UML

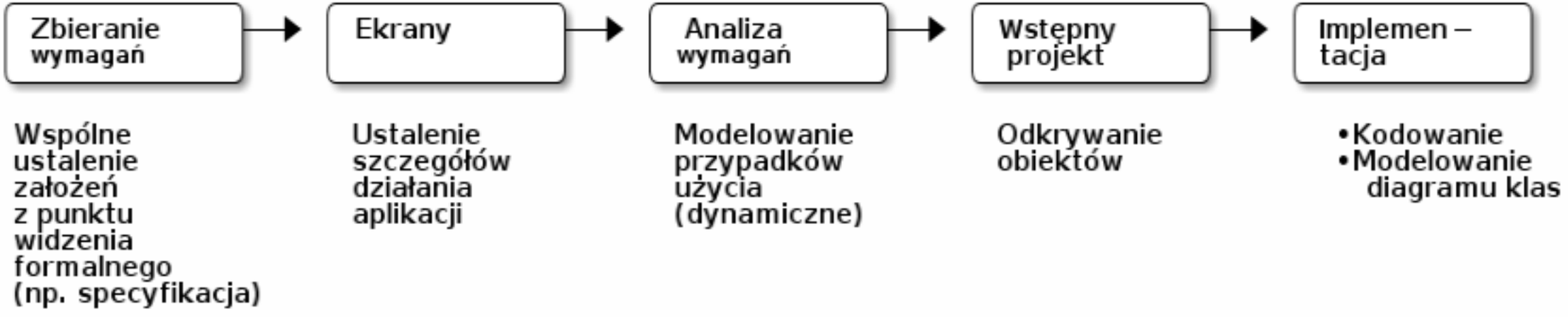
UML

- Co to jest UML
- Ujęcie procesowe
- Diagram przypadków użycia
- Modelowanie przypadków użycia
- Diagram robustness
- Diagram klas
- Modelowanie dziedziny problemu

Co to jest UML?

- *ang.* Unified Modeling Language
  - Zunifikowany język modelowania
- Wizualne przedstawianie programów obiektowych
- Może stanowić język komunikacji między klientem zamawiającym program a programistą
- Analiza problemu za pomocą obiektów i klas przedstawianych jako diagramy

Ujęcie całości procesu



Ujęcie całości procesu

Korzystamy z diagramów

- Przypadków użycia
  - Sprecyzowania oczekiwań użytkowników co do systemu
  - Wyodrębnienia wstępnego modelu dziedziny
- Klas analitycznych (roboustness)
  - Wyodrębnienia kolejnych obiektów
  - Uszczegółowienia istniejącego modelu
  - Opisania zachowania systemu w interakcji z użytkownikiem
- Klas
  - Reprezentowania aktualnej wiedzy o systemie

Diagram przypadków użycia

Diagram przypadków użycia

- To ogólny opis funkcjonalności systemu
- Przedstawia "co" system ma robić a nie "jak" to ma robić
- Modeluje usługi, które system wystawia na zewnątrz
- Jest identyfikacją i dokumentacją wymagań systemu
- Jest platformą współpracy i komunikacji między twórcami oprogramowania a klientem

Elementy

Przypadek użycia

- ma unikalną nazwę opisującą czynność
- jest specyfikacją zbioru akcji, które system może wykonać
- przedstawia jedną funkcjonalność systemu

Elementy

Aktor

- ma unikalną nazwę w formie rzeczownika
- to osoba fizyczna, organizacja lub system zewnętrzny
- wchodzi w interakcję z systemem
- użytkuje jeden lub więcej przypadków użycia

Elementy

Interakcja

- modeluje oddziaływanie aktora na przypadek użycia
- wskazuje na kierunek przepływu informacji
- ma formę prostej linii

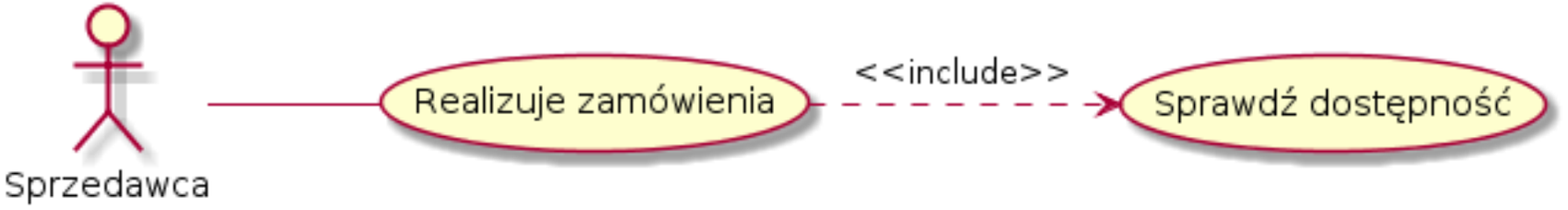
Elementy

Relacja <<*extend*>>

- *bazowy przypadek użycia* **może** być rozszerzony przez dodatkowe operacje (*rozszerzający przypadek użycia*)
- składa się z warunku i referencji do punktu rozszerzenia
- stosowana do modelowania obsługi błędów i obsługi systemu menu

Elementy

Relacje <<*include*>>



- *bazowy przypadek użycia* **zawsze** zawiera (używa) *zawierany przypadek użycia*
- bazowy przypadek użycia "wie" kiedy i dlaczego zawierany przypadek użycia ma być zrealizowany
- stosowana do współdzielenia operacji przez wiele przypadków użycia

Elementy

Generalizacja

- Przypadki użycia - przypadek potomny dziedziczy zbiór akcji, punkty rozszerzenia i atrybuty przypadku macierzystego

- Aktorzy - wprowadza hierarchię - aktorzy wyspecjalizowani mają dodatkowe kompetencje do interakcji z systemem

Opis tekstowy

Opis przypadku użycia (jego specyfikacja)

- Nazwę przypadku użycia
- Krótki opis
- Wymienionych aktorów
- Warunki wejściowe (ang. pre-condidtions)
- Podstawowy przepływ zdarzeń
- Alternatywny przepływ zdarzeń
- Warunki końcowe (ang. post-conditions)

Przykład

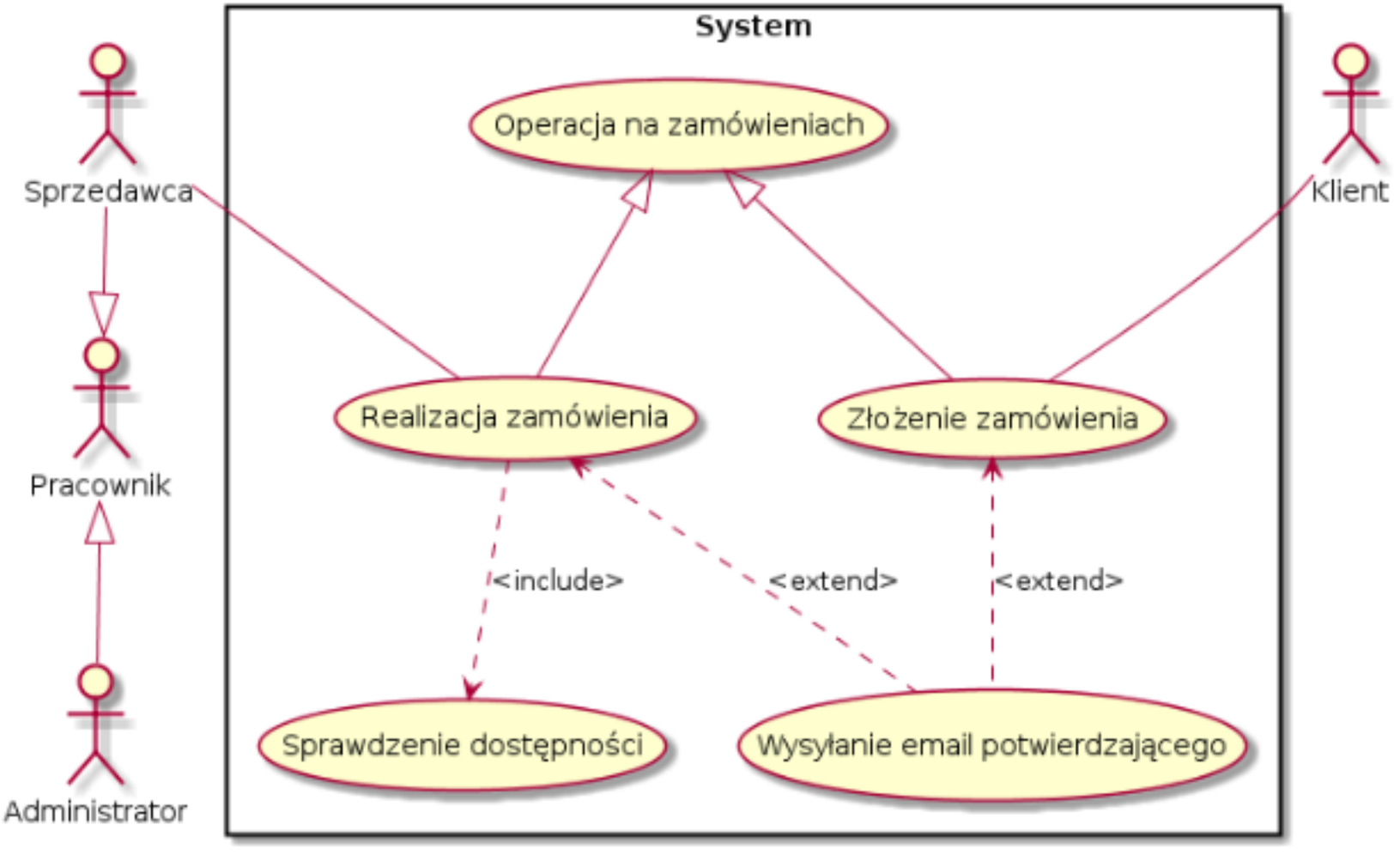

System
Operacja na zamówieniach
Sprzedawca
Klient
Pracownik
Realizacja zamówienia
Złożenie zamówienia
<include>
<extend>
<extend>
Administrator
Sprawdzenie dostępności
Wysyłanie email potwierdzającego

Diagram klas

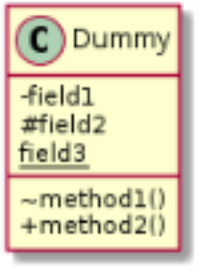
Elementy

Klasa

- Symbol klasy podzielony jest na trzy sekcje - każda dla innego typu inwariantów: nazwy, atrybutów i metod klasy.
- W razie potrzeby można dodać nowe sekcje np. dla wyjątków

Elementy

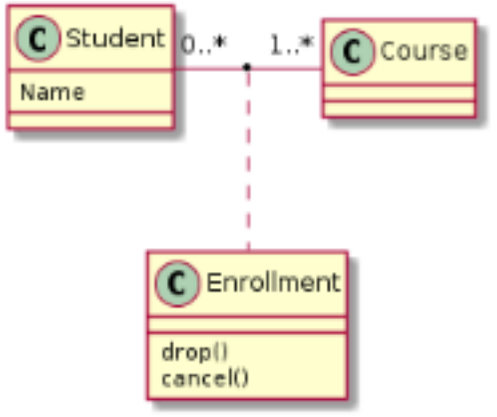
Poziomy dostępu



- Poziomy dostępu określają widoczność atrybutu / metody z innych klas:
  - \- prywatny
  - ~ zakres pakietu
  - # chroniony
  - \+ publiczny
- Atrybut lub metoda podkreślona oznacza statyczność - przynależność do klasy a nie do obiektu.

Elementy

Krotność



- Krotność – określa minimalną i maksymalną liczbę obiektów, jakie można powiązać z daną cechą
  - `1` – dokładnie jeden obiekt
  - `0…1` – 0 lub maksymalnie 1 obiekt
  - `1…*` – przynajmniej 1 obiekt
  - `1, 2, 5` – konkretne liczby obiektów
  - `0…*` – dowolna liczba obiektów oznaczana też jako `*`

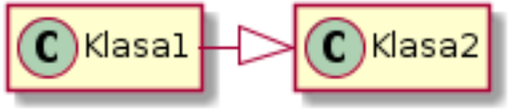
Elementy

Związki pomiędzy klasami

	**Zależność** - najsłabszy związek między klasami, gdy jedna z nich używa innych klas.
	**Asocjacja** - Asocjacja wskazuje na trwałe powiązanie pomiędzy obiektami danych klas. Asocjacja występuje, gdy instancje mają “swoje życie” i żadna z klas asocjacyjnych nie ma właściciela pomiędzy sobą.
	**Agregacja** - związek typu całość-część. Występuje tutaj relacja posiadania — co oznacza, że elementy częściowe mogą należeć do większej całości, jednak również mogą istnieć bez niej
	**Kompozycja** - zwana również złożeniem, jest związkiem typu całość-część. W relacji kompozycji, części należą tylko do jednej całości, a ich okres życia jest wspólny — razem z całością niszczone są również części.

Elementy

Dziedziczenie



Dziedziczenie – tworzy związek klasa – podklasa. Jest hierarchią klas od ogólnych do bardziej szczegółowych. Pozwala wyodrębnić części wspólne klas.

Podsumowanie

- Diagram klas przedstawia statyczną strukturę systemu
- Pokazuje klasy systemu wraz z ich atrybutami i metodami
- Modeluje związki między klasami
- Za jego pomocą przedstawia się dziedzinę systemu – zestaw bytów ze świata rzeczywistego, które wspiera projektowany system

Przykład

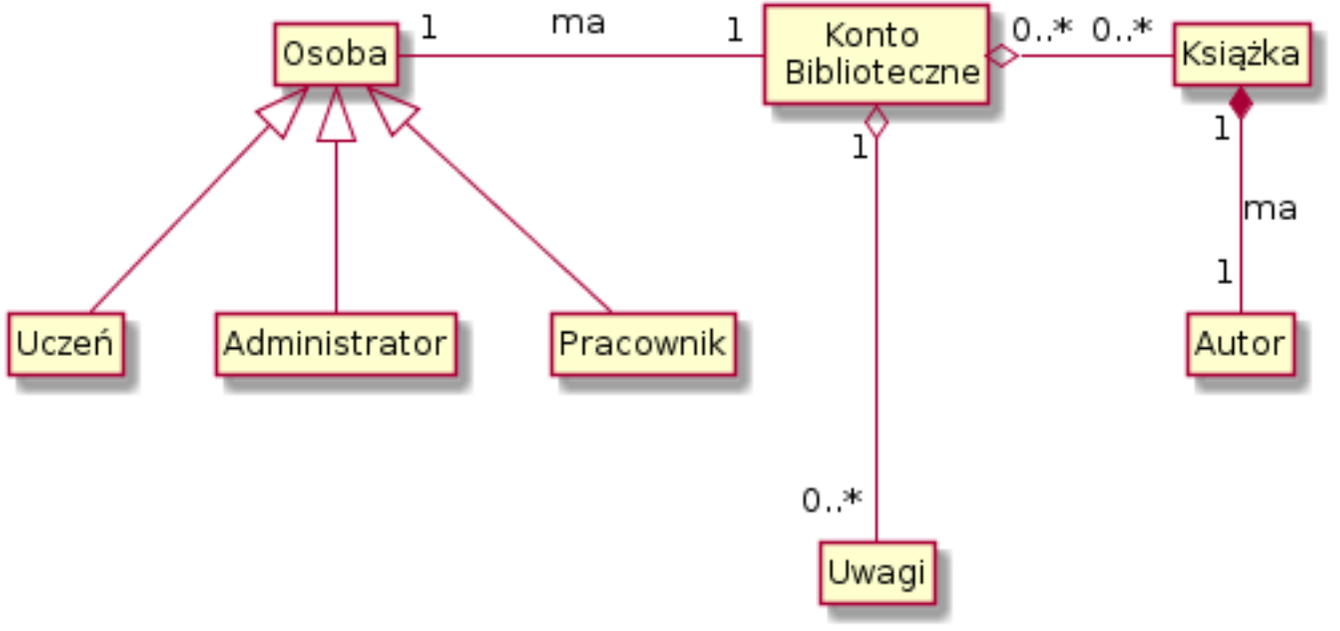

Osoba
1
ma
1
Konto Biblioteczne
0..*
0..*
Książka
1
ma
1
Autor
1
0..*
Uwagi
Uczeń
Administrator
Pracownik

Diagram sekwencji

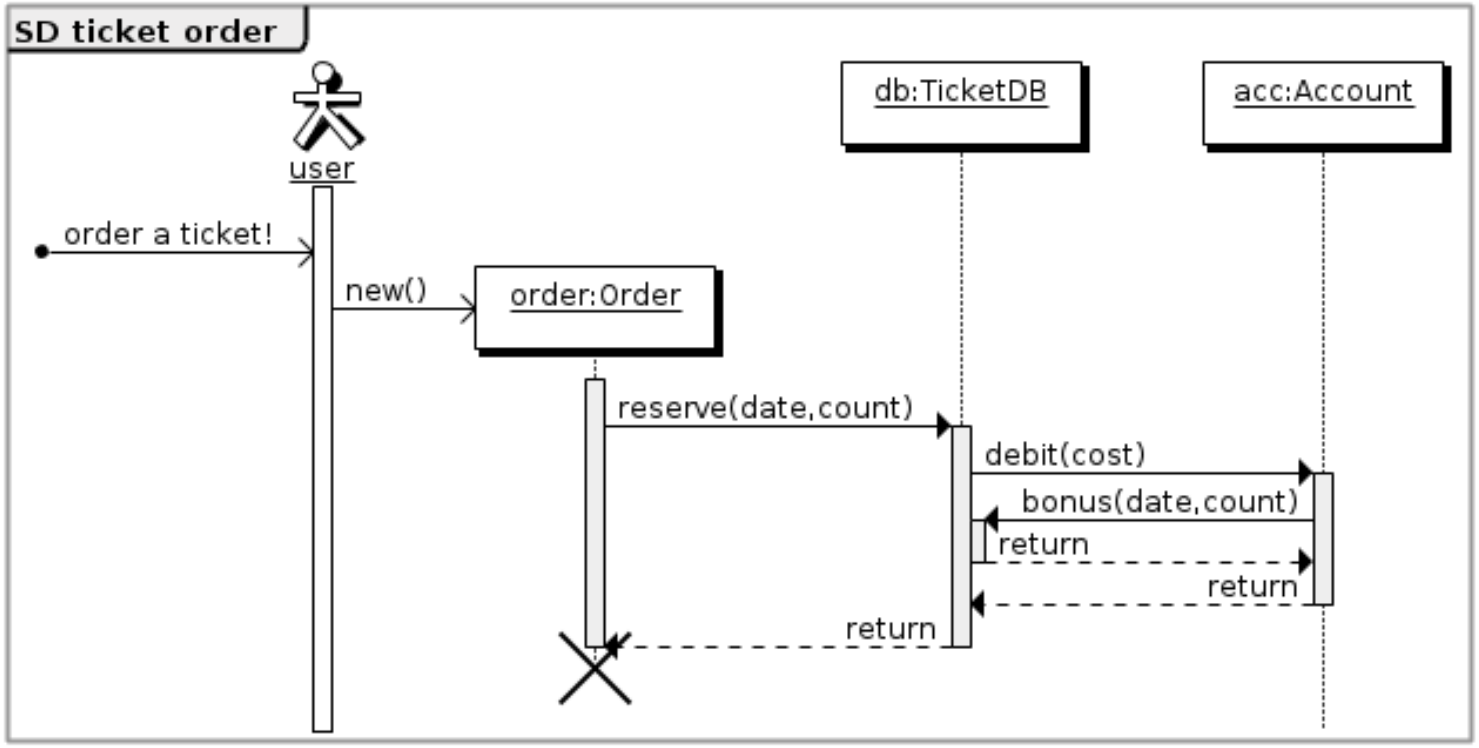


Diagram sekwencji

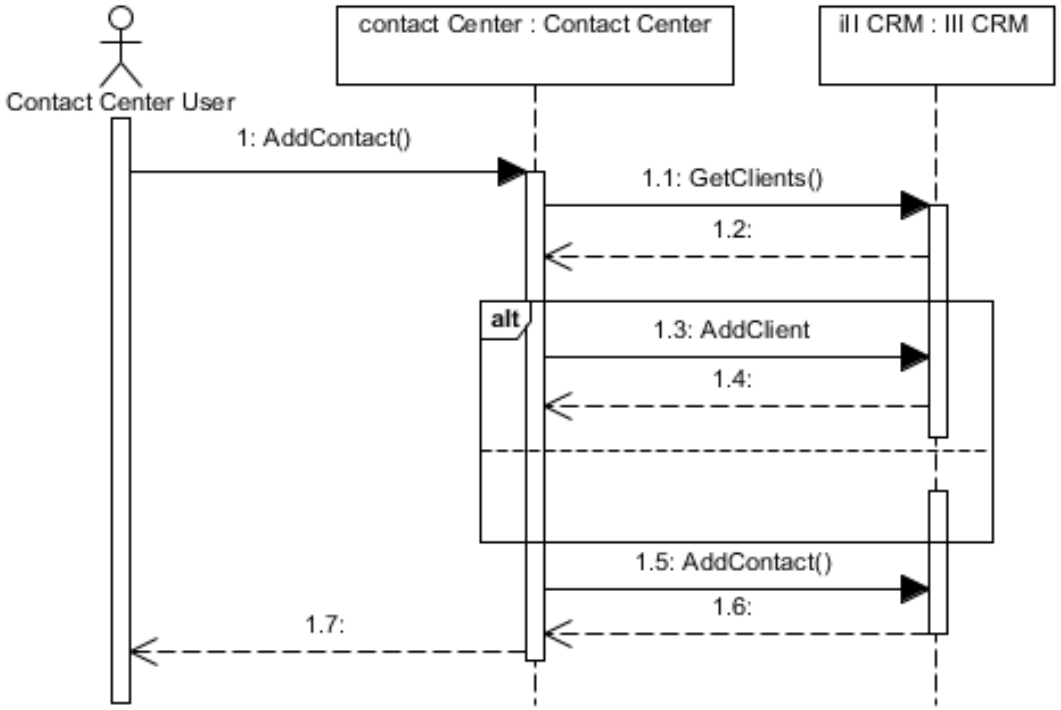


Diagram sekwencji

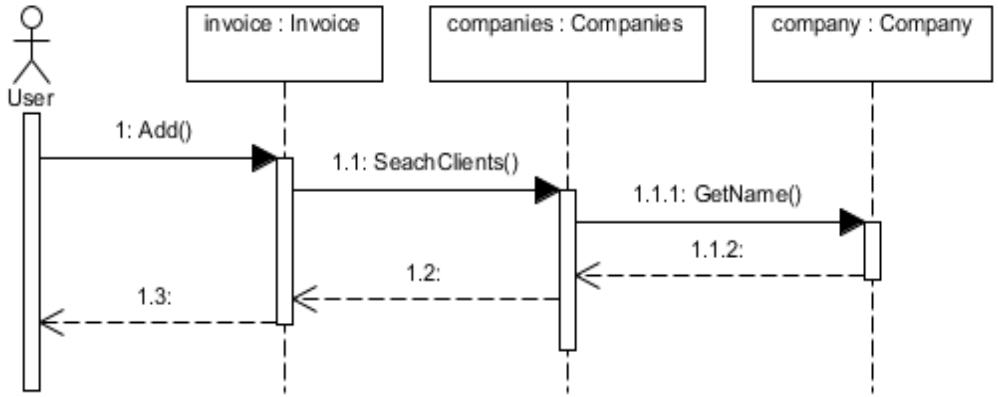
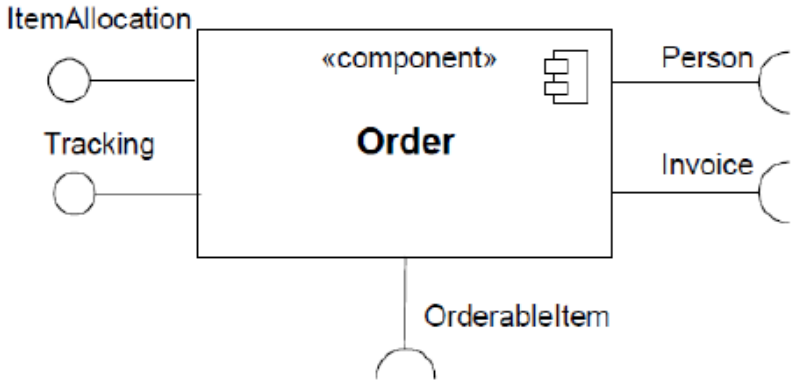


Diagram komponentów



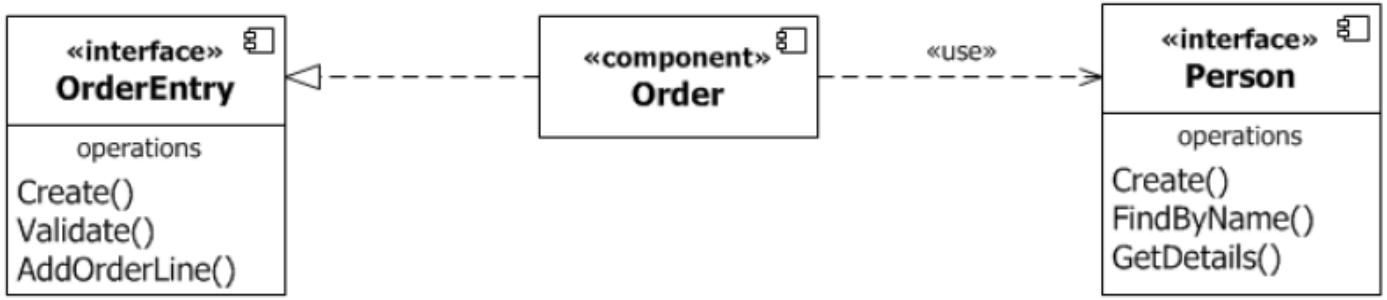


Diagram komponentów

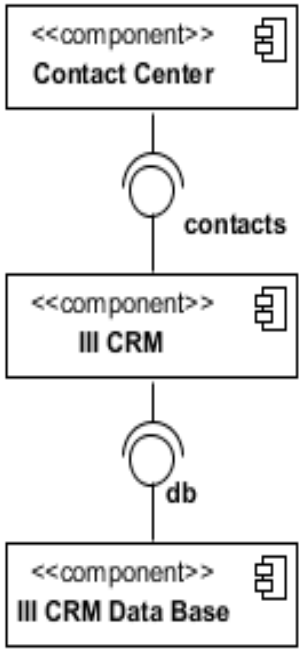
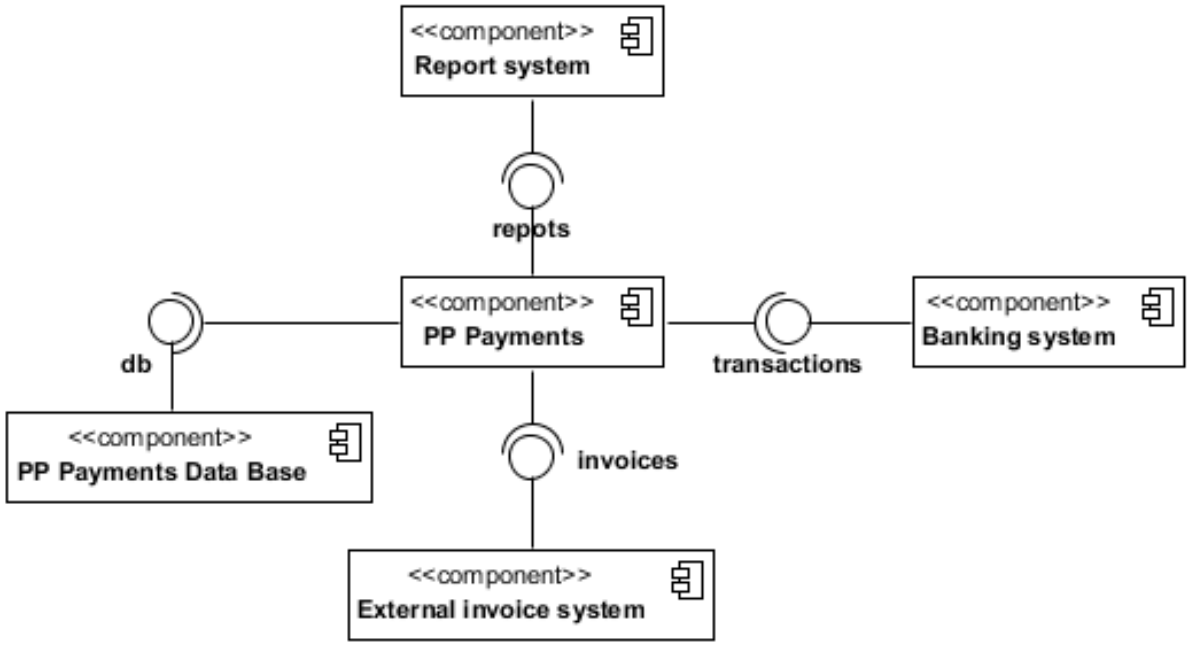


Diagram komponentów



Który diagram wybrać?