Pracownik** pozostaje w buforze.



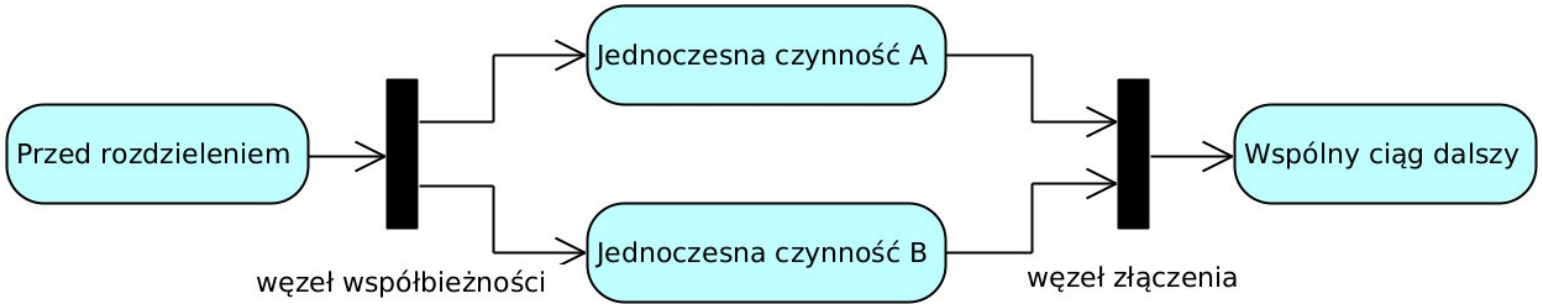
Węzeł decyzji

- Rozpoczyna i/lub kończy przepływy alternatywne względem siebie.
- **Węzeł decyzji** /decision/ – wychodzą z niego alternatywne przepływy.
  - Każda krawędź wychodząca ma warunek wyboru w nawiasach [ ].
  - Węzeł może mieć (jeśli to konieczne) opis decyzji: tekst przy węźle decyzji lub w notatce ze stereotypem *«decisionInput»*.
  - Warunek i opis decyzji nie tworzą informacji, ale ją sprawdzają.
- **Węzeł złączenia** /merge/ – wchodzą do niego alternatywne przepływy:
  - wychodzący z niego przepływ jest ich identycznym ciągiem dalszym.
- Węzeł złączenia jest też węzłem decyzji, gdy ma kilka wejść i kilka wyjść.



Węzeł współbieżności

- Rozpoczyna i/lub kończy przepływy współbieżne ze sobą.
- **Węzeł współbieżności** /fork/ – wychodzą z niego współbieżne przepływy.
- **Węzeł złączenia** /join/ – wchodzą do niego współbieżne przepływy.
  - Wychodzący z niego przepływ:
    - jest ich wspólnym ciągiem dalszym,
    - zaczyna się dopiero po skończeniu wszystkich wchodzących przepływów współbieżnych – węzeł synchronizacji.
- Węzeł złączenia jest też węzłem współbieżności, gdy ma kilka wejść i kilka wyjść.



Węzeł początkowy /initial/

- Element diagramu lub czynności zawierającej akcje:
  - rozpoczyna jego/jej przepływ sterowania ,
  - wskazuje jego początkową czynność lub akcję.
- Diagram lub czynność może mieć tylko 1 węzeł początkowy.

Diagram czynności (aktywności)

Węzeł końcowy /activity final/

- Element diagramu lub czynności zawierającej akcje:
  - kończy jego/jej przepływ sterowania,
  - przerywa jego/jej pozostałe nieukończone współbieżne przepływy.
- Diagram lub czynność może mieć wiele węzłów końcowych.

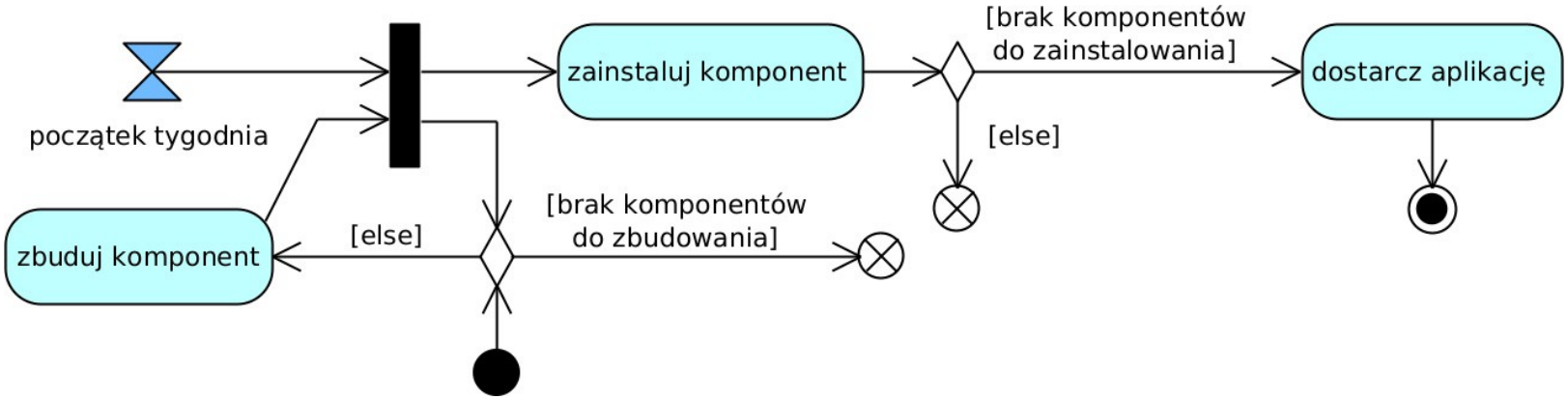
Węzeł końca przepływu /flow final/

- Element diagramu lub czynności zawierającej akcje:
  - kończy jego/jej przepływ sterowania,
  - NIE przerywa jego/jej pozostałych nieukończonych współbieżnych przepływów.
- Diagram lub czynność może mieć wiele węzłów końca przepływu.

Przykład węzłów końcowych i pętli

na podst. Unified Modeling Language (UML)

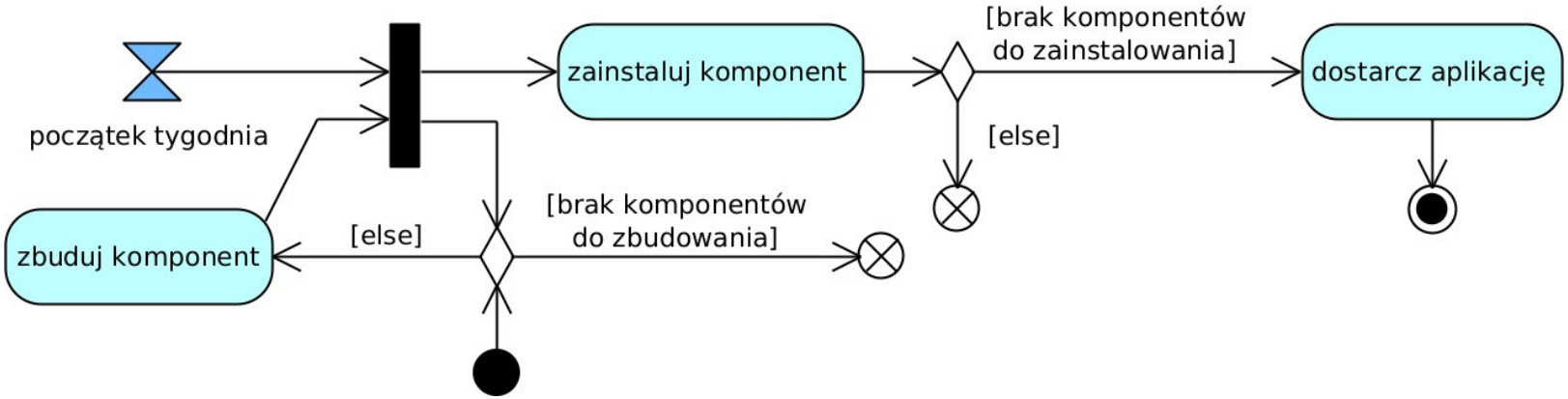
- Tworzenie aplikacji składa się z dwóch przepływów sterowania:
  - przepływ 1 z akcją **zbuduj komponent**,
  - przepływ 2 z akcjami **zainstaluj komponent** i **dostarcz aplikację**.
- Zapętlony przepływ 1 zawiera akcję **zbuduj komponent**.
- Przepływ 1 kończy się, gdy **brak komponentów do zbudowania**.
  - nie przerywa to przepływu 2 (rozpoczętego w poprzedniej iteracji pętli).



Przykład węzłów końcowych i pętli

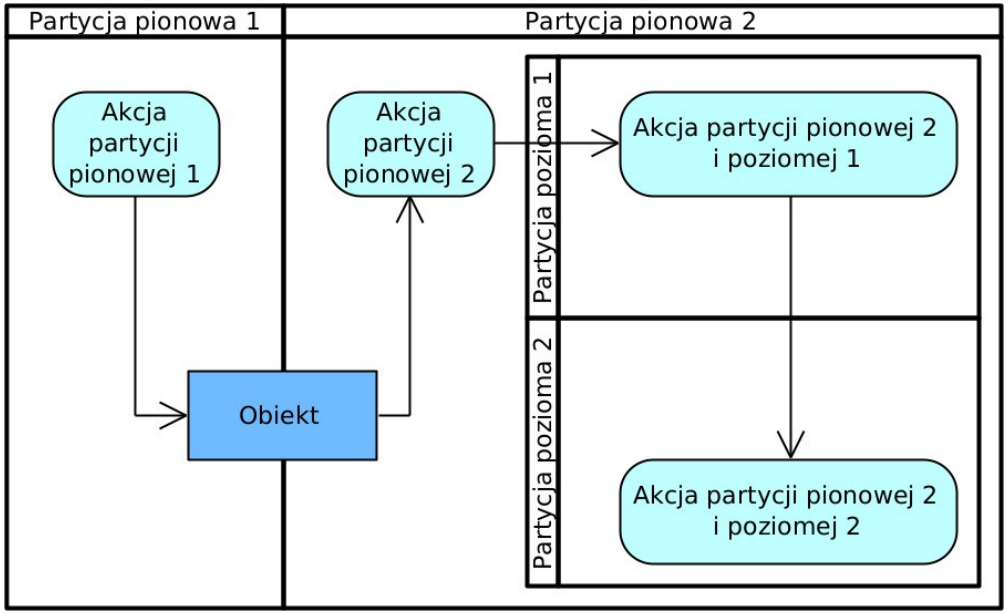
na podst. Unified Modeling Language (UML)

- Po zajściu **początku tygodnia** i po zbudowaniu komponentu rozpoczyna się przepływ 2 zawierający akcje **zainstaluj komponent** i (opcjonalnie) **dostarcz aplikację**.
- Przepływ 2 kończy się akcją **dostarcz aplikację**, gdy **brak komponentów do zainstalowania**:
  - przerywa to przepływ 1 (bieżącą iterację pętli).
- W innym wypadku przepływ 2 kończy się bez tej akcji:
  - nie przerywa to przepływu 1 (bieżącej iteracji pętli).



Partycja /partition/
(dawniej zwana torem /swimlane/)

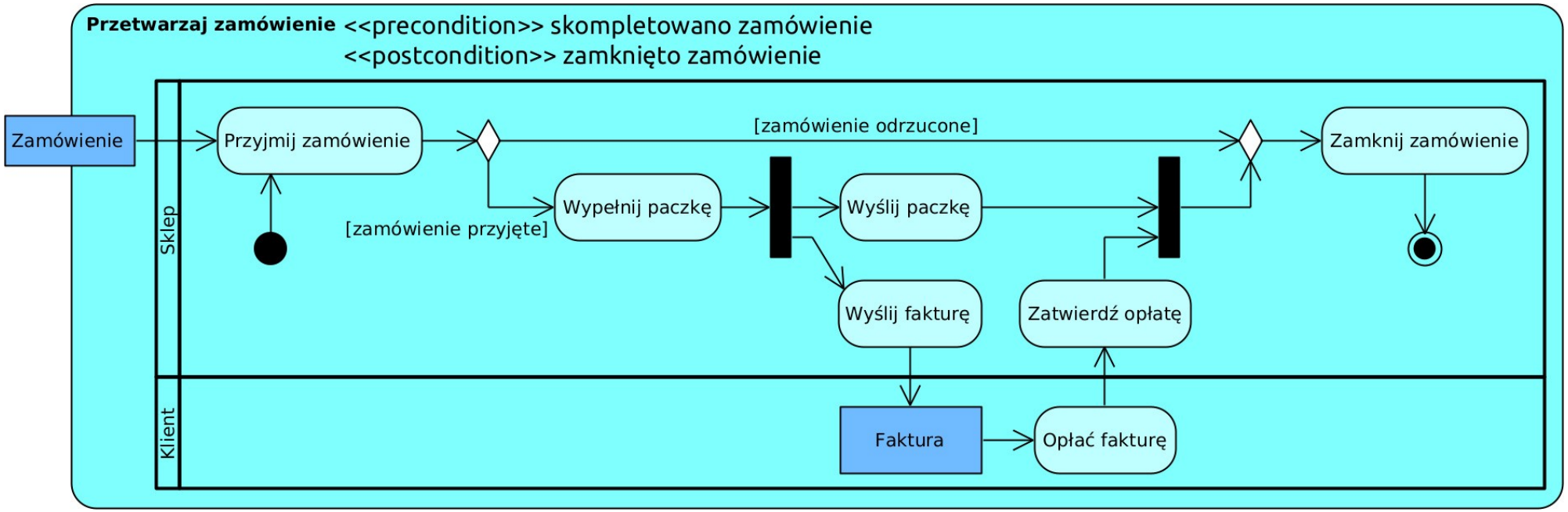
- Segregacja czynności i akcji (NIE dotyczy obiektów):
  - na różnym poziomie abstrakcji
    (np. na podstawie udziału w nich aktorów, miejsca ich wykonania, ich implementacji).
- Każda czynność i akcja musi być w konkretnej partycji.
- Obiekt może być w konkretnej partycji lub na granicy między nimi.
- Podział na partycje może być pionowy lub poziomy.
- Partycje mogą być zagnieżdżone.



Przykład partycji i węzłów decyzji i współbieżności

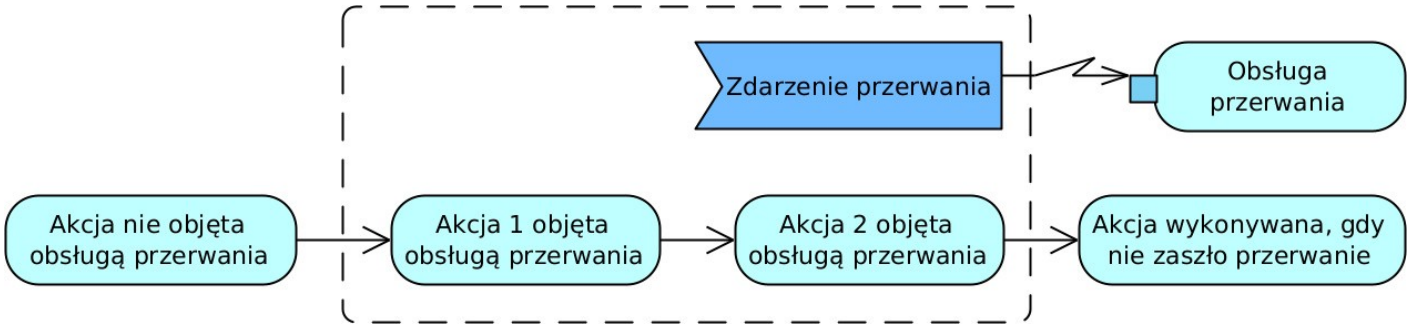
na podst. Unified Modeling Language (UML)

- Czynność **Przetwarzaj zamówienie** zaczyna się akcją **Przyjmij zamówienie** z parametrem wejściowym **Zamówienie**.
- Akcja **Wyślij paczkę** jest współbieżna z przepływem z akcjami: **Wyślij fakturę**, **Zatwierdź fakturę** i **Zatwierdź opłatę**.
- Akcja **Wyślij fakturę** przekazuje obiekt **Faktura** do akcji **Opłać fakturę**.
- Akcja **Opłać fakturę** wykonywana jest przez **Klienta**, inne – przez **Sklep**.



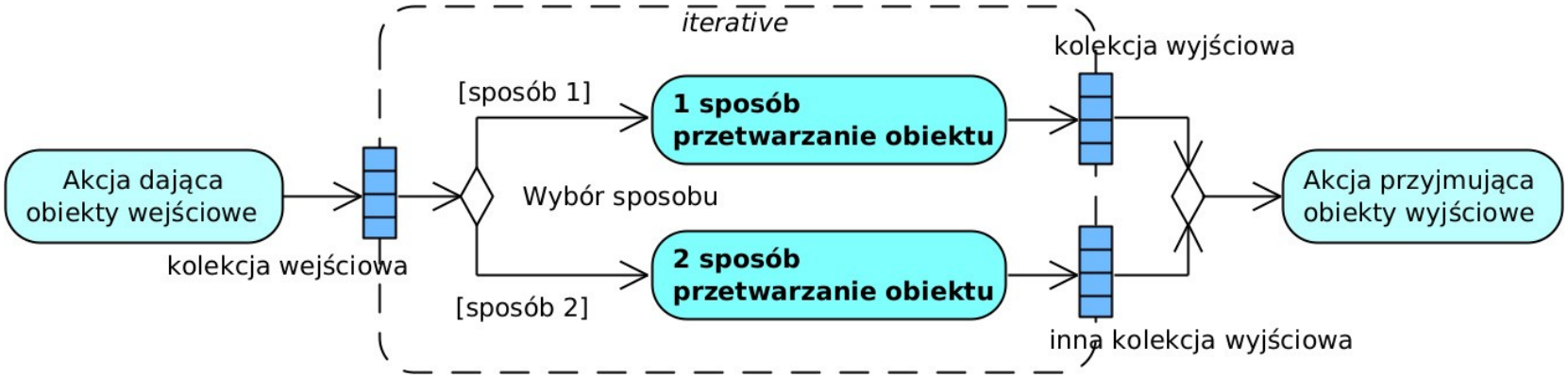
Obszar przerywalny /interruptible region/

- **Sekcja krytyczna i obsługa zdarzenia przerwania lub wyjątku.**
- Zawiera akcję oczekiwania na to zdarzenie.
  - Wychodzi z niej przepływ obsługi przerwania /exception handler/ (ma kształt pioruna lub obok symbol pioruna).
  - Przepływ ten wchodzi do czynności lub akcji obsługi przerwania.
- Zawiera czynności i akcje, w czasie których może zajść to zdarzenie.
  - Innych czynności i akcji NIE zawiera.



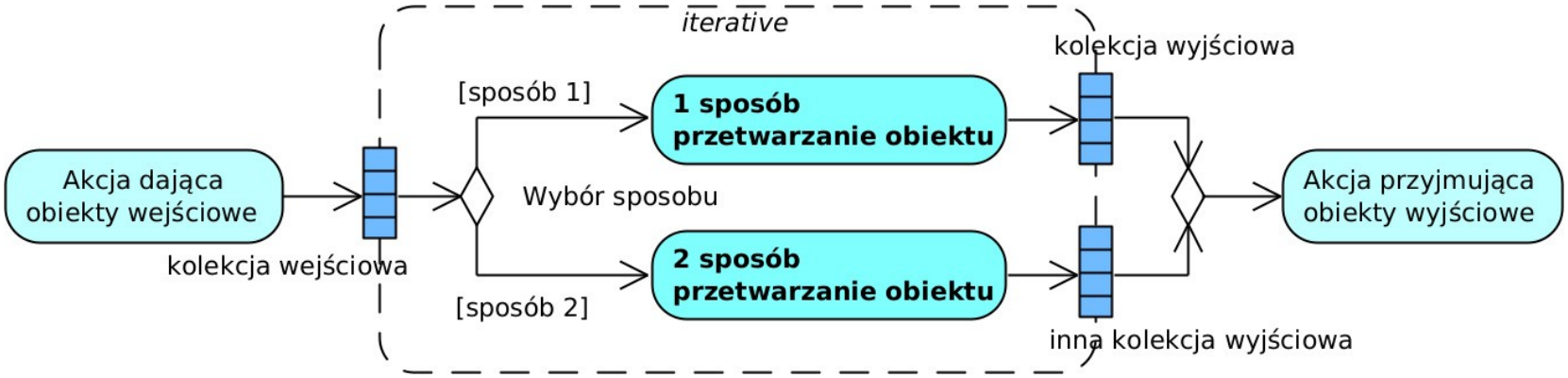
Obszar rozszerzenia /expansion region/

- Modeluje **przetwarzanie każdego obiektu z kolekcji obiektów**.
- Zawiera przepływy z czynnościami i akcjami przetwarzania 1 obiektu z wykorzystaniem odpowiednich węzłów.
- Przepływ wchodzący do tego obszaru wchodzi przez wierzchołek wejściowy /input expansion node/ z nazwą wejściowej kolekcji obiektów.
- Przepływ wychodzący z tego obszaru wychodzi przez wierzchołek wyjściowy /output expansion node/ z nazwą wyjściowej kolekcji obiektów.
- Może być kilka wierzchołków wejściowych i/lub wyjściowych.

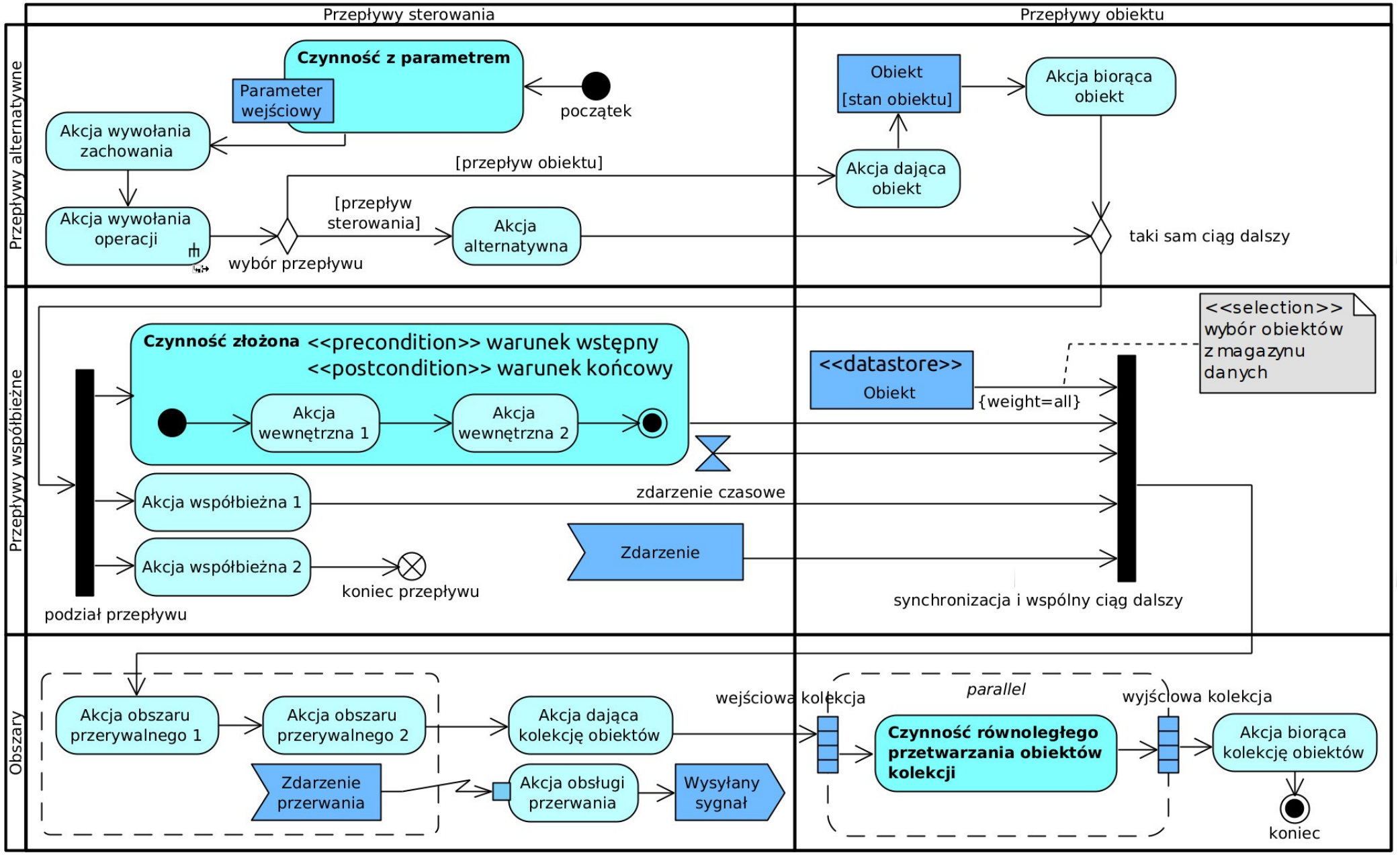


Obszar rozszerzenia

- Sposób przetwarzania obiektów podaje się u góry obszaru:
  - **szeregowo (*«iterative»*)** – otrzymanie całej kolekcji obiektów, a następnie kolejne ich przetworzenie, jeden po drugim itd.;
  - **strumieniowo (*«stream»*)** – kolejne przetworzenie obiektów, jeden po drugim itd., w miarę ich otrzymywania;
  - **współbieżnie (*«paralel»*)** – otrzymanie całej kolekcji obiektów, a następnie współbieżne ich przetworzenie, jednego z drugim itd.



Podsumowanie



5

Diagram stanów

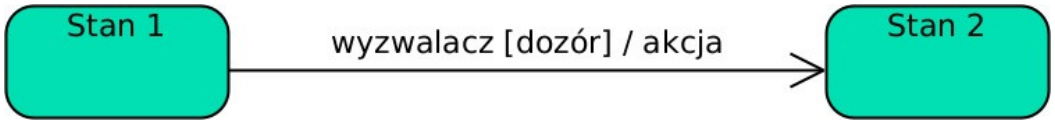
Diagram stanów /state diagram, state mashine diagram/

- Modeluje stan i dynamikę obiektu: systemu, komponentu, instancji klasy itd.
- Ma postać maszyny skończenie-stanowej i zawiera:
  - **stany**, w których obiekt może być;
  - **przejścia** (zmiany stanów) np. na skutek zewnętrznego zdarzenia.
- NIE modeluje algorytmu np. wykonania operacji klasy.
  - Do tego służy diagram czynności i diagram sekwencji.

- **Stan** /state/ – ustalona cecha obiektu (systemu, jego elementu, instancji klasy itd.):
  - określa, co się w danym czasie z nim dzieje – jakie ma wartości atrybutów i jakie wykonuje operacje.
- **Przejście** /transition/ – zmiana stanu obiektu:
  - uzyskanie, zachowanie, lub utrata określonych własności obiektu;
  - wejście w inny stan lub ponowne wejście w ten sam stan.

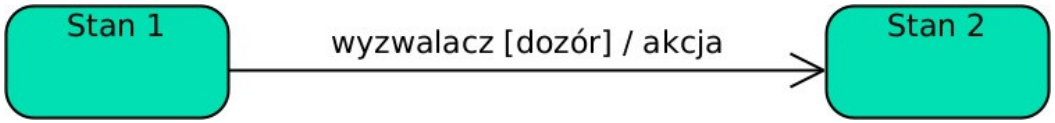
Opis przejścia

- Opis przejścia (i jego kolejność): ***wyzwalacz [dozór] /akcja***
  - Każda z części opisu jest opcjonalna.
- **Wyzwalacz** /trigger/ – zewnętrzne zdarzenie:
  - pochodzi spoza modelowanego obiektu,
  - powoduje przejście, jeśli jest dostępne.
- **Dozór** /guard/ – wyrażenie logiczne, warunek wykonania przejścia:
  - wynika z bieżącego stanu obiektu lub wyzwalacza przejścia,
  - udostępnia przejście tylko, gdy jest prawdziwy.
- **Akcja** /action/ – wewnętrzne zdarzenie:
  - pochodzi z modelowanego obiektu,
  - skutek przejścia zachodzący w trakcie jego wykonywania.



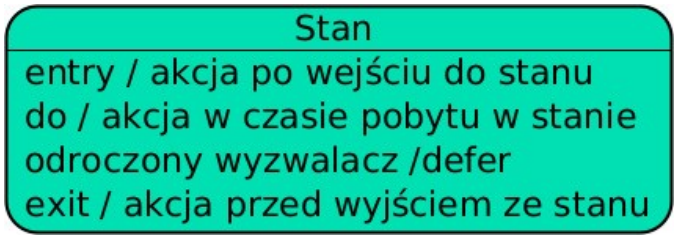
Opis przejścia

- **Które przejście wykonać, gdy ze stanu wychodzi kilka przejść?**
- Najwyższy priorytet ma przejście mające wyzwalacz, gdy zachodzi związane z nim zdarzenie.
  - Wybór losowy, jeśli jest więcej takich przejść.
- Średni priorytet ma przejście mające prawdziwy dozór.
  - Wybór losowy, jeśli jest więcej takich przejść.
- Najniższy priorytet ma przejście bez opisu.
  - Wybór losowy, jeśli jest więcej takich przejść.
- **NIE może być wykonane przejście, które:**
  - ma NIEprawdziwy dozór,
  - ma wyzwalacz, ale NIE zachodzi związane z nim zdarzenie.



Operacje stanu

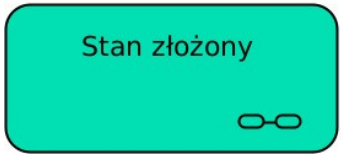
- Obiekt w danym stanie może wykonać operację na skutek zajścia zdarzenia.
- Opis operacji stanu: ***wyzwalacz /akcja***
- **Wyzwalacz** – zewnętrzne zdarzenie:
  - pochodzi spoza modelowanego obiektu,
  - powoduje akcję.

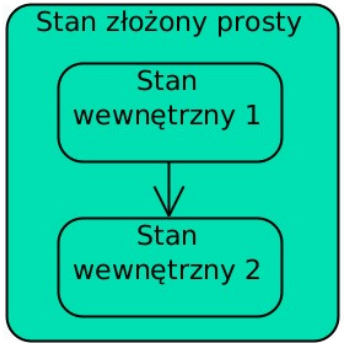


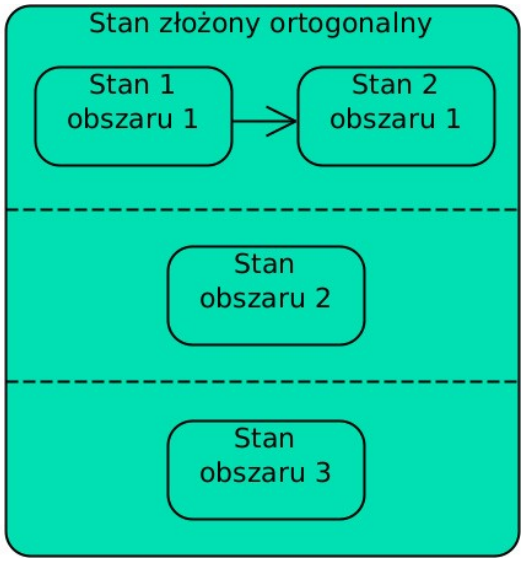
- **Akcja** – wewnętrzne zdarzenie:
  - pochodzi z modelowanego obiektu – jego operacja.
- Wyzwalacze:
  - ***entry*** – zdarzenie wejścia w stan (akcja jest zaraz po tym wejściu),
  - ***do*** – „zdarzenie” pobytu w stanie (akcja jest w czasie trwania stanu),
  - ***exit*** – zdarzenie wyjścia ze stanu (akcja jest zaraz przed tym wyjściem),
  - odroczony – zdarzenie niepowodujące zmiany stanu (***/defer*** zamiast akcji)
    - to zdarzenie modelowane jest czynnością,
    - ***all*** – dotyczy wszystkich zdarzeń.

Stan złożony /composite state/

- Stan może być **złożony** z 1 lub więcej stanów:
  - przez dekompozycję cech obiektu,
  - te stany też mogą być złożone itd.,
  - przejścia mogą łączyć stany z różnych poziomów zagnieżdżenia.
- **Stan złożony prosty** – składa się ze stanu lub ich szeregowej sekwencji.
- **Stan złożony ortogonalny** – składa się ze współbieżnych obszarów:
  - każdy obszar /region/ zawiera stan lub szeregową sekwencję stanów.
- Stan oznacza się „okularami”, jeśli jego stany wewnętrzne przedstawia inny diagram (mający jego nazwę).





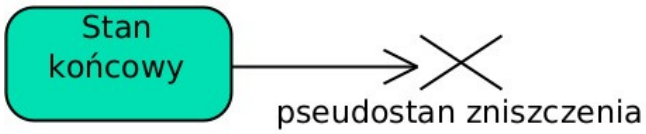


Pseudostan /pseudostate/

- Pseudostan NIE jest stanem – obiekt nie może się w nim znajdować.
- **Pseudostan początkowy** /initial/ – wskazuje domyślny początkowy stan diagramu, stanu złożonego prostego lub obszaru stanu złożonego ortogonalnego.
  - Diagram, stan złożony lub jego obszar może mieć tylko 1 pseudostan początkowy.
- **Pseudostan końcowy** /final/ – kończy sekwencję stanów.
  - NIE ma wpływu na inne stany współbieżne – NIE przerywa pozostałych nieukończone współbieżnych sekwencji stanów.
  - Diagram, stan złożony lub jego obszar może mieć wiele pseudostanów końcowych.

Pseudostan

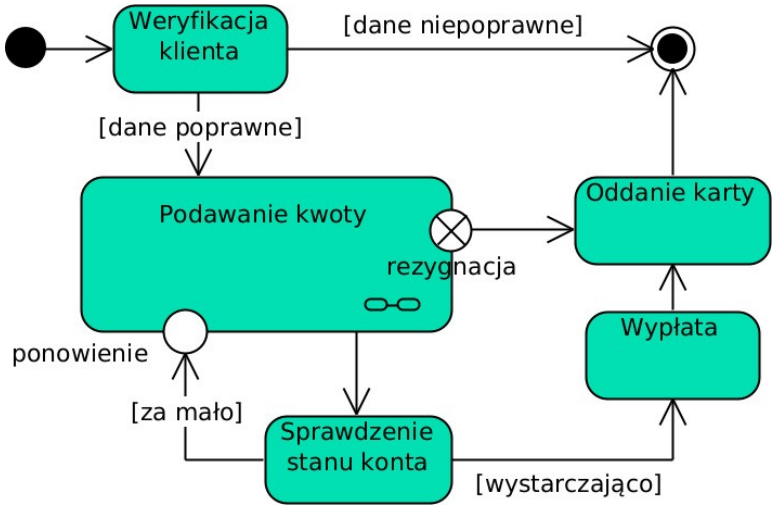
- **Pseudostan zniszczenia** /terminate/ – kończy sekwencję stanów i działanie całej maszyny stanowej:
  - przerywa pozostałe nieukończone współbieżne sekwencje stanów.
- **Punkt wejścia** /entry point/ (identyfikowany nazwą):
  - dla zawierającego go stanu: pokazuje konkretne (inne niż domyślne) rozpoczęcie sekwencji jego stanów wewnętrznych;
  - dla poprzedniego stanu: do niego wchodzi przejście z tego stanu.
- **Punkt wyjścia** /exit point/ (identyfikowany nazwą):
  - dla zawierającego go stanu: pokazuje konkretne (inne niż domyślne) zakończenie sekwencji jego stanów wewnętrznych;
  - dla następnego stanu: z niego wychodzi przejście do tego stanu.

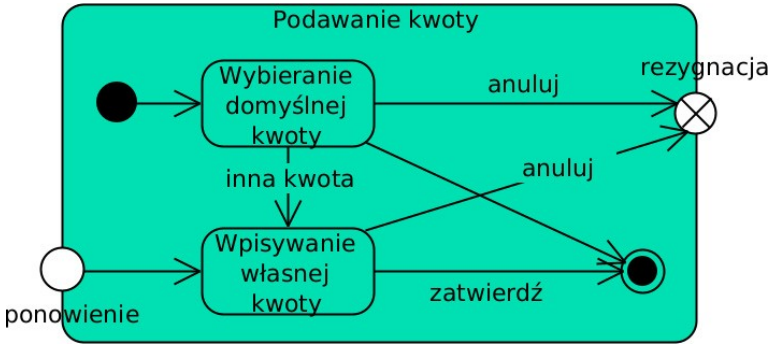


Przykład stanu złożonego z punktami wejścia i wyjścia

na podst. Unified Modeling Language (UML)

- Diagram 1 – sekwencja stanów pracy bankomatu wydającego pieniądze.
- Diagram 2 – wewnętrzne stany stanu **Podawanie kwoty**.
- W stanie **Podawanie kwoty** (diagram 2):
  - pseudostan początkowy wskazuje kontynuację przejścia ze stanu **Weryfikacja klienta** do stanu **Podawanie kwoty**,
  - punkt **ponowienie** wskazuje kontynuację przejścia ze stanu **Sprawdzenie stanu konta** do stanu **Podawanie kwoty**.

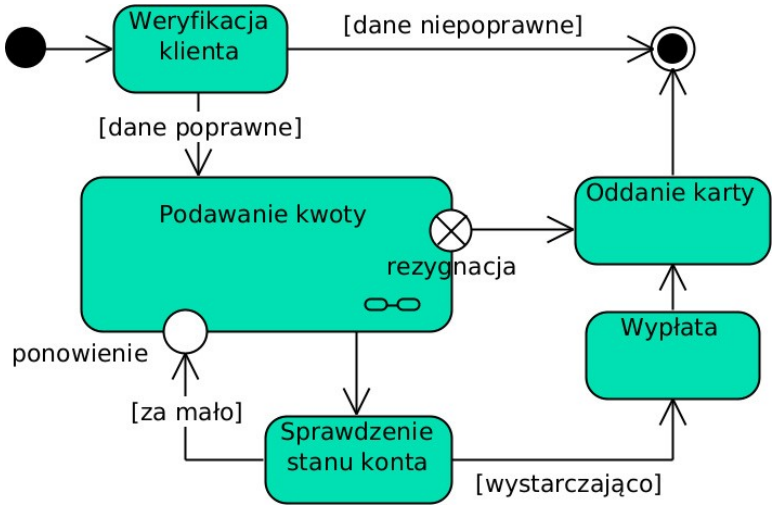


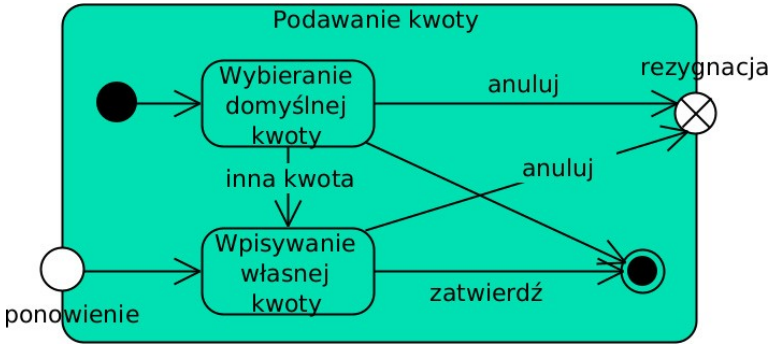


Przykład stanu złożonego z punktami wejścia i wyjścia

na podst. Unified Modeling Language (UML)

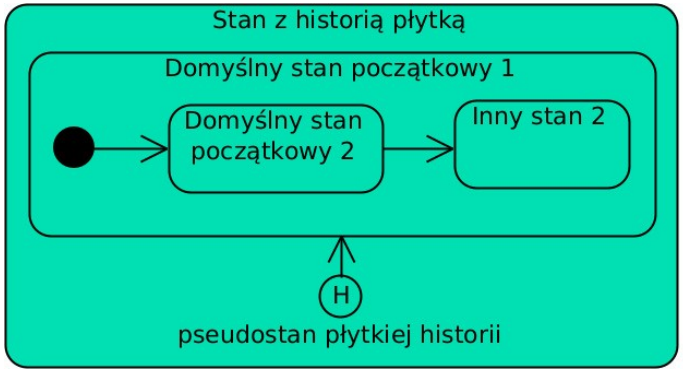
- W stanie **Podawanie kwoty** (diagram 2):
  - osiągnięcie pseudostanu końcowego powoduje przejście ze stanu **Podawanie kwoty** do stanu **Sprawdzenie stanu klienta**,
  - osiągnięcie punktu **rezygnacja** powoduje przejście ze stanu **Podawanie kwoty** do stanu **Oddanie karty**.





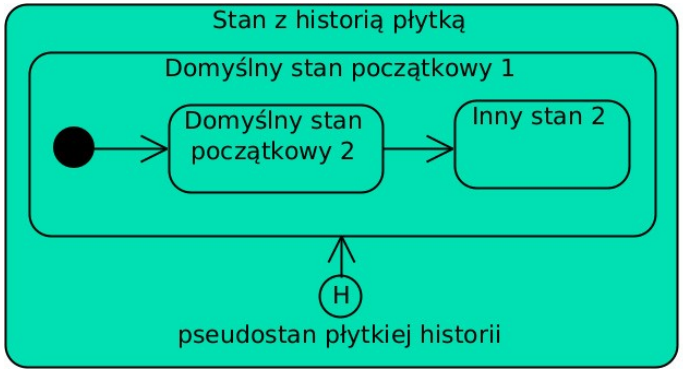
Pseudostan

- **Pseudostan płytkiej historii** /shallow history/ – wskazuje początkowy i pamięta ostatnio aktywny stan stanu złożonego:
  - przy pierwszym wejściu do stanu wskazuje początkowy stan wewnętrzny;
  - przy każdym kolejnym wejściu powoduje powrót do ostatnio aktywnego stanu wewnętrznego
    (ALE jego stany wewnętrzne rozpoczną się od domyślnego stanu).



Pseudostan

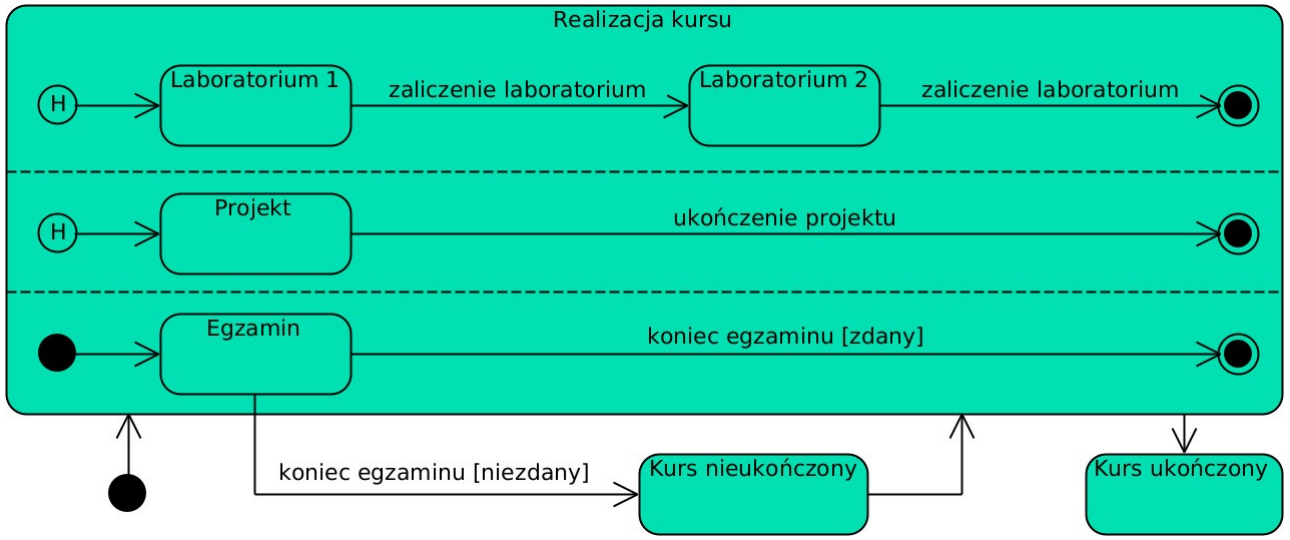
- **Pseudostan głębokiej historii** /deep history/ – wskazuje początkowy i rekurencyjnie pamięta ostatnio aktywny stan stanu złożonego:
  - przy pierwszym wejściu do stanu wskazuje początkowy stan wewnętrzny;
  - przy każdym kolejnym wejściu powoduje powrót do ostatnio aktywnego stanu wewnętrznego i jego stanów wewnętrznych NA KAŻDYM poziomie zagnieżdżenia.



Przykład stanu złożonego i pseudostanu historii

na podst. Unified Modeling Language (UML)

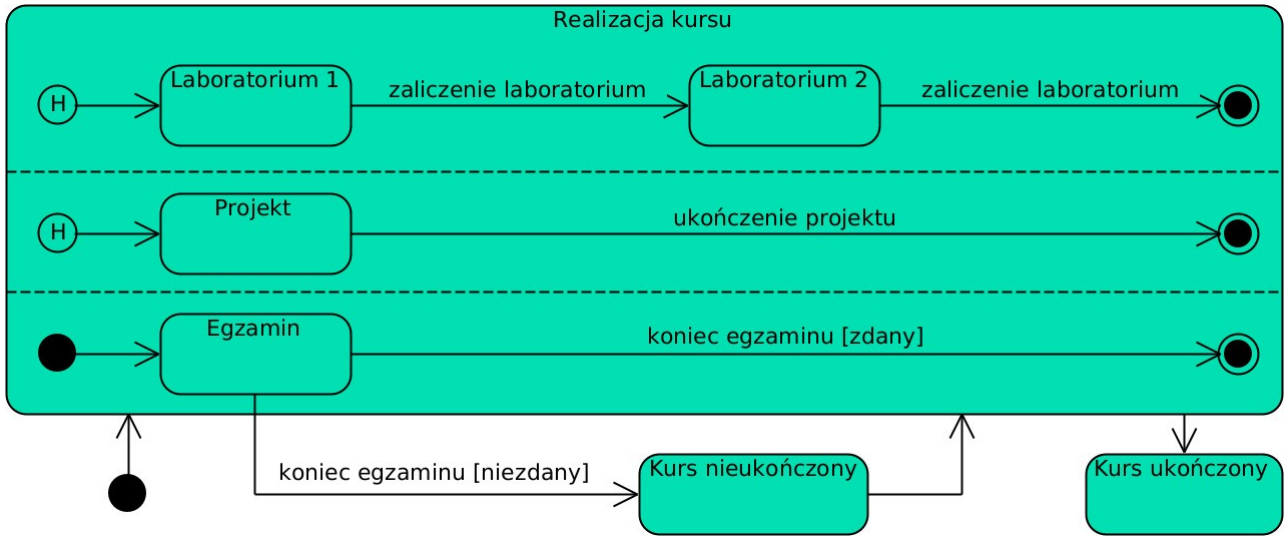
- Obiekt *Kurs* początkowo jest w stanie **Realizacja kursu**, a w nim:
  - niezależnie od siebie zaczynają się 3 sekwencje stanów:
    - stan **Laboratorium 1** → zdarzenie **zaliczenie laboratorium** → stan **Laboratorium 2** → zdarzenie **zaliczenie laboratorium**;
    - stan **Projekt** → zdarzenie **ukończenie projektu**.
    - stan **Egzamin** → zdarzenie **koniec egzaminu**.



Przykład stanu złożonego i pseudostanu historii

na podst. Unified Modeling Language (UML)

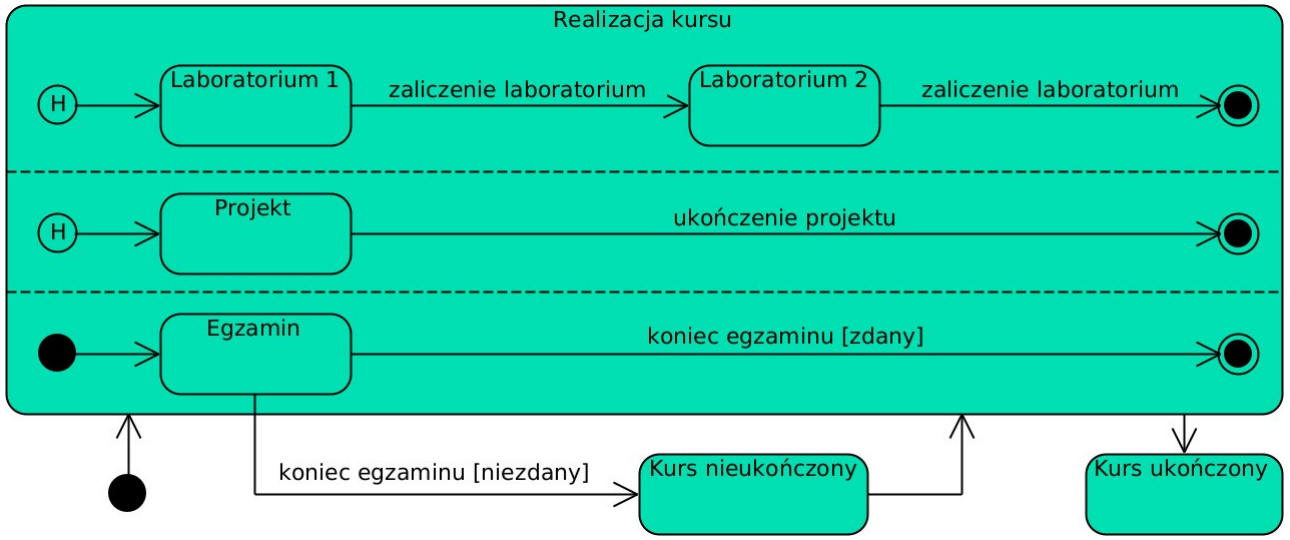
- Stan **Realizacja kursu** przechodzi w stan **Kurs ukończony**, gdy wszystkie obszary tego stanu się zakończą:
  - w stanie **Laboratorium 2** zaszło zdarzenie **zaliczenie laboratorium**,
  - w stanie **Projekt** zaszło zdarzenie **ukończenie projektu**,
  - w stanie **Egzamin** zaszło zdarzenie **koniec egzaminu** i spełniono warunek **zdany**.
- Kolejność kończenia obszarów nie ma znaczenia.



Przykład stanu złożonego i pseudostanu historii

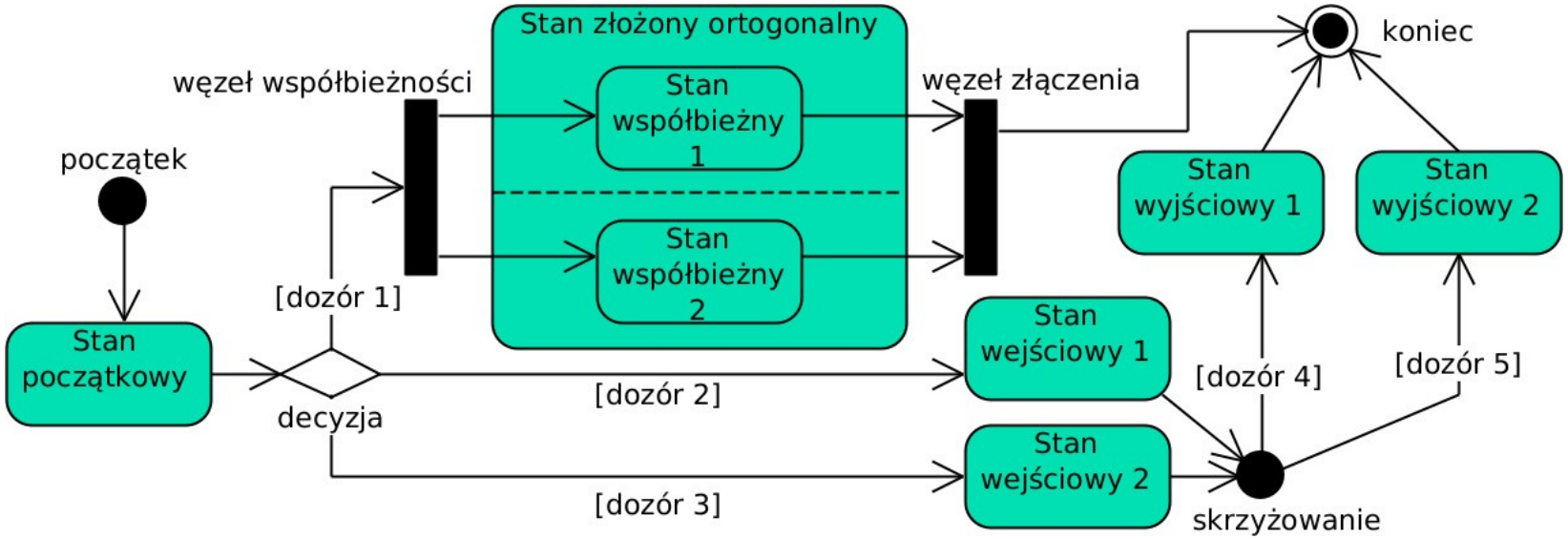
na podst. Unified Modeling Language (UML)

- Stan **Realizacja kursu** przechodzi w stan **Kurs nieukończony**, gdy w stanie **Egzamin** zachodzi zdarzenie **koniec egzaminu** i spełniono warunek **niezdany**.
  - To przejście powoduje też opuszczenie wewnętrznych stanów stanu **Realizacja kursu**, jeśli jeszcze są aktywne.
- Następnie stan **Kurs nieukończony** przechodzi w stan **Realizacja kursu**:
  - pierwsze 2 obszary: kontynuacja od ostatnio aktywnego stanu (jeśli był);
  - 3. obszar: rozpoczęcie w stanie **Egzamin**.



Węzeł /node/

- **Węzeł wyboru** /choice/ – rozdziela sekwencję stanów na alternatywne sekwencje stanów lub złącza alternatywne sekwencje stanów w identyczny ciąg dalszy.
  - warunek wyboru przejścia podaje się w nawiasach [ ]
- **Węzeł skrzyżowania** /junction/ – węzeł wyboru:
  - wejście do niego nie gwarantuje wyjścia z niego.
- **Węzeł współbieżności** – rozdziela /fork/ sekwencję stanów na współbieżne sekwencje stanów lub złącza i synchronizuje /join/ współbieżne sekwencje stanów we wspólny ciąg dalszy.



6

Diagram komunikacji

Diagram Komunikacji /communiaction diagram/
(dawniej zwany diagramem współpracy /collaboration/)

- Modeluje **interakcję** – przepływ sterowania między jej uczestnikami, np.:
  - złożony przypadek użycia lub operację jego realizacji,
  - algorytm wykonania operacji klasy.
- **Uczestnik interakcji** – aktor /actor/ lub **linia życia** /lifeline/: klasa, instancja klasy (obiekt), komponent, system itd.
- Nazwa linii życia:
  - dowolna, ogólna – dla diagramu konceptualnego;
  - ***instancja:klasa*** – uczestnikiem jest konkretna instancja klasy;
  - ***:klasa*** – uczestnikiem jest anonimowa instancja klasy lub sama klasa;
  - ***self*** – uczestnikiem jest posiadacz tego diagramu.
- Interakcję można umieścić w torach określających role jej uczestników (odpowiednik partycji z diagramu czynności).

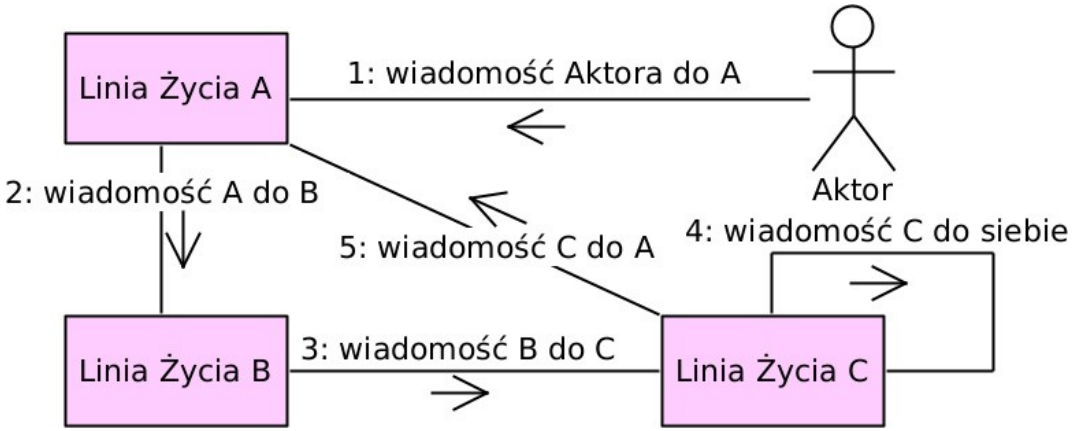
Diagram komunikacji

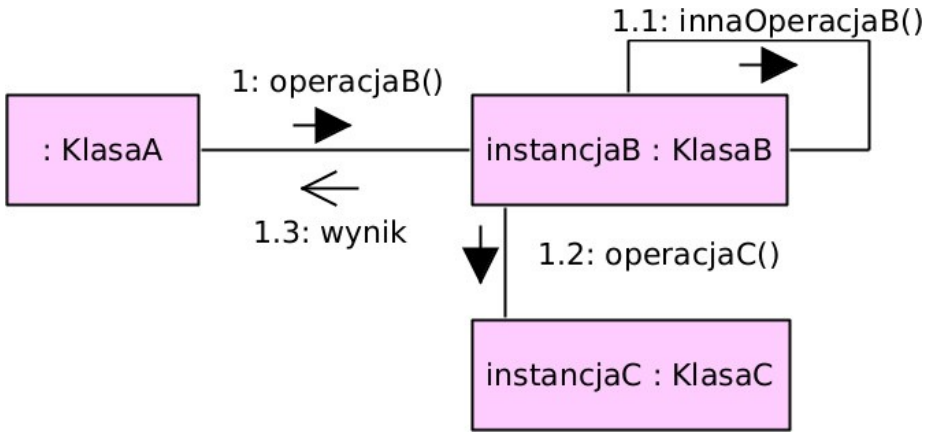
Wiadomość /message/

- **Relacja wiadomości**:
  - przekazanie sterowania (i opcjonalnie obiektu) między uczestnikami interakcji: od nadawcy do odbiorcy;
  - wykonanie operacji przez odbiorcę w odpowiedzi na żądanie nadawcy.
- **Odbiorcą** może być nadawca lub inny uczestnik interakcji.
- **Numer wiadomości** – kolejność jej wykonania;
  - pierwsza jest wiadomość nr 1,
  - numery mogą mieć podnumery itd. (np. 1.1).
- **Opis wiadomości** – zwykle operacja do wykonania przez odbiorcę.
- **Symbol strzałki** – wskazuje odbiorcę i rodzaj wiadomości.

Wiadomość

- **Wiadomość asynchroniczna** /asynchronous/ – asynchroniczne wywołanie operacji odbiorcy:
  - NIE towarzyszy jej wiadomość zwrotna,
  - grot relacji jest otwarty.
- **Wiadomość synchroniczna** /synchronous/ – synchroniczne wywołanie operacji odbiorcy:
  - może jej towarzyszyć asynchroniczna wiadomość zwrotna (otwarty grot),
  - grot relacji jest zamknięty.





7

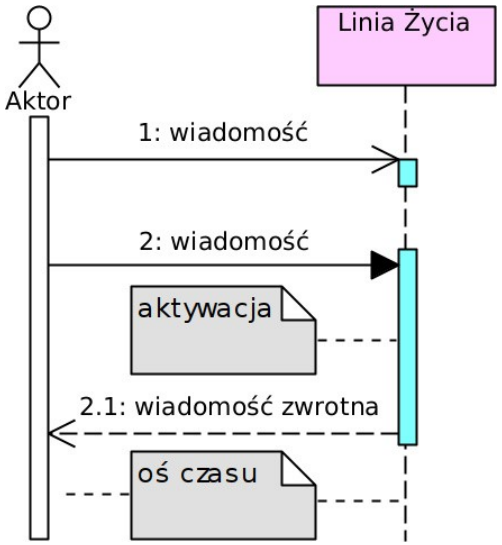
Diagram sekwencji

Diagram Sekwencji /sequence diagram/

- Modeluje **interakcję** – przepływ sterowania między jej uczestnikami, np.:
  - złożony przypadek użycia lub operację jego realizacji,
  - algorytm wykonania operacji klasy.
- **Uczestnik interakcji** – aktor /actor/ lub **linia życia** /lifeline/: klasa, instancja klasy (obiekt), komponent, system itd.
- Nazwa linii życia:
  - ogólna – dla diagramu konceptualnego;
  - ***instancja:klasa*** – uczestnikiem jest konkretna instancja klasy;
  - ***:klasa*** – uczestnikiem jest anonimowa instancja klasy lub sama klasa;
  - ***self*** – uczestnikiem jest posiadacz tego diagramu.

Linia życia /lifeline/

- Uczestnik interakcji + oś czasu jego funkcjonowania (im niżej, tym później).
- Do linii życia wchodzą relacje **wiadomości**, powodując:
  - utworzenie tej linii życia,
  - zakończenie tej linii życia,
  - przyjęcie informacji przez tę linię życia,
  - rozpoczęcie wykonywania operacji przez tę linię życia,
  - otrzymanie wyniku operacji wykonanej przez tę lub inną linię życia.



- Czas wykonywania operacji przez linię życia pokazuje **aktywacja** /activation, execution specification/ – wąski prostokąt na jej osi czasu (i opcjonalnie wymiarowanie z wartościami czasu).
- Gdy uczestnikiem interakcji jest **aktor** (byt spoza modelowanego systemu):
  - ikona i nazwa aktora są nad osią czasu,
  - wysyłanie i przyjmowanie wiadomości jest ograniczone.

Wiadomość /message/

- Przekazanie sterowania (i opcjonalnie obiektu) między uczestnikami interakcji: od nadawcy do odbiorcy;
- Wykonanie operacji przez odbiorcę w odpowiedzi na żądanie nadawcy.
- Wiadomości NIE dotyczące tej samej linii życia mogą być wykonywane w dowolnej kolejności.
- **Główne rodzaje wiadomości:**
  - asynchroniczna, – własna, – znaleziona, –trwająca.
  - synchroniczna, – rekursywna, – tworząca obiekt,
  - zwrotna, – zgubiona, – usuwająca obiekt,
- **Numer** wiadomości – kolejność jej wykonania:
  - pierwsza jest wiadomość nr 1,
  - numery mogą mieć podnumery itd. (np. 1.1).
- **Opis wiadomości** – zwykle operacja do wykonania przez odbiorcę.
  - Zwrot relacji wiadomości wskazuje odbiorcę wiadomości.

Opis wiadomości żądania wykonania operacji

- **Składnia:**

```
<request-message-label> ::= <message-name> [‘(‘[<input-argument-list>] ’)’]

<input-argument-list> ::= <input-argument> [‘,’<input-argument>]*

<input-argument> ::= [<in-parameter-name> ‘=’] <value-specification> | ‘-’
```

  - **<message-name>** – nazwa wiadomości (np. operacji),
  - **<in-parameter-name>** – nazwa wejściowego parametru wiadomości,
  - **<value-specification>** – parametr wiadomości.
- **Przykłady:**
  - `wiadomość`
  - `operacja()`
  - `operacja(parametr)`
  - `operacja(parametr1, parametr2)`
  - `operacja(nazwa1=parametr1, nazwa2=parametr2)`
- Często dodaje się na koniec typ wiadomości zwrotnej `:<return-type>`, np.: `operacja():int`

Opis wiadomości zwrotnej

- **Składnia:**

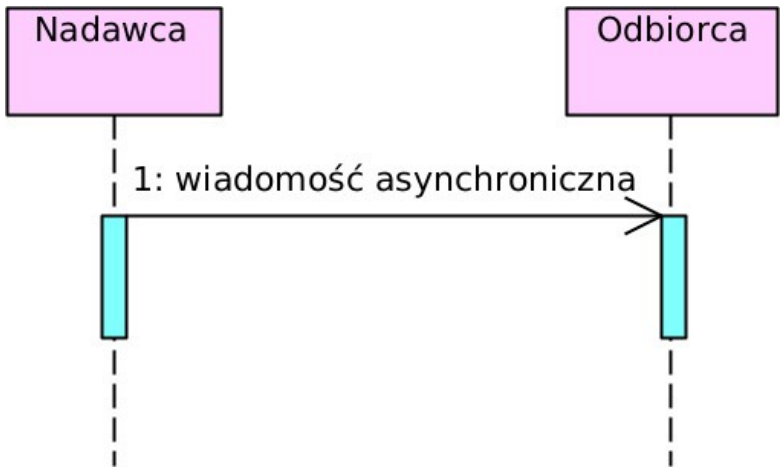
```
<reply-message-label> ::= [<assignment-target> ‘=’] <message-name>
            [‘(’ [<output-argument-list>] ‘)’] [‘:’ <value-specification>]

<output-argument-list> ::= <output-argument> [‘,’<output-argument>]*

<output-argument> ::= <out-parameter-name> ‘:’ <value-specification> |
<assignment-target> ‘=’ <out-parameter-name> [‘:’ <value-specification>]
```

  - **<assignment-target>** (dla wiadomości) – wyjściowy parametr wiadomości żądania, z którą związana jest wiadomość zwrotna (dla wiadomości zwrotnej lub jej parametru);
  - **<value-specification>** – typ wiadomości zwrotnej lub jej parametr;
  - **<message-name>** – nazwa lub treść wiadomości (np. nazwa operacji);
  - **<out-parameter-name>** – nazwa wyjściowego parametru wiadomości.

- **Przykłady:**
  - `wynik`
  - `wynik:string`
  - `out = wynik(-):45`
    gdzie *wynik()* to operacja, która kończy się tą wiadomością zwrotną
  - `out = wynik(nazwa1=parametr1:5, nazwa2=parametr2:10)`

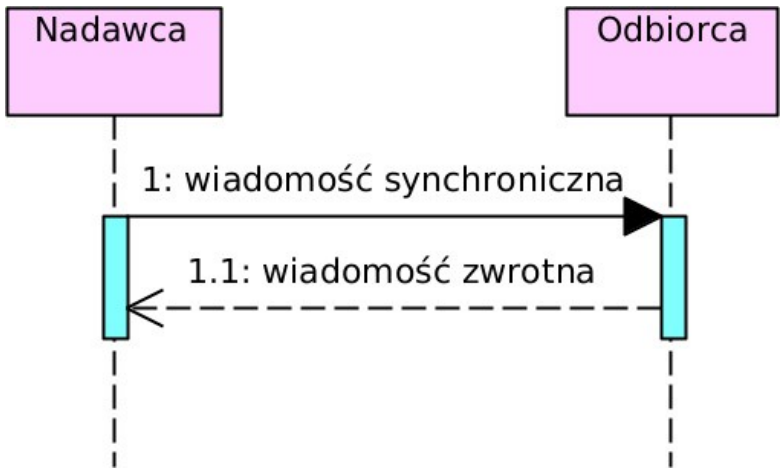
Wiadomość asynchroniczna /asynchronous/

- Asynchroniczne wywołanie operacji odbiorcy przez nadawcę:
  - nadawca NIE oczekuje na zakończenie tej operacji,
  - nadawca NIE otrzymuje wiadomości zwrotnej od odbiorcy.
- **Odbiorcą** może być nadawca lub inny uczestnik interakcji.
- **Opis relacji wiadomości** – operacja do wykonania przez odbiorcę.
- **Linia relacji wiadomości** – ciągła z otwartym grotem wskazującym wykonawcę operacji.



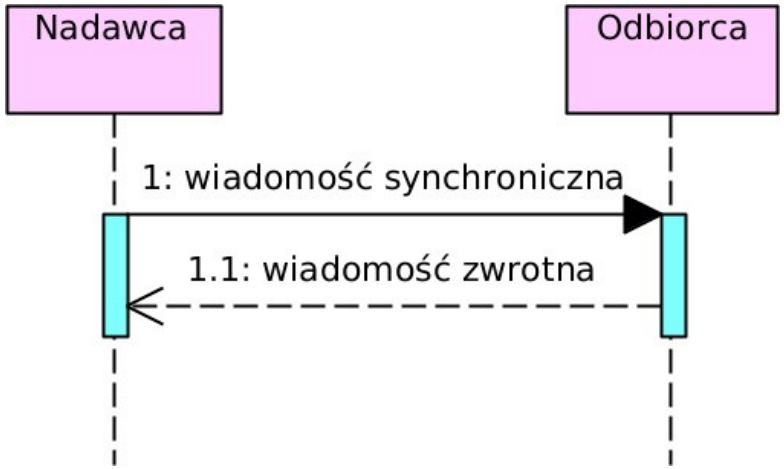
Wiadomość synchroniczna /synchronous/

- Synchroniczne wywołanie operacji odbiorcy przez nadawcę:
  - nadawca oczekuje na zakończenie tej operacji,
  - nadawca może otrzymać wiadomość zwrotną od odbiorcy.
- **Odbiorcą** może być nadawca lub inny uczestnik interakcji.
- **Opis relacji wiadomości** – operacja do wykonania przez odbiorcę.
- **Linia relacji wiadomości** – ciągła z zamkniętym grotem wskazującym wykonawcę operacji.



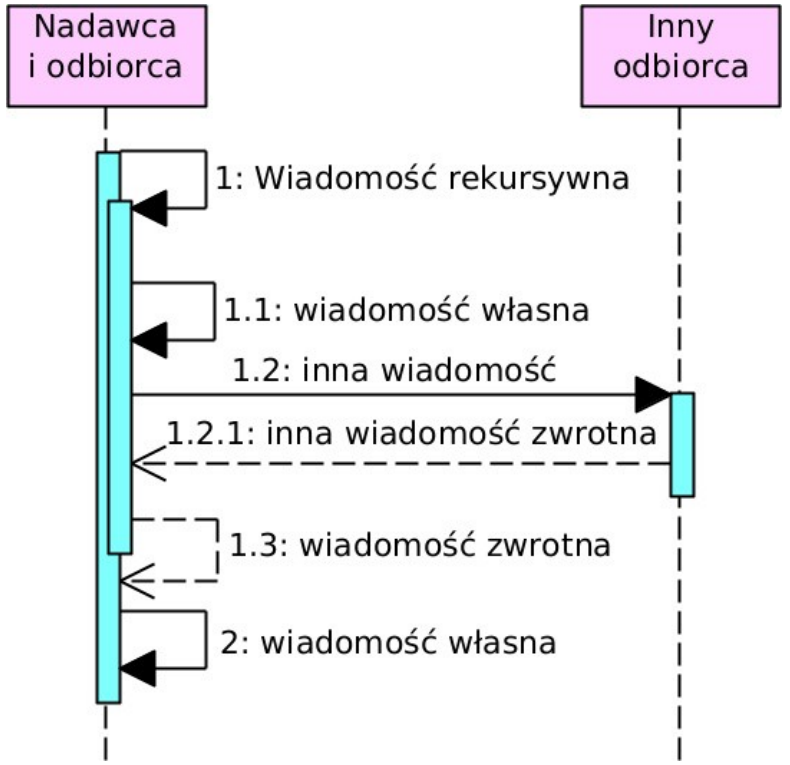
Wiadomość zwrotna /reply/

- Kończy wykonywanie operacji przez nadawcę:
  - nadawca zwraca sterowanie odbiorcy,
  - może też przekazać odbiorcy wynik kończonej operacji.
- **Odbiorcą** jest nadawca synchronicznej wiadomości, która wywołała kończoną operację (wiadomość zwrotna wchodzi do aktywacji odbiorcy).
- Aktywacja nadawcy kończy się po wysłaniu wiadomości zwrotnej.
- **Opis relacji wiadomości** – nic lub dane zwracane przez operację.
- **Linia relacji wiadomości** – przerywana z otwartym grotem wskazującym odbiorcę.



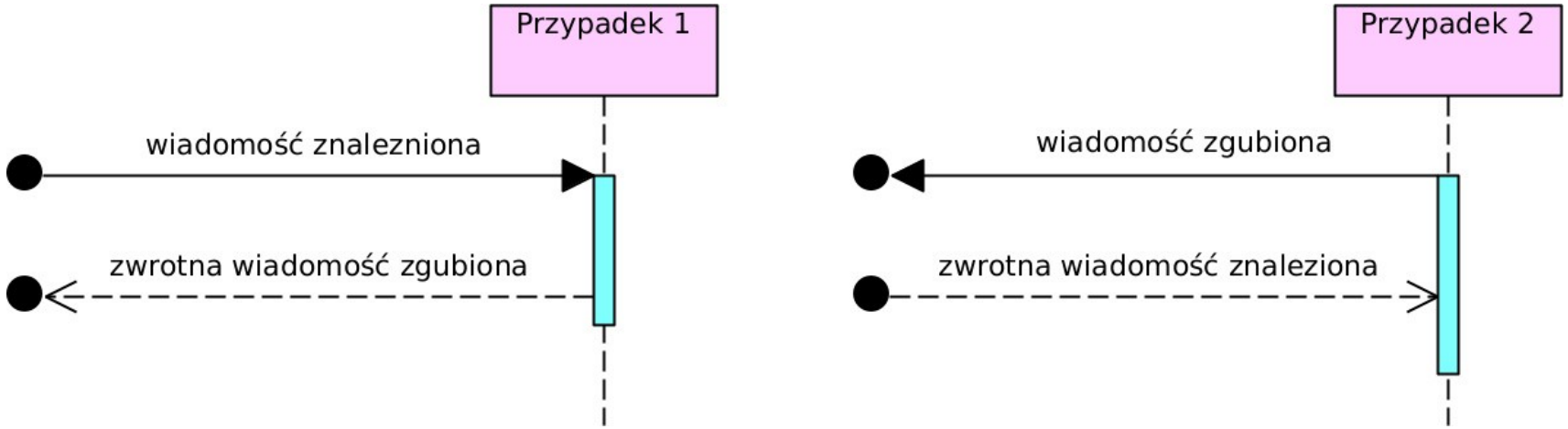
Wiadomość własna i rekursywna

- Wywołanie operacji odbiorcy przez nadawcę – nadawca jest odbiorcą.
- **Opis relacji wiadomości** – operacja do wykonania przez odbiorcę.
- **Linia relacji wiadomości** – zgięta w kształt ⊐, ciągła, wskazująca wykonawcę operacji.
- **Wiadomość własna** /self/
  - Wykonywana operacja NIE powoduje wysłania innych wiadomości
    - nawet tym wiadomości zwrotnej.
  - NIE umieszcza na osi czasu odbiorcy aktywacji wykonania tej operacji.
- **Wiadomość rekursywna** /recursive/
  - Wykonywana operacja powoduje wysłanie innych wiadomości.
  - Umieszcza na osi czasu odbiorcy aktywację wykonania tej operacji.



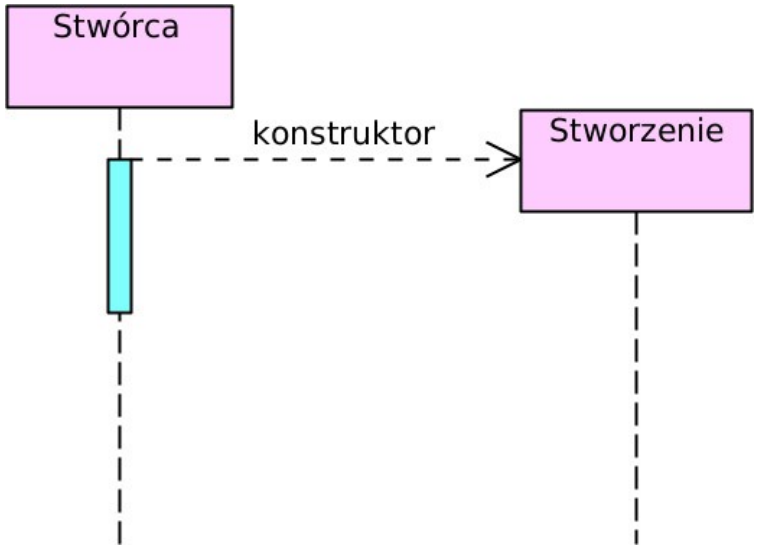
Wiadomość zgubiona i znaleziona

- **Wiadomość zgubiona** /lost/
  - Odbiorca wiadomości nie jest znany lub nie istnieje.
  - **Linia relacji wiadomości** – wychodzi z osi czasu nadawcy i wskazuje na węzeł *czarne koło*.
- **Wiadomość znaleziona** /found/
  - Nadawca wiadomości nie jest znany.
  - **Linia relacji wiadomości** – wychodzi z węzła *czarne koło* i wskazuje na oś czasu odbiorcy.
- **Opis relacji wiadomości** – operacja do wykonania przez odbiorcę.



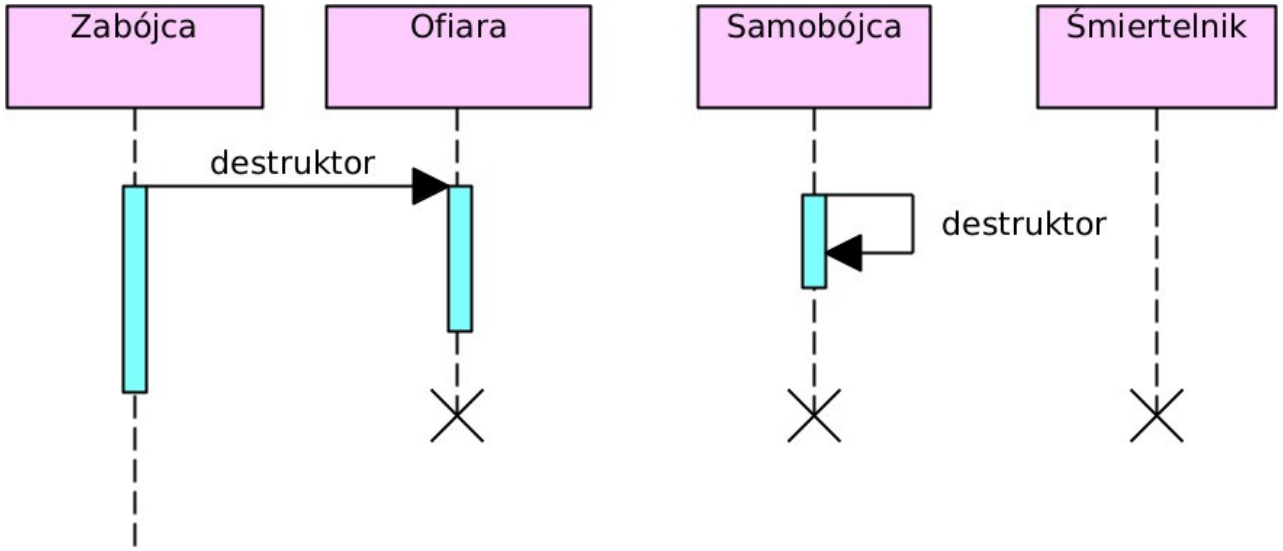
Wiadomość tworząca obiekt /object creation/

- Utworzenie nowej linii życia (obiektu, np. instancji klasy) przez inną.
- **Linia relacji wiadomości** – przerywana z otwartym grotem wskazującym nazwę tworzonej linii życia.
- **Nadawcą** NIE jest odbiorca.
- Nadawca NIE otrzymuje wiadomości zwrotnej od odbiorcy.
- **Opis relacji wiadomości** – operacja utworzenia obiektu (konstruktor).



Wiadomość usuwająca obiekt /object deletion/

- Usunięcie linii życia obiektu (np. instancji klasy) przez nią samą lub inną.
- **Linia relacji wiadomości** – ciągła z grotem wskazującym oś czasu usuwanej linii życia.
- Następnie usuwana linia życia kończy się znakiem X.
- **Nadawcą** może być odbiorca.
- Nadawca może otrzymać wiadomość zwrotną od odbiorcy (jeśli nie jest odbiorcą).
- **Opis relacji wiadomości** – operacja usunięcia obiektu (destruktor).



Parametry czasowe

- Określają ilościowo w nawiasach { } i przy pomocy „wymiarowania”:
  - moment wysłania lub otrzymania wiadomości,
  - długość czasu wysyłania wiadomości,
  - odstęp czasu między zdarzeniami wysłania lub otrzymania wiadomości.
- Konkretne liczby lub symbolizujące je zmienne, które mogą być zależne od:
  - ***now*** – zaobserwowany moment czasu,
  - ***duration*** – zaobserwowana długość czasu.

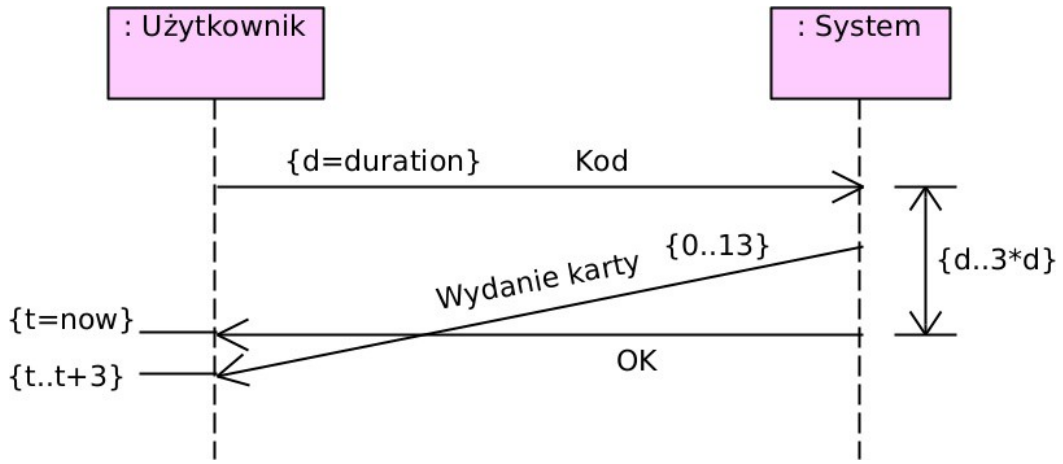
Wiadomość trwająca /duration message/

- Otrzymanie wiadomości następuje zauważalny czas po jej nadaniu.
- **Relacja wiadomości:**
  - jest skierowana pochyło w dół (nie jest pozioma),
  - może przecinać inne relacje wiadomości.

Przykład parametrów czasowych i wiadomości trwającej

na podst. Unified Modeling Language (UML)

- **Użytkownik** wysyła do **Systemu** wiadomość **Kod**:
  - czas wysyłania **Kod** jest mierzony i zapisywany jako **d**.
- **System** odpowiada wysłaniem do **Użytkownika** wiadomości trwającej **Wydanie karty**, a następnie wiadomości **OK**:
  - czas wysyłania **Wydanie karty** trwa od 0 do 13 jednostek;
  - odstęp czasu między wysłaniem (i odebraniem) **Kod**, a wysłaniem (i odebraniem) **OK** wynosi od **d** do **3*d**;
  - moment wysłania (i otrzymania) **OK** jest zapisywany jako **t**.
  - moment otrzymania **Wydanie karty** wynosi od **t** do **t+3**.



Połączony fragment /combined fragment/

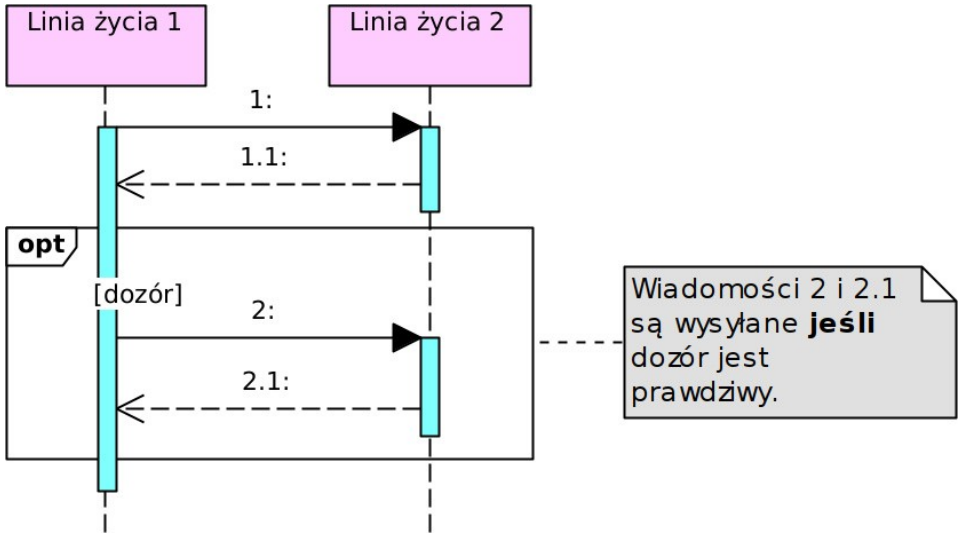
- Otacza część diagramu, aby nadać jej określone znaczenie:
  - obejmuje określony czas (przez rozciągnięcie w pionie),
  - obejmuje określone linie życia (przez rozciągnięcie w poziomie),
  - obejmuje relacje wiadomości przechodzące między tymi liniami życia.
- **Operator** /operator/ – określa interpretację zawartości połączonego fragmentu.
- **Operand** /operand/ – część połączonego fragmentu (ich liczba zależy od operatora).
  - Operandy rozdzielane są poziomą przerywaną linią.
  - Może mieć **dozór** /guard/ w nawiasach [ ]:
    - logiczny warunek uwzględnienia zawartych relacji wiadomości,
    - brak dozoru oznacza dozór [*true*].
  - Dotyczy tylko tych relacji wiadomości, które są wysyłane lub otrzymywane przez linię życia objętą tym operandem.

Rodzaje operatorów

- ***opt*** – opcja (1 operand w połączonym fragmencie),
- ***alt*** – alternatywy (2 lub więcej),
- ***par*** – współbieżność (2 lub więcej),
- ***seq*** – słaba kolejność (2 lub więcej),
- ***strict*** – ścisła kolejność (2 lub więcej),
- ***neg*** – niepoprawność (1),
- ***critical*** – region krytyczny (1),
- ***ignore*** – pominięcie (1),
- ***consider*** – uwzględnienie (1),
- ***assert*** – założenie (1),
- ***loop*** – pętla (1),
- ***break*** – opuszczenie (1),
- ***ref*** – użycie interakcji (1).

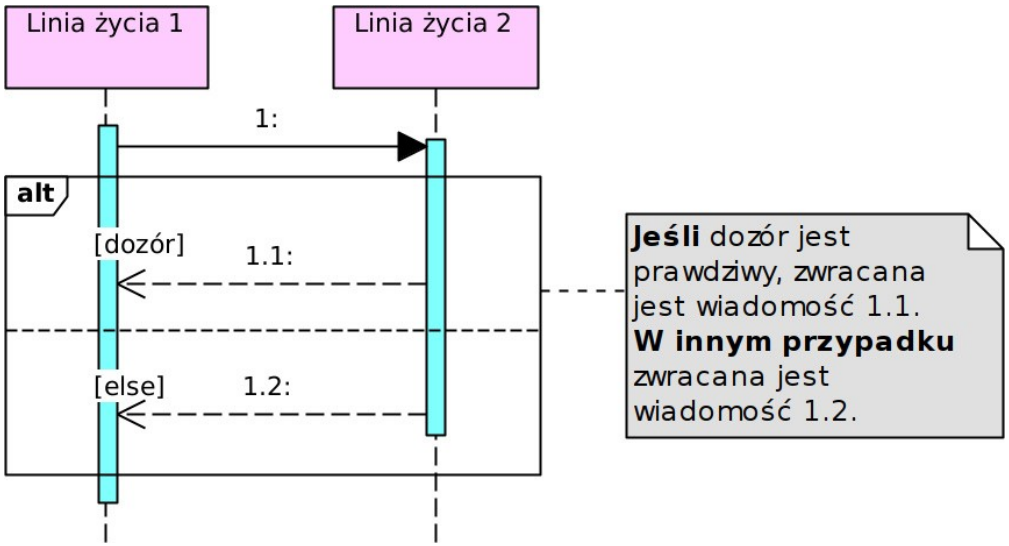
opt (opcja /option/)

- ***opt*** (1 operand):
  - jeśli dozór operandu jest prawdziwy, wykonywane są interakcje zawarte w operandzie;
  - w innym przypadku są pomijane.
- Odpowiada konstrukcji *if* (BEZ *else*).



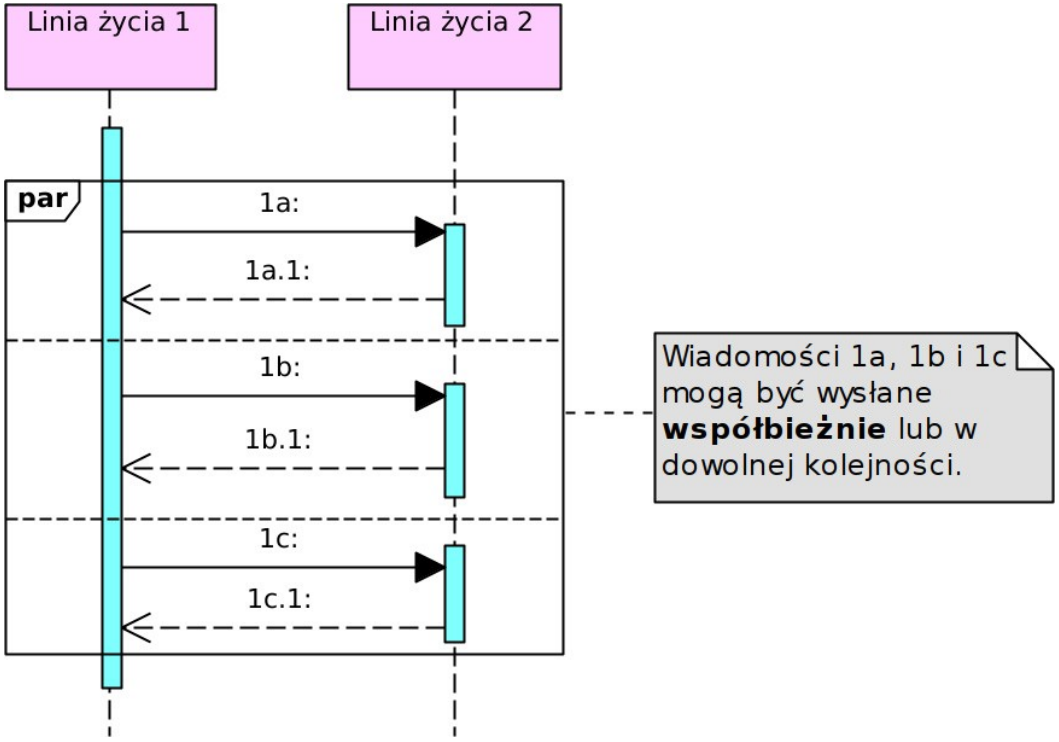
alt (alternatywy /alternatives/)

- ***alt*** (2 lub więcej operandów):
  - sprawdzana jest prawdziwość dozorów kolejnych operandów (od góry);
  - jeśli dozór danego operandu jest prawdziwy:
    - wykonywane są interakcje zawarte w tym operandzie,
    - interakcje zawarte w pozostałych operandach są pomijane;
  - w innym przypadku jego interakcje są pomijane;
  - dozór [*else*] w ostatnim operandzie oznacza wszystkie inne przypadki.
- Odpowiada konstrukcjom *if*… *else* i *switch* (z *break*).



par (współbieżność /parallel/)

- ***par*** (2 lub więcej operandów):
  - interakcje zawarte w tym samym operandzie wykonywane są normalnie,
  - wszystkie operandy są współbieżne:
    - interakcje zawarte w różnych operandach mogą być wykonywane współbieżnie lub w dowolnej kolejności.



seq (słaba kolejność /weak sequencing/)

- ***seq*** (2 lub więcej operandów):
  - interakcje zawarte w tym samym operandzie wykonywane są normalnie,
  - wszystkie operandy są częściowo współbieżne:
    - interakcje zawarte w różnych operandach i dotyczące różnych linii życia mogą być wykonywane współbieżnie lub w dowolnej kolejności;
    - interakcje zawarte w różnych operandach i dotyczące tej samej linii życia wykonywane są normalnie i w kolejności ich operandów.

strict (ścisła kolejność /strict sequencing/)

- ***strict*** (2 lub więcej operandów):
  - interakcje zawarte w tym samym operandzie wykonywane są normalnie,
  - interakcje zawarte w różnych operandach wykonywane są w kolejności ich operandów, nawet jeśli dotyczą różnych linii życia.

neg (niepoprawność /negative/)

- ***neg*** (1 operand):
  - interakcje zawarte w operandzie są niepoprawne (nie powinny być wykonane).

critical (region krytyczny /critical region/)

- ***critical*** (1 operand):
  - podczas wykonywania interakcji zawartych w operandzie, nie mogą być wykonywane interakcje spoza niego, które dotyczą tych samych linii życia
  - także wtedy, gdy połączony fragment *critical* znajduje się w połączonym fragmencie *par*.

ignore (pominięcie)

- ***ignore{wiadomości}*** (1 operand):
  - wylicza wiadomości niepokazywane przez operand:
    - mogą być przekazywane, ale są nieistotne;
    - ich przekazywanie jest ignorowane.
  - `wiadomości ::= nazwa-wiadomości[‘,’ nazwa-wiadomości]*`
- Operator *ignore* po nazwie diagramu dotyczy całego diagramu.
- Przydatne w specyfikacji testu systemu (w tym oprogramowania).

consider (uwzględnienie)

- ***consider{wiadomości}*** (1 operand):
  - wylicza jedynie wiadomości pokazywane przez operand:
    - INNE wiadomości mogą być przekazywane, ale są nieistotne;
    - przekazywanie INNYCH wiadomości jest ignorowane.
  - `wiadomości ::= nazwa-wiadomości[‘,’ nazwa-wiadomości]*`
- Operator *consider* po nazwie diagramu dotyczy całego diagramu.
- Przydatne w specyfikacji testu systemu (w tym oprogramowania).

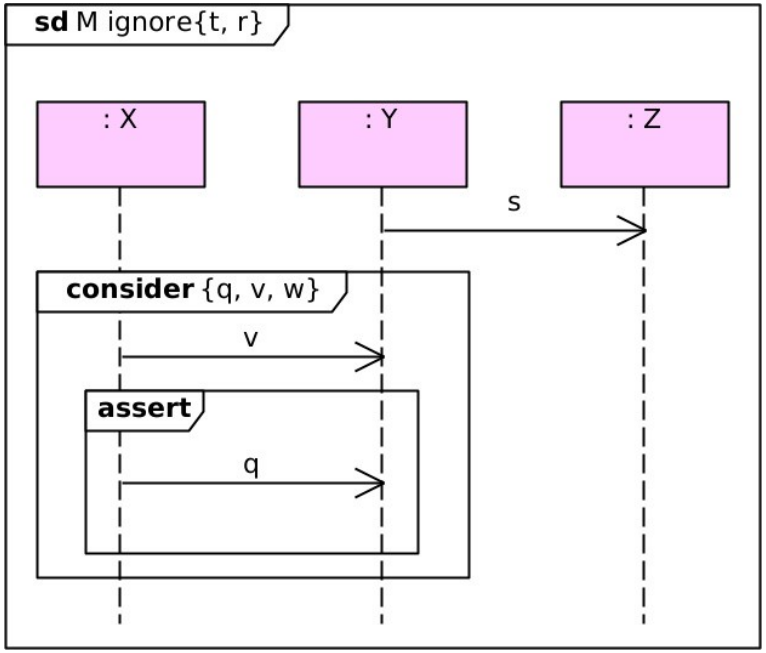
assert (założenie /assertion/)

- ***assert*** (1 operand):
  - zakłada (narzuca) wykonanie interakcji zawartych w operandzie:
    - w miejscu położenia tego połączonego fragmentu dozwolone są TYLKO interakcje zawarte w operandzie,
    - wszystkie inne WTEDY NIE są wykonywane.
- Uzupełniają połączone fragmenty *ignore* i *consider*.
- Przydatne w specyfikacji testu systemu (w tym oprogramowania).

Przykład połączonych fragmentów *ignore*, *consider* i *assert* w modelu testu systemu

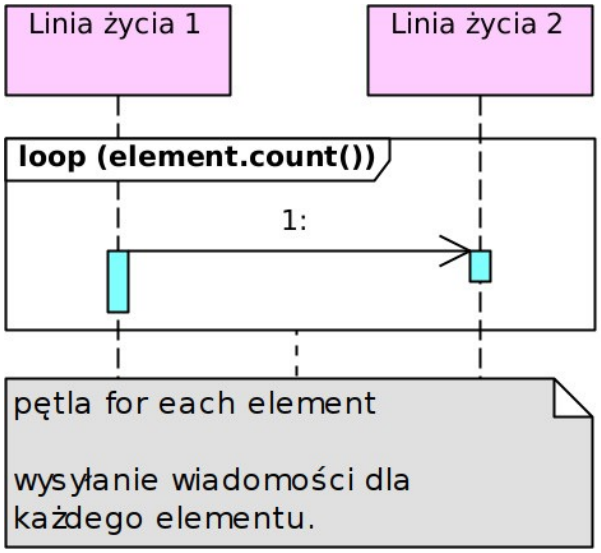
na podst. Unified Modeling Language (UML)

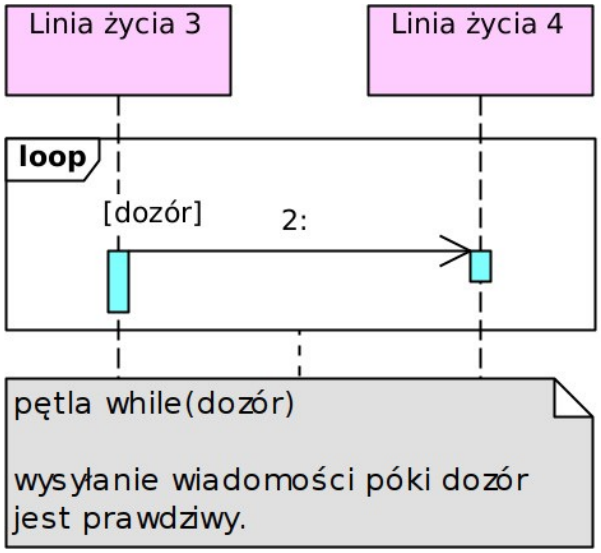
- Diagram **M** modeluje test systemu z liniami życia **X**, **Y** i **Z**, ignorując wykonywanie wiadomości **t** i **r**.
  - Dla testu nie jest ważne, jak system obsługuje ich wykonywanie.
- Po wykonaniu wiadomości **s** mogą być wykonywane dowolne wiadomości (poza **t** i **r**), ale:
  - tylko wykonywanie wiadomości **q**, **v** i **w** jest dalej uwzględniane w teście;
  - zaraz po wykonaniu wiadomości **v**:
    - zakłada się, że nastąpi wykonanie wiadomości **q**;
    - wykonanie innej wiadomości zamiast niej jest niepoprawne.

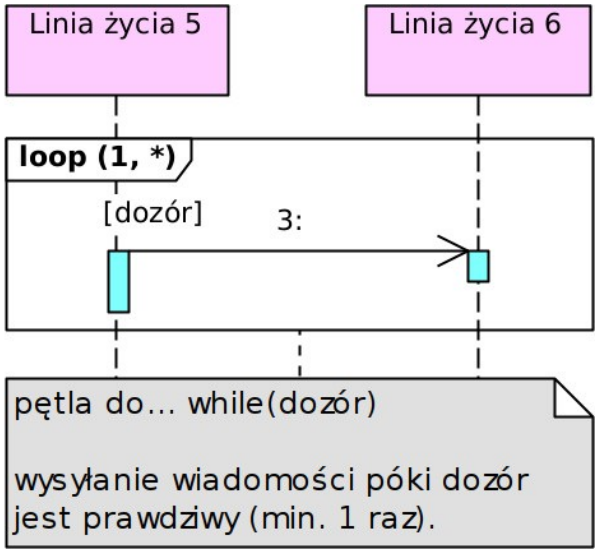


loop (pętla)

- ***loop(min,max)*** (1 operand):
  - interakcje zawarte w operandzie wykonywane są normalnie;
  - to wykonanie odbywa się przynajmniej *min* razy i najwyżej *max* razy:
    - *loop(ile)* oznacza *loop(ile,ile)* – dokładnie *ile* razy,
    - brak *(min,max)* oznacza *loop(0,*)* – 0 lub dowolnie wiele razy,
    - nie powtórzy się, gdy odbyło się choć *min* razy i dozór jest fałszywy.
  - w innym przypadku są pomijane.
- Odpowiada konstrukcjom *for, do… while(…)* i *while(…)*.

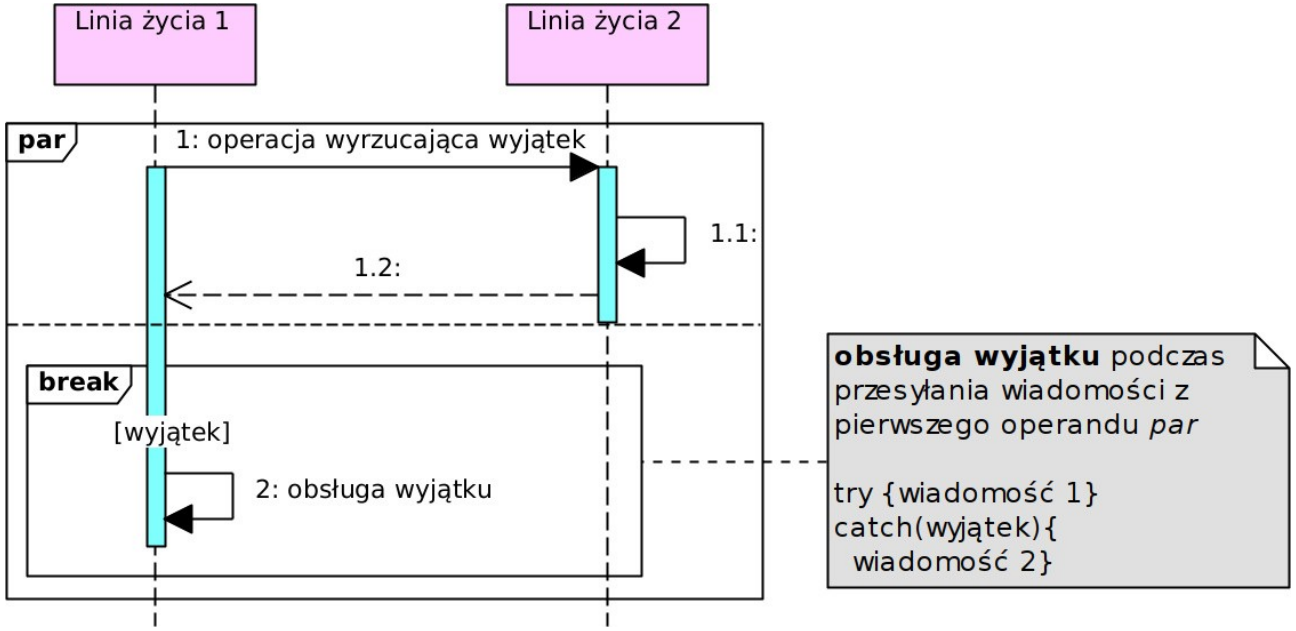






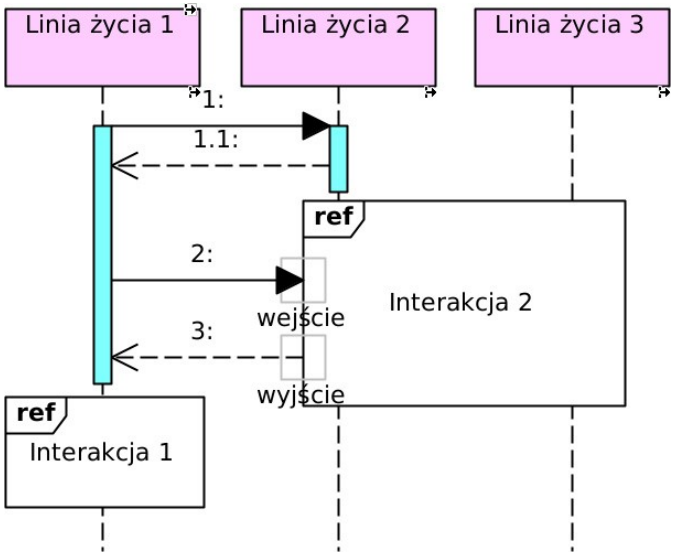
break (opuszczenie)

- ***break*** (1 operand):
  - całkowicie zawiera się w innym połączonym fragmencie;
  - kiedy dozór operandu jest prawdziwy:
    - inne interakcje zawierającego fragmentu są pomijane,
    - wykonywane są interakcje zawarte w operandzie;
  - w innym przypadku są pomijane;
  - brak dozoru oznacza jego losową prawdziwość.
- Odpowiada konstrukcjom *break* wyjścia z pętli i *catch* w obsłudze wyjątków.

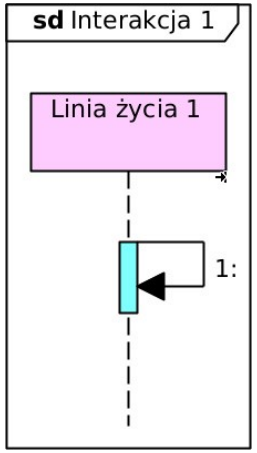


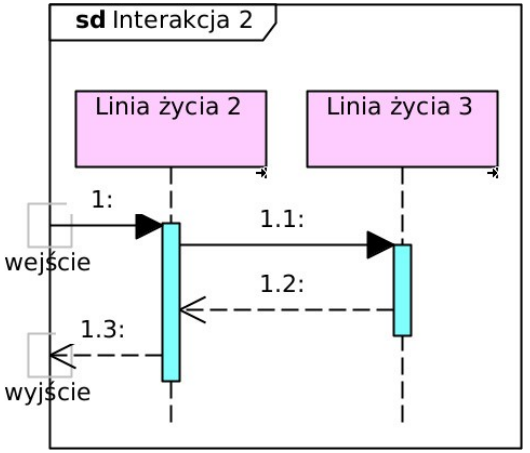
ref (użycie interakcji /interaction use/)

- ***ref*** (1 operand):
  - zastępuje część interakcji diagramu:
    - która znajduje się na innym diagramie;
    - obejmuje linie życia, które tam też są;
    - ma nazwę (w tym nazwę tego diagramu);
    - NIE zawiera żadnych interakcji.



- Pozwala wielokrotnie użyć tę samą interakcję i oczyścić diagram ze „zbędnych” interakcji.
- Do połączonego fragmentu *ref* mogą wchodzić i wychodzić wiadomości:
  - przez nazwane **bramki**,
  - diagram związany z tym fragmentem ma tak samo nazwane bramy,
  - lub bezpośrednio.





ref (użycie interakcji)

- **Składnia nazwy:**

```
<name> ::= [<attribute-name> '=' ] [<collaboration-use> '.']
    <interaction-name> ['(' <io-argument> [',' <io-argument>]* ')']
    [':' <return-value>]

<io-argument> ::= <in-argument> | 'out' <out-argument>
```

<interaction-name> – nazwa interakcji i związanego z nią diagramu;

<attribute-name> – atrybut linii życia (z diagramu zawierającego ten połączony fragment *ref*), do którego trafi wartość zwracana przez interakcję;

<collaboration-use> – określenie części większej interakcji;

<in-argument> – wejściowy parametr interakcji;

<out-argument> – wyjściowy parametr interakcji;

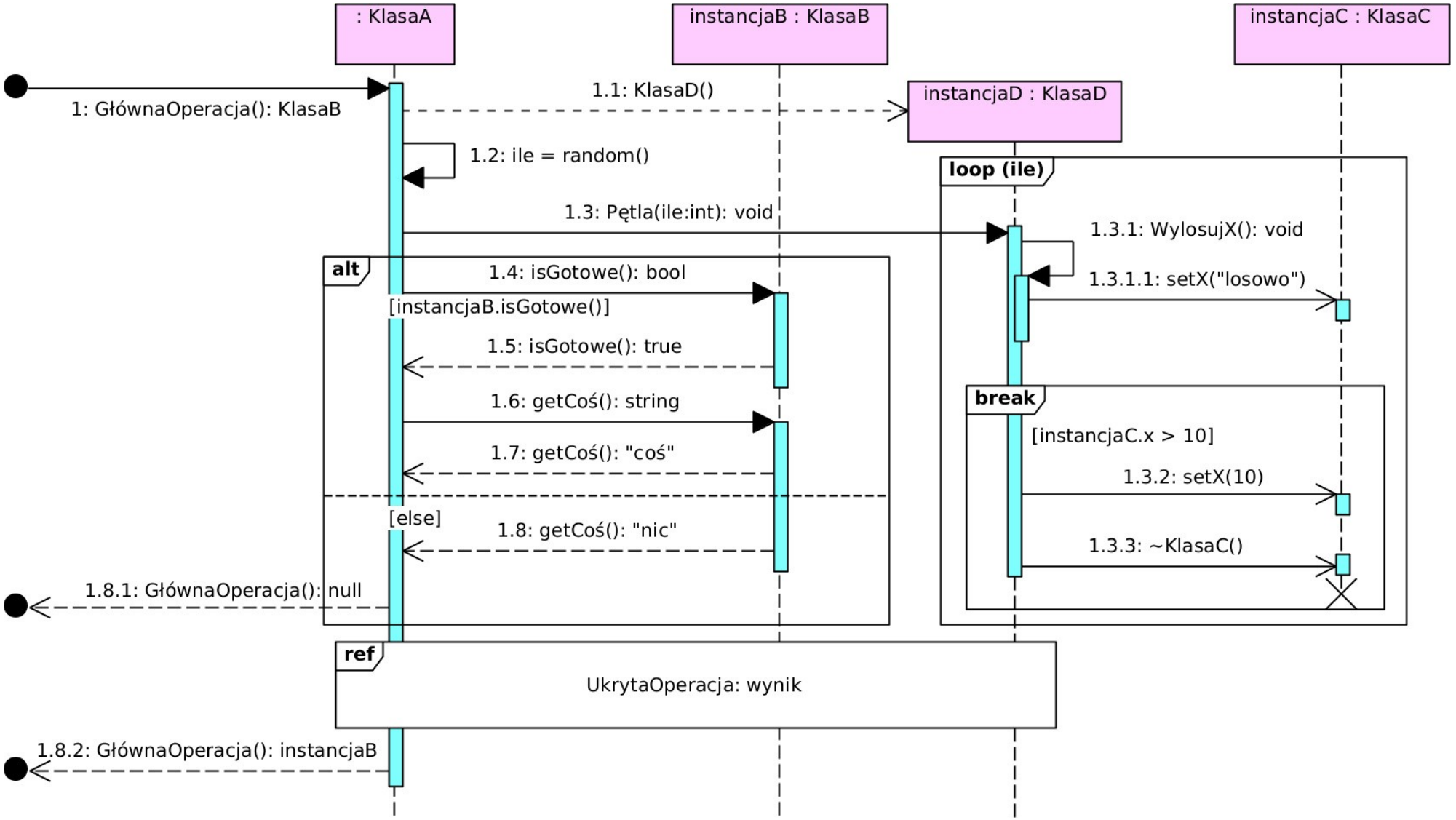
<return-value> – wartość zwracana przez interakcję.

- **Przykłady:**

```
Nazwa diagramu

Operacja(parametr1, parametr2)

liniaŻycia.atrybut = Operacja(parametr): wynikInterakcji
```

Przykład implementacyjnego modelu operacji klasy

- Wykonanie operacji **GłównaOperacja()** klasy **KlasaA**.



8

Diagram przeglądu interakcji

Diagram przeglądu interakcji
/interaction overview diagram/

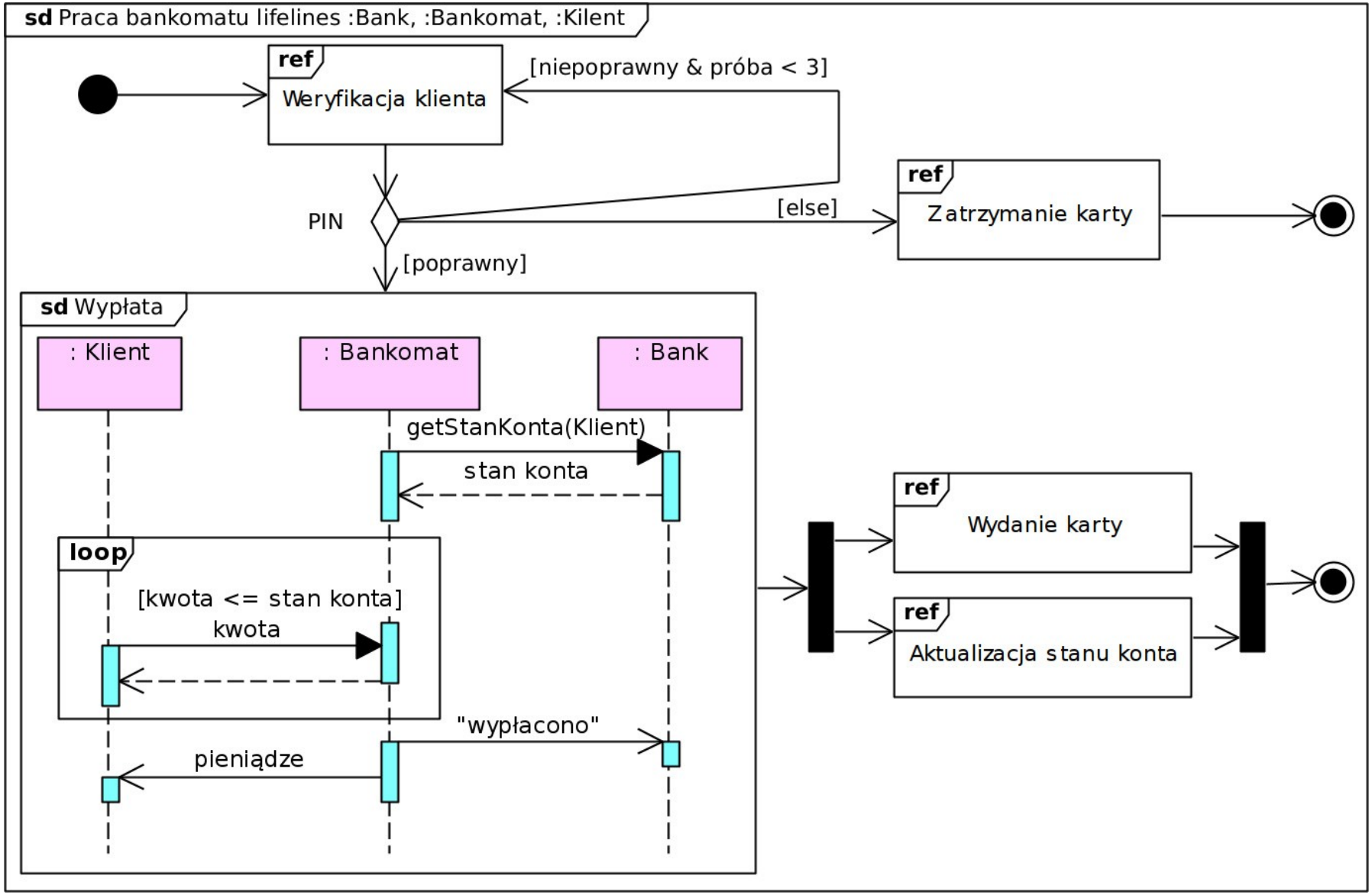
- Modeluje **interakcję** – przepływ sterowania między jej uczestnikami, np.:
  - złożony przypadek użycia lub operację jego realizacji,
  - algorytm wykonania operacji klasy.
- Diagram czynności:
  - wykorzystuje m.in. węzły decyzji i współbieżności,
  - zawiera zamiast akcji i czynności:
    - użycie interakcji (ramkę *ref* – referencja do innego diagramu);
    - diagram interakcji (ramkę z diagramem interakcji, np. sekwencji).
  - wymienia uczestników interakcji (po nazwie diagramu):

```
sd <diagram-name> lifelines <lifelines>
<lifelines> ::= <lifeline> [‘,’ <lifeline>]*
```

- **Uczestnik interakcji** – aktor /actor/ lub **linia życia** /lifeline/: klasa, instancja klasy (obiekt), komponent, system itd.
  - Może być pokazany tylko w ramce zawartego diagramu.

Przykład diagramu przeglądu interakcji

- Interakcje między liniami życia **:Bank**, **:Bankomat** i **:Klient**.



9

Diagram czasowy

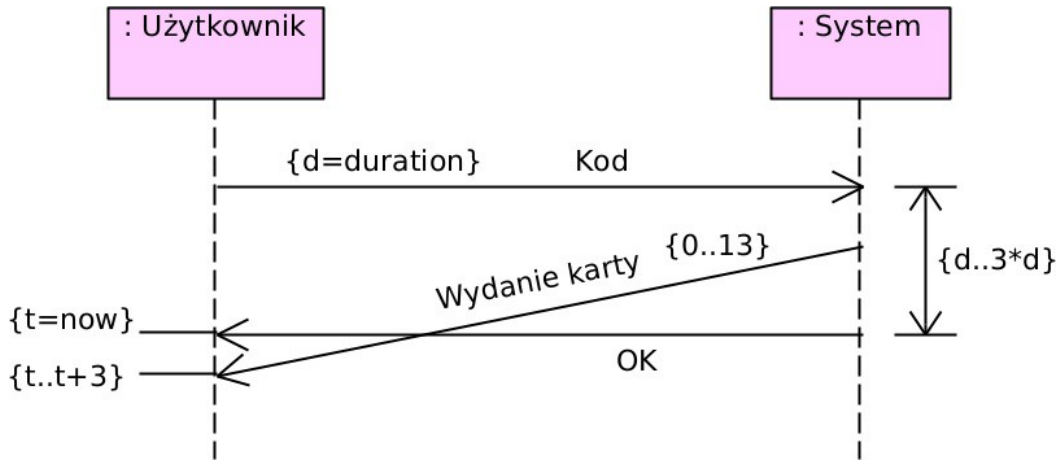
Diagram czasowy /timing diagram/

- Pokazuje **zmiany stanów linii życia w czasie:**
  - pionowa oś pokazuje różne stany,
  - pozioma oś pokazuje zdarzenia zmiany stanów (czas biegnie w prawo) oraz ich parametry czasowe.
- Nazwa linii życia:
  - ogólna – dla diagramu konceptualnego;
  - ***instancja:klasa*** – uczestnikiem jest konkretna instancja klasy;
  - ***:klasa*** – uczestnikiem jest anonimowa instancja klasy lub sama klasa;
  - ***self*** – uczestnikiem jest posiadacz tego diagramu.
- Linie życia umieszcza się w osobnych torach (odpowiednik partycji z diagramu czynności).
- **Wiadomość** może łączyć zdarzenia zmiany stanu dwóch linii życia:
  - zdarzenie wysłania wiadomości → zdarzenie otrzymania wiadomości.

Przykład z diagramem sekwencji ze str. 96

na podst. Unified Modeling Language (UML)

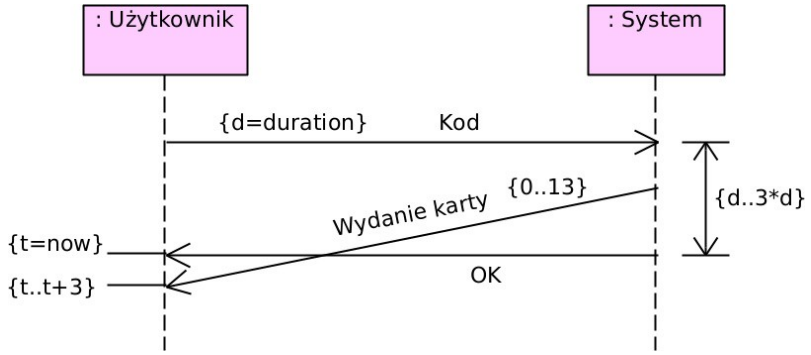
- **Użytkownik** wysyła do **Systemu** wiadomość **Kod**:
  - czas wysyłania **Kod** jest mierzony i zapisywany jako **d**.
- **System** odpowiada wysłaniem do **Użytkownika** wiadomości trwającej **Wydanie karty**, a następnie wiadomości **OK**:
  - czas wysyłania **Wydanie karty** trwa od 0 do 13 jednostek;
  - odstęp czasu między wysłaniem (i odebraniem) **Kod**, a wysłaniem (i odebraniem) **OK** wynosi od **d** do **3*d**;
  - moment wysłania (i otrzymania) **OK** jest zapisywany jako **t**.
  - moment otrzymania **Wydanie karty** wynosi od **t** do **t+3**.

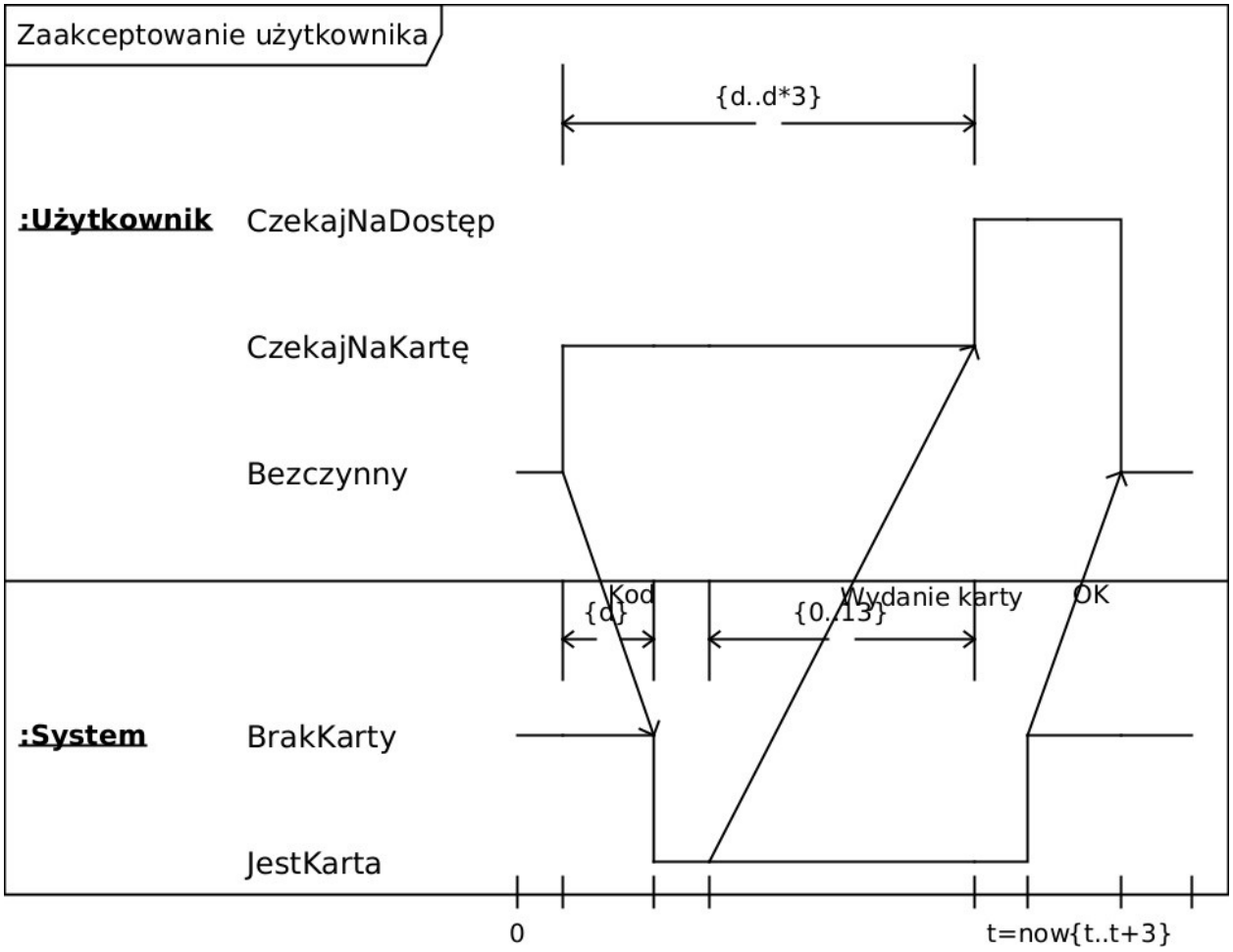


Przykład diagramu z 2 liniami życia i wiadomościami

na podst. Unified Modeling Language (UML)

- Ta sama interakcja między **Użytkownikiem** i **Bankiem**.
- Stany Użytkownika:
  - **CzekajNaDostęp**,
  - **CzekajNaKartę**,
  - **Bezczynny**.
- Stany Systemu:
  - **BrakKarty**,
  - **JestKarta**.





Temat następnej prezentacji

Język UML – diagramy struktury