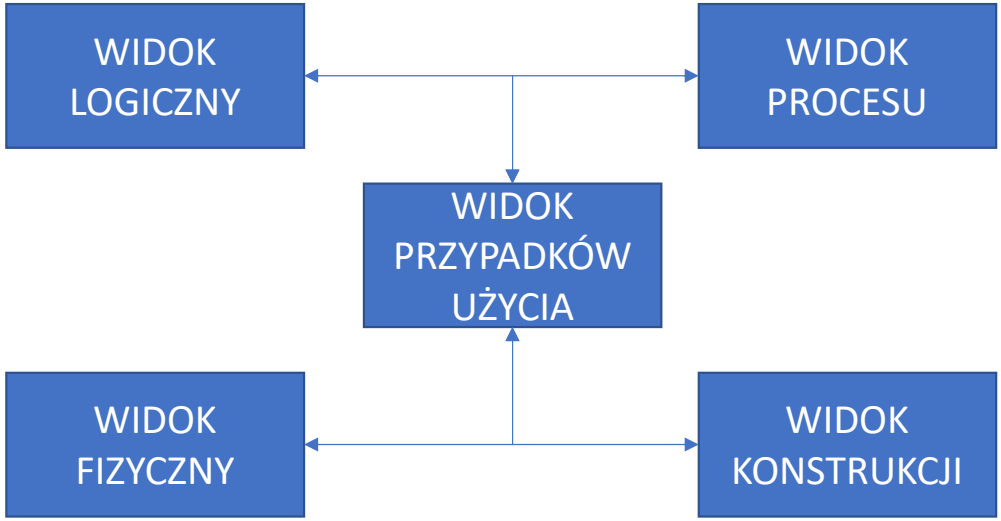
UML

Podsumowanie

Rodzaje widoków na modelowany system

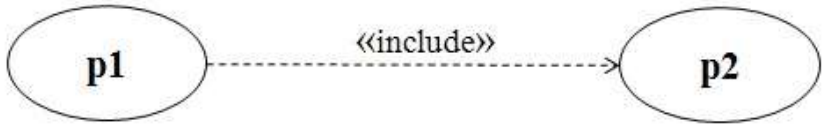


Widok logiczny w systemie 4+1 Kruchtena używany jest do modelowania części systemu oraz sposobów, w jaki one ze sobą współdziałają.

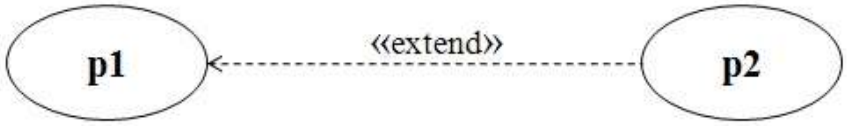
Ten widok tworzą zazwyczaj diagramy:

- klas
- obiektów
- maszyny stanowej
- interakcji

Diagram Przypadków Użycia



Przebieg podstawowy (sekwencyjny): p1 zawsze włącza (używa) p2.



Przebieg opcjonalny (alternatywny): p1 jest czasami rozszerzane o p2 (inaczej: p2 czasami rozszerza p1).

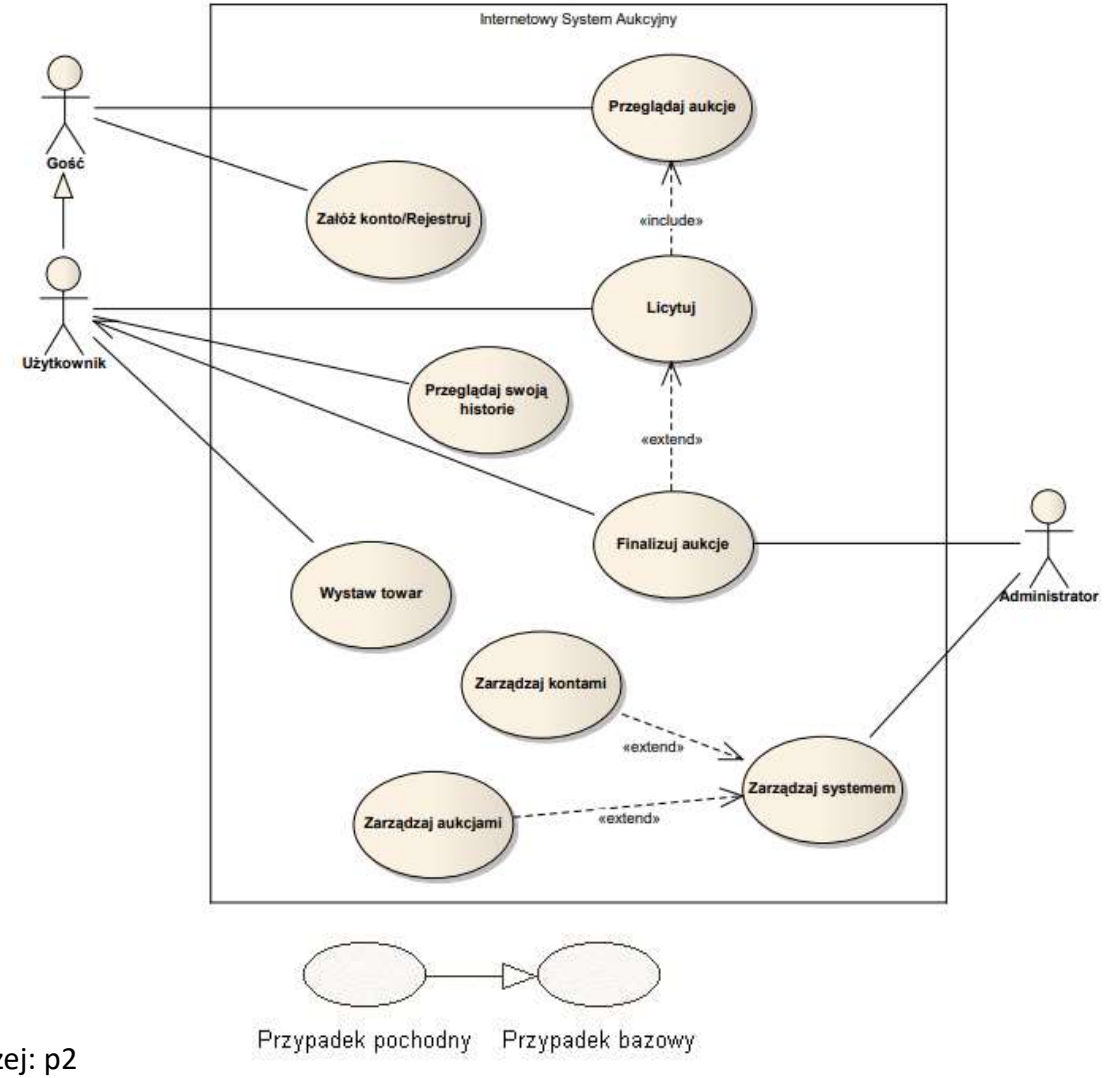
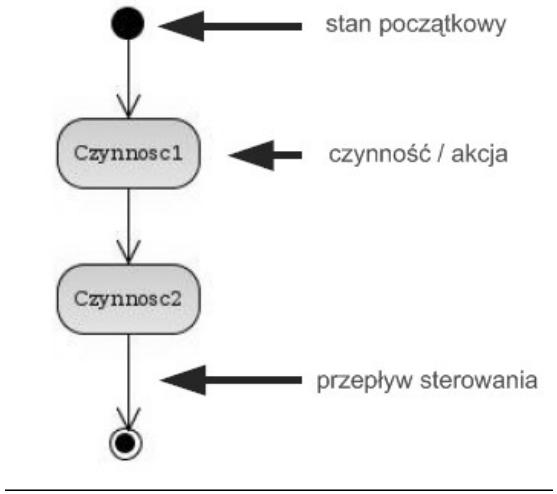
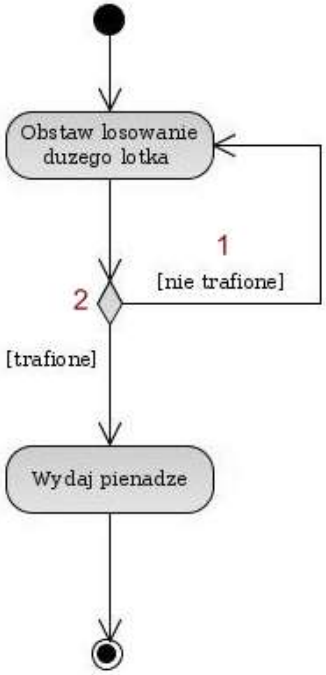
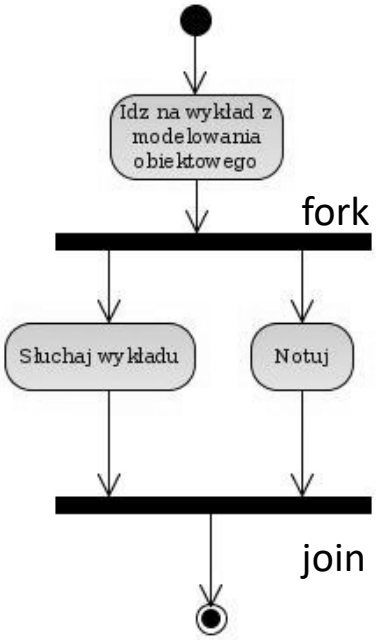
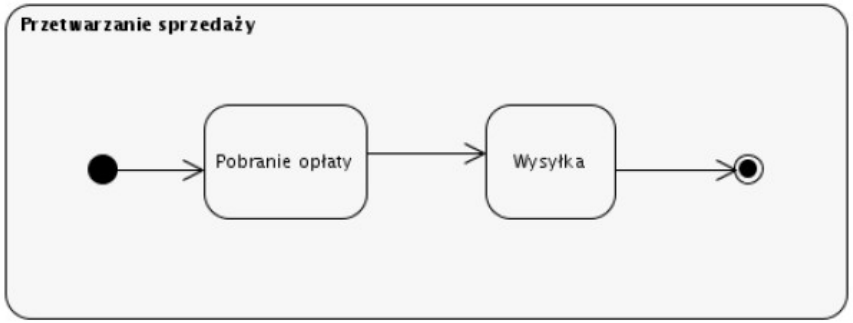


Diagram aktywności







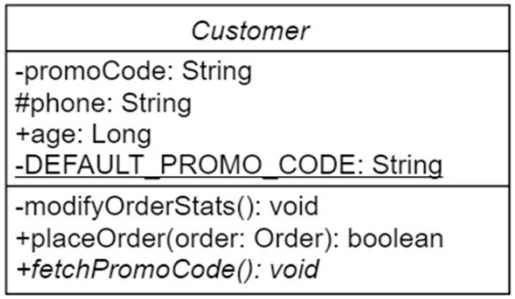


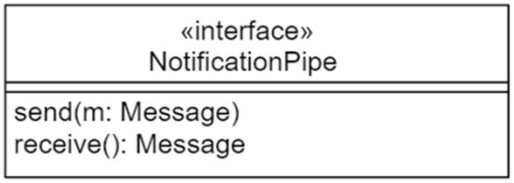
Ścieżki synchroniczne

Ścieżki współbieżne

Czynności złożone, pokazują wewnętrzne czynności (oznaczenie*) lub akcje:

Diagram klas





\+ – element publiczny,
\# – element „chroniony” (może odpowiadać protected w języku Java),
\- – element prywatny.

Implementacja

Dziedziczenie

Zależność

Zależność oznacza, że jedna klasa w pewnym momencie używa innej, na przykład jako parametr, czy wartość zwracana metody.

Asocjacja.

Zapis, który może zastąpić atrybut klasy

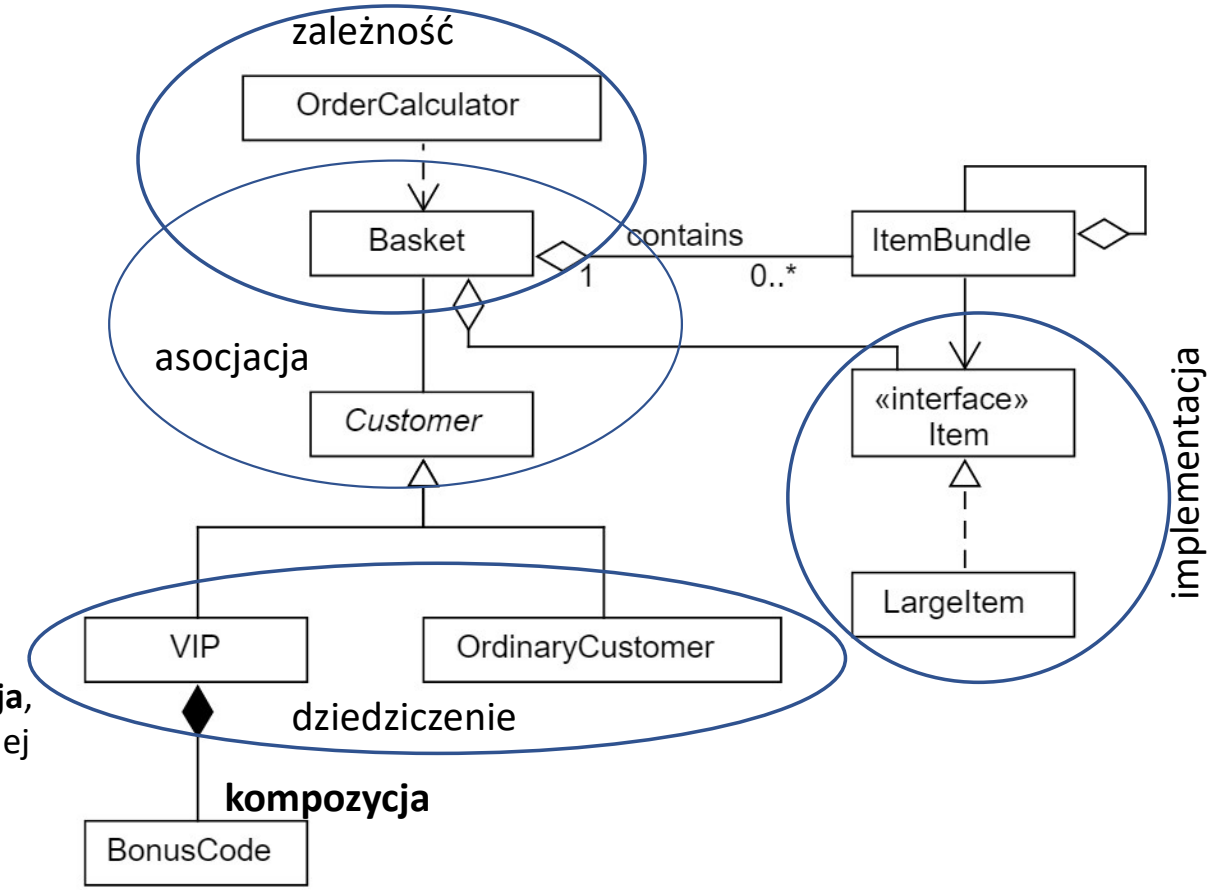
Agregacja

Wprowadza w relacji stronę, która jest „właścicielem”. Jedna klasa agreguje inną.

Kompozycja

„Właściciel” jest odpowiedzialny za tworzenie (cykl życia) elementów, które grupuje.

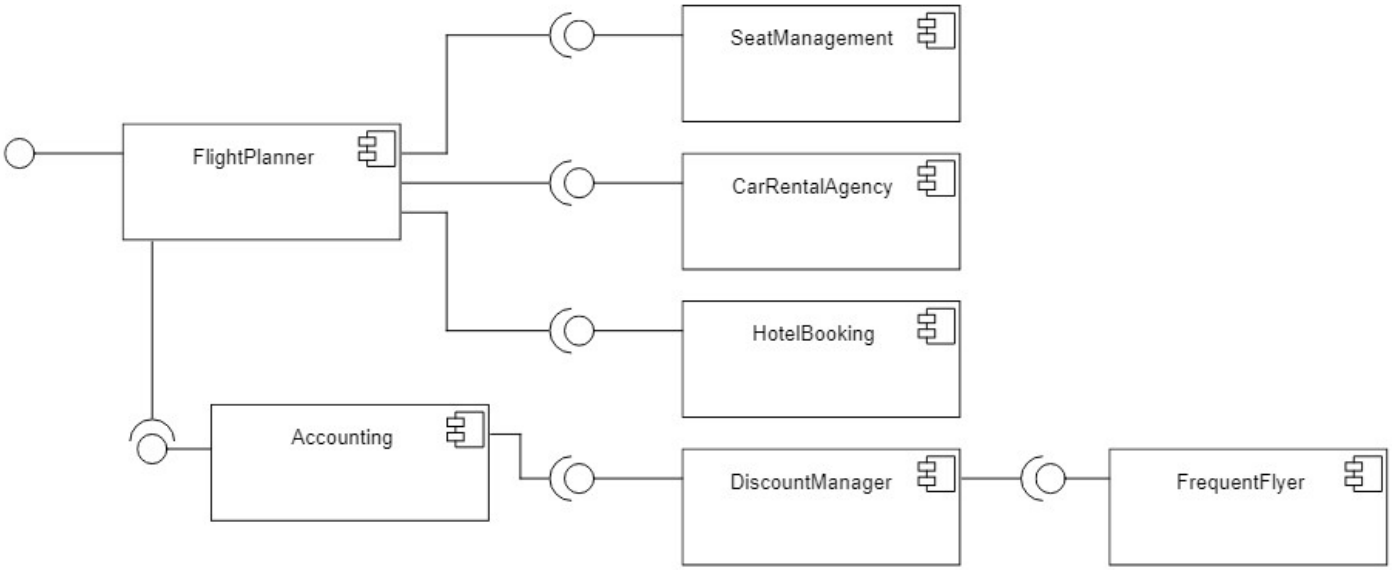
Możliwe relacje pomiędzy klasami



- Klasa *Largeitem* implementuje interfejs *Item* – **implementacja**,
- Klasy *VIP* i *Ordinarycustomer* dziedziczą po klasie abstrakcyjnej *Customer* – **dziedziczenie**,
- Klasa *Ordercalculator* używa klasy *Basket* – **zależność**,
- Klasa *Basket* wie o kliencie z którym jest powiązana (klasie *Customer*), odwrotne stwierdzenie także jest prawdziwe – **asocjacja**,
- Klasa *Basket* może zawierać wiele instancji klasy *Item* – **agregacja**,
- Klasa *VIP* zawiera wiele instancji klasy *BonusCode* i zarządza ich cyklem życia – **kompozycja**.

Strzałka oznacza kierunek relacji. Na przykład asocjacja pomiędzy *Itembundle* a *Item* jest jednokierunkowa. *Itembundle* wie o powiązanej klasie *Item*, *Item* zaś nie wie nic o *Itembundle*. Jeśli strzałka nie jest umieszczona oznacza to, że relacja jest dwukierunkowa – można „przejść” z jednej klasy do drugiej w obu kierunkach.

Diagram komponentów

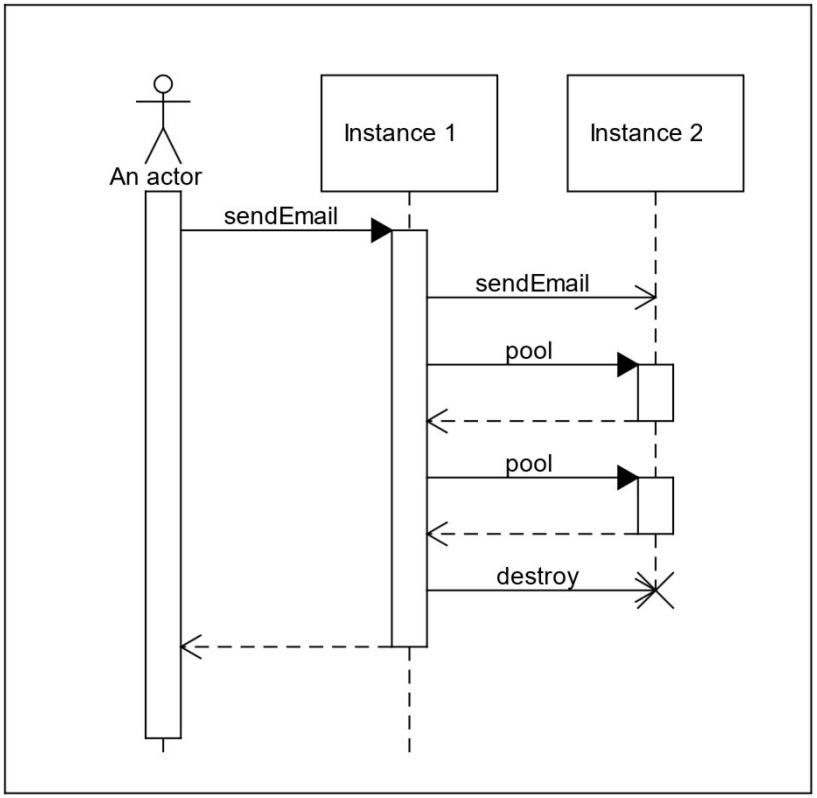


Komponent może zawierać rozbudowaną implementację w jednej klasie, poprzez ich zestaw znajdujący się w jednym pakiecie/module a na sporej części aplikacji kończąc.

Komponent na rysunku wyżej wymaga dwóch interfejsów i sam dostarcza jeden. Komponent *UserManagement* wymaga dostępu do interfejsu *persistence* a sam zapewnia dwa inne *register* i *ban*.

Component represents a modular part of a system that encapsulates its contents and whose manifestation is replaceable within its environment.

Diagram sekwencji



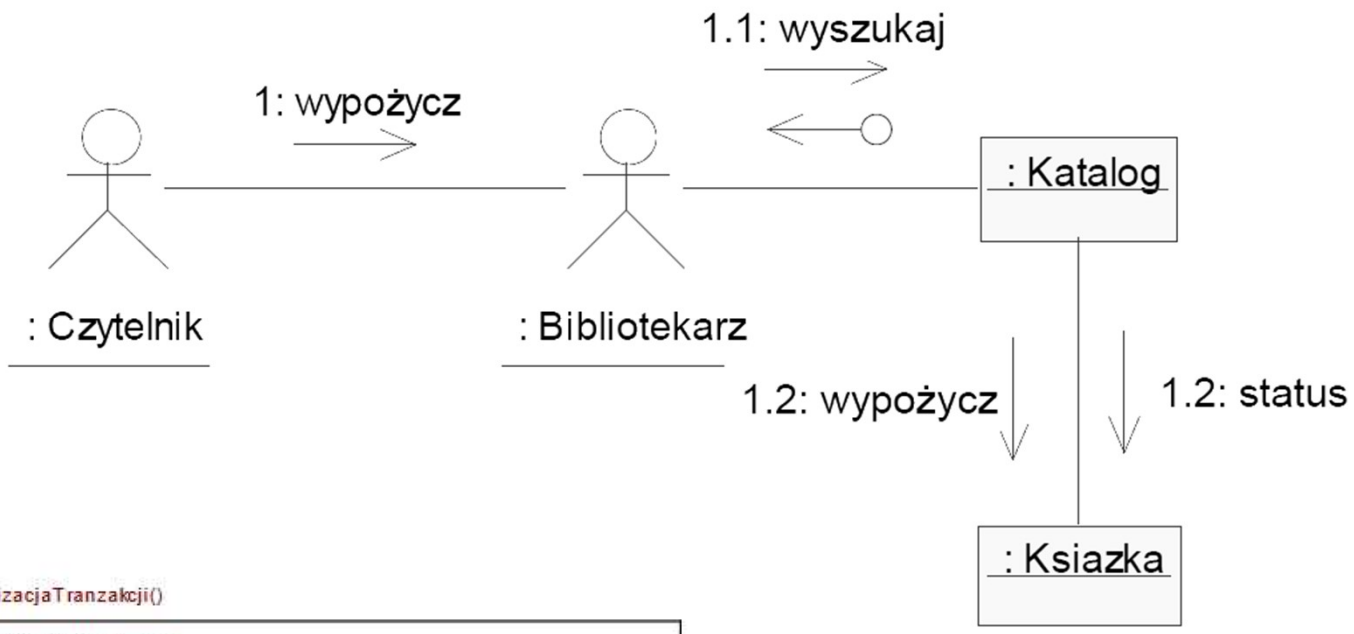
Wysyłanie komunikatu

Odpowiedź

Asynchroniczne

Wysyłanie komunikatu

Diagram komunikacji



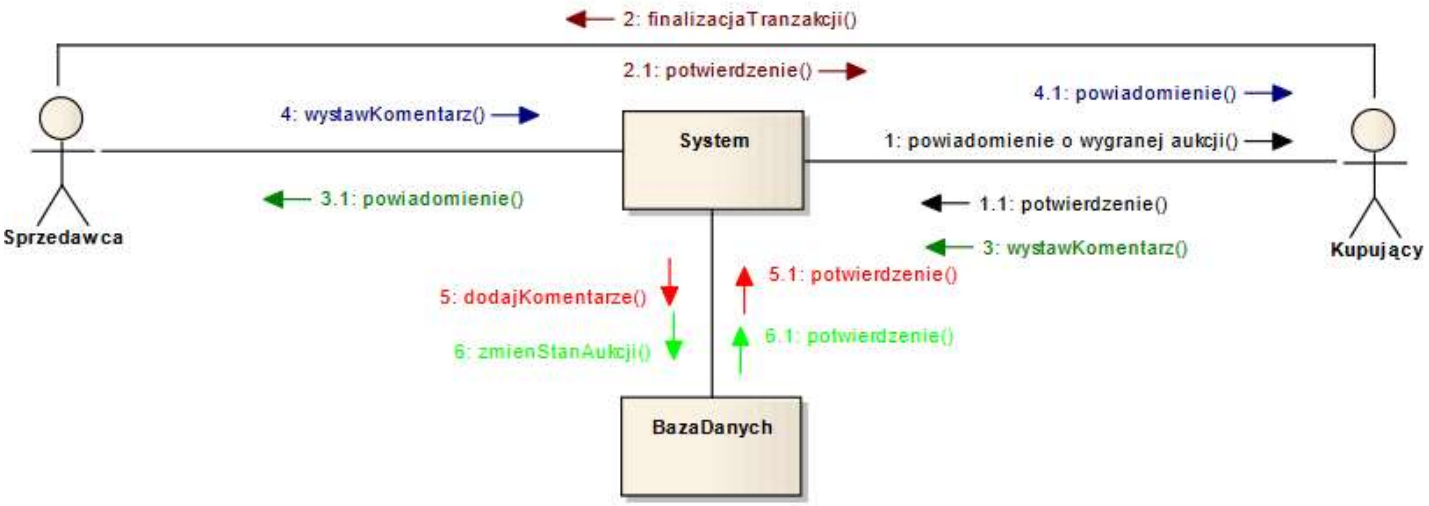


Diagram komunikacji dla przypadku użycia "Finalizacja transakcji"

Diagram przeglądowy

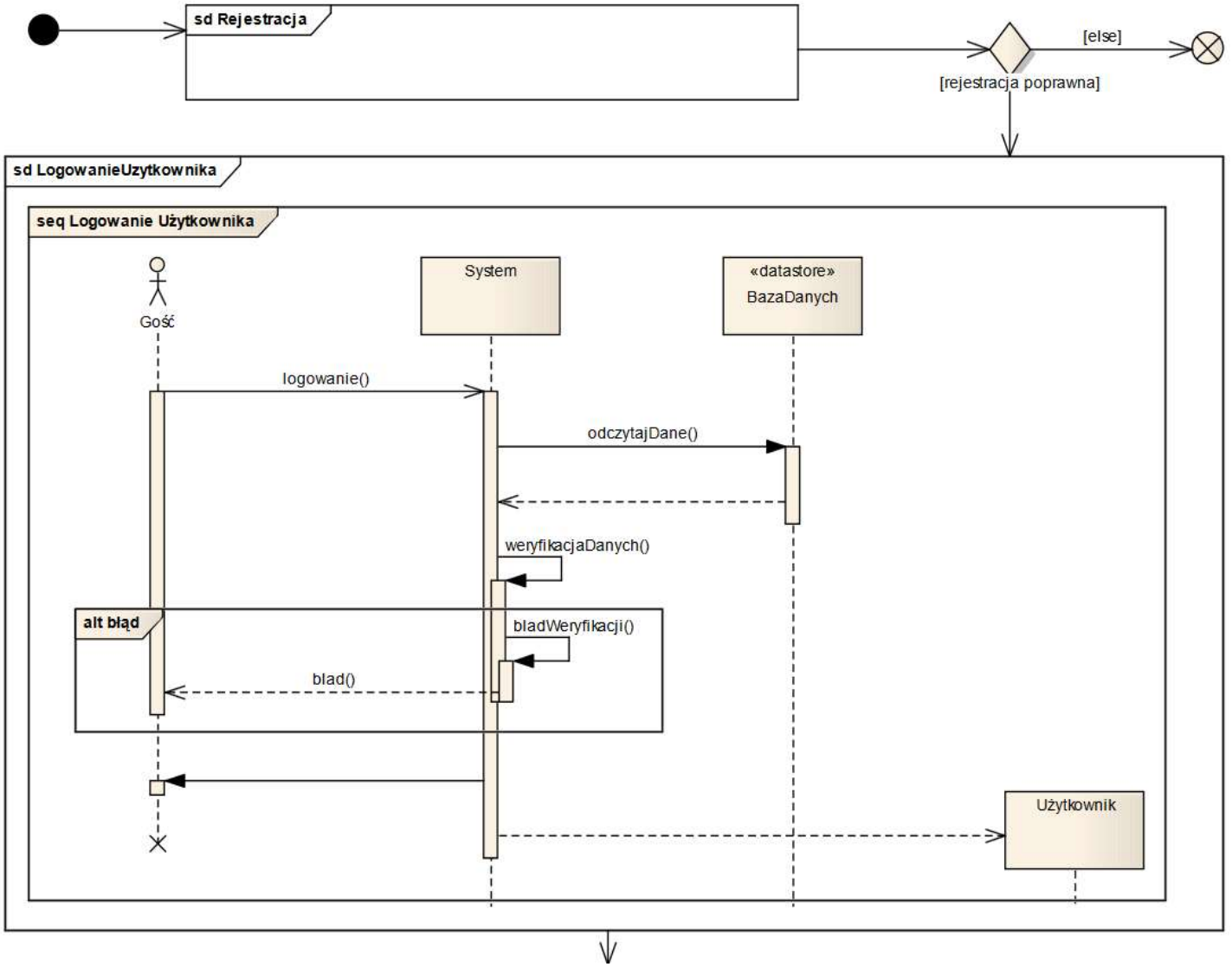


Diagram przeglądowy, cd.

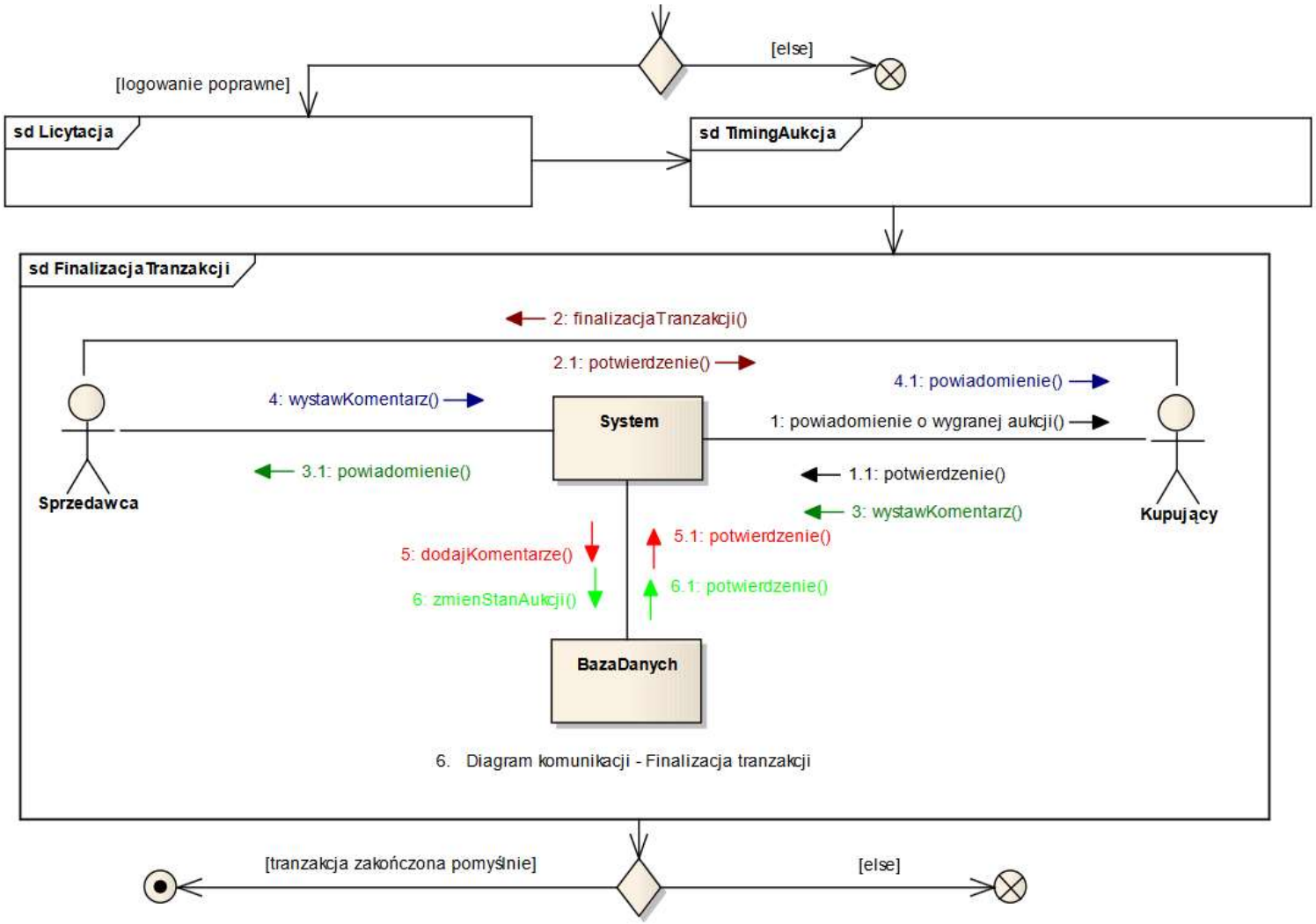


Diagram wdrożenia