Tworzenie modelu konceptualnego systemu informatycznego – część 1

1. **Elementy diagramów przypadków użycia (use-cases)**
2. **Wytyczne tworzenia diagramów przypadków użycia (use-cases)**

   (wg Booch G., Rumbaugh J., Jacobson I., UML przewodnik użytkownika)
3. **Model konceptualny – identyfikacja i specyfikacja wymagań**
4. **Przykłady diagramów przypadków użycia (use-cases)**

Tworzenie modelu konceptualnego systemu informatycznego – część 1

1. **Elementy diagramów przypadków użycia (use-cases)**

Diagramy UML 2 – część pierwsza

Na podstawie

UML 2.0 Tutorial

http://sparxsystems.com.au/resources/uml2_tutorial/

Dwa rodzaje diagramów UML 2

Diagramy UML <u>modelowania strukturalnego</u>

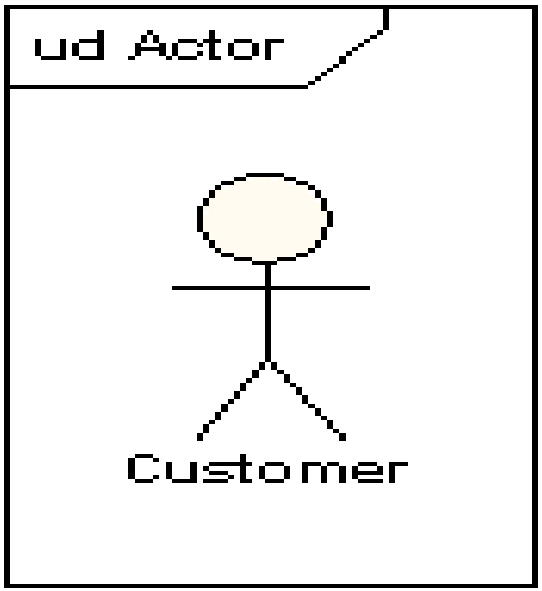
- ***Diagramy pakietów***
- ***Diagramy klas***
- **Diagramy obiektów**
- **Diagramy mieszane**
- **Diagramy komponentów**
- **Diagramy wdrożenia**

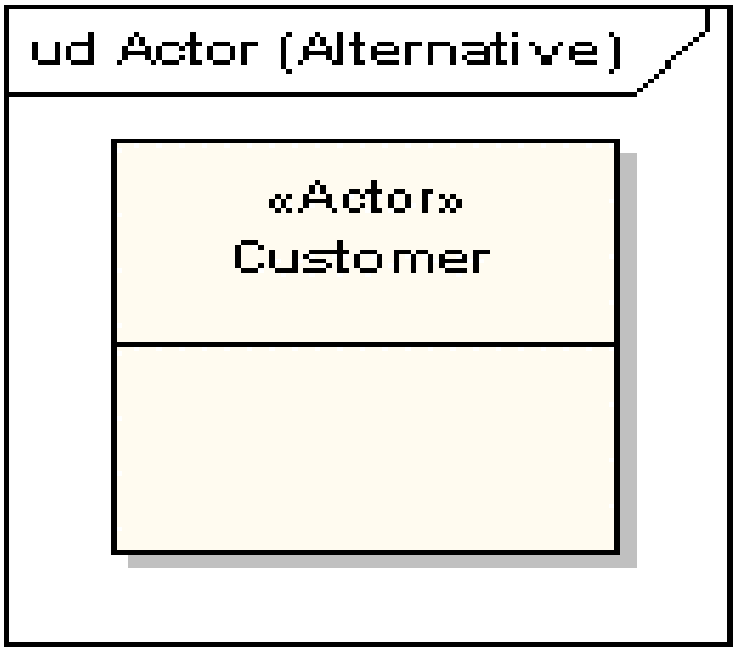
Diagramy UML <u>modelowania zachowania</u>

- ***Diagramy przypadków użycia***
- **Diagramy aktywności**
- **Diagramy stanów**
- **Diagramy komunikacji**
- ***Diagramy sekwencji***
- **Diagramy czasu**
- **Diagramy interakcji**

Diagramy przypadków użycia (Use Case Diagram)

- **Diagramy przypadków użycia** opisują wymagania systemu
- **Przypadki użycia (Use cases)** oznaczają funkcje udostępniane użytkowników lub innych zewnętrznych systemów **(actors)** przez projektowany system

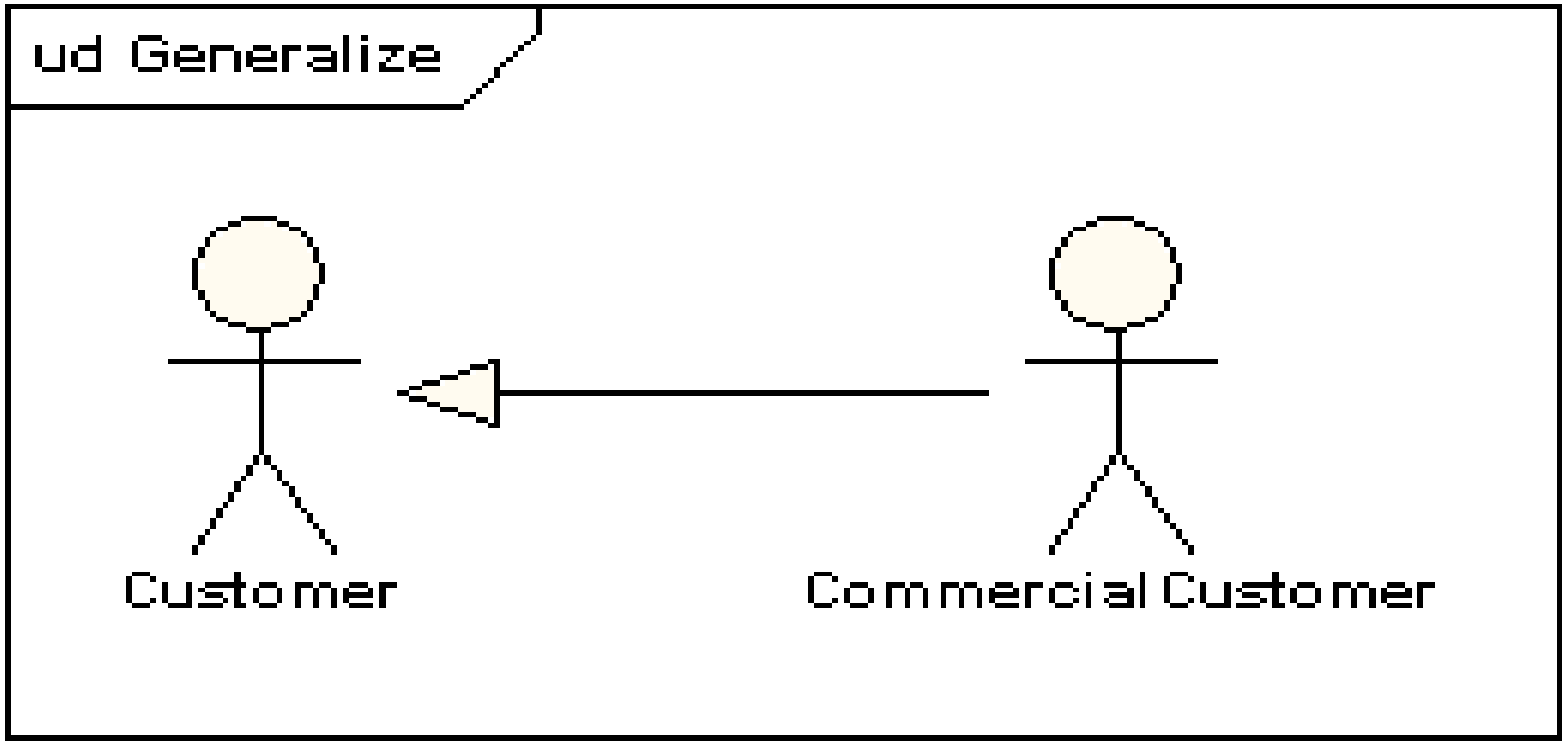




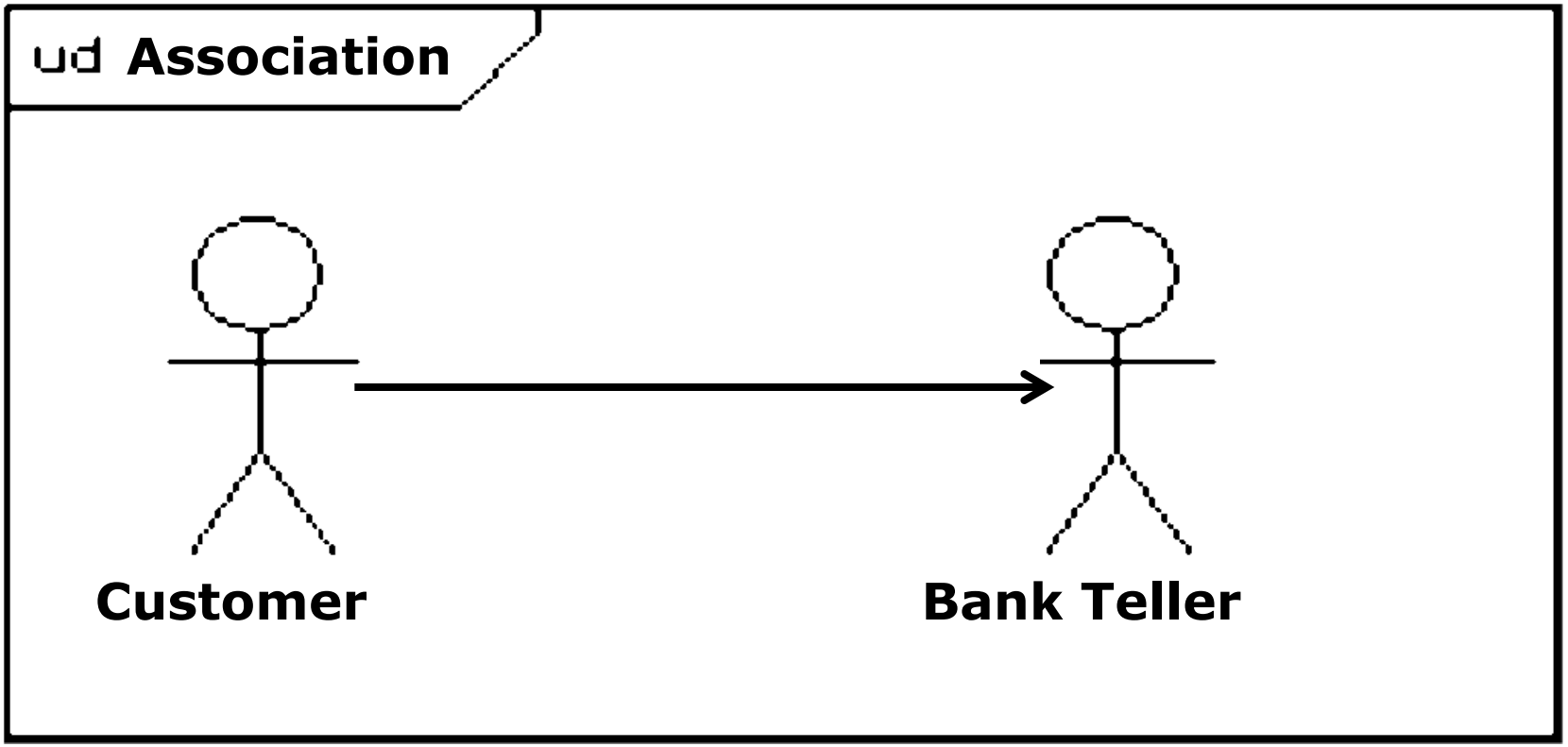
Actors

Zewnętrzni użytkownicy: ludzie, sprzęt, system

rysowania jako figura lub klasa ze słowem kluczowym **«actor»**



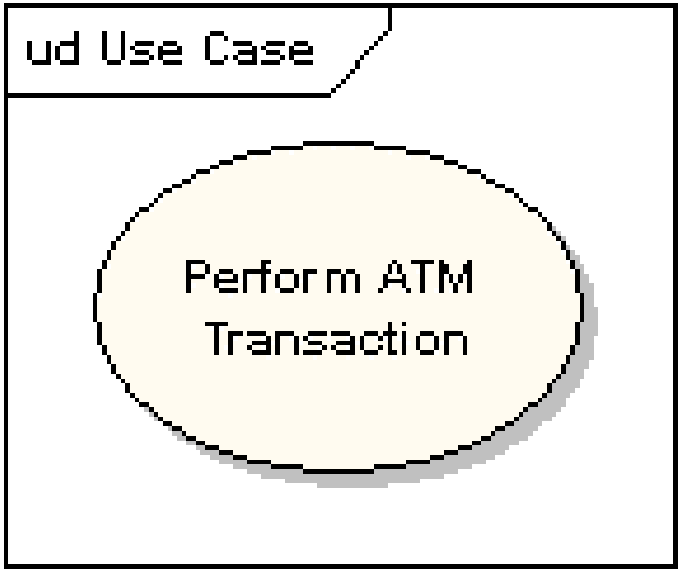
Związek miedzy aktorami typu Generalization
Actors mogą genaralizować innych **Actors.**
Oznacza to dziedziczenie funkcji (przypadków użycia) przez aktora **CommercialCustomer** od aktora **Customer** oraz korzystanie z nowych przypadków użycia

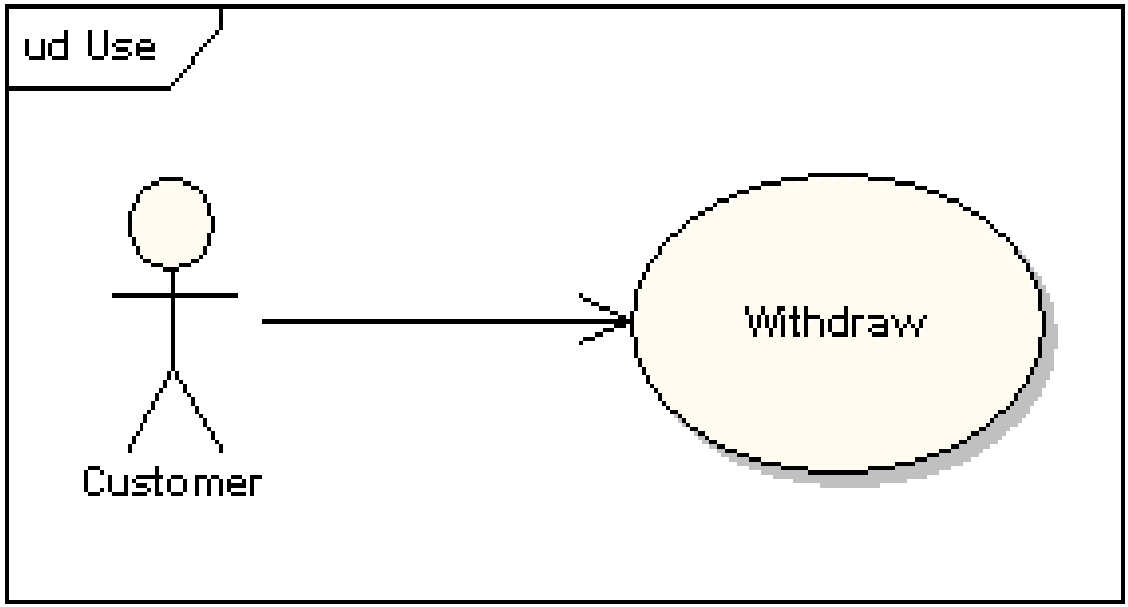


Związek miedzy aktorami typu Association

Actors mogą być powiązani z innymi **Actors,** czyli mogą pośredniczyć w dostępie do przypadków użycia.

Oznacza to, że aktor **Customer** za pośrednictwem aktora **BankTeller** korzysta z jego przypadków użycia





Przypadek użycia (Use Cases)

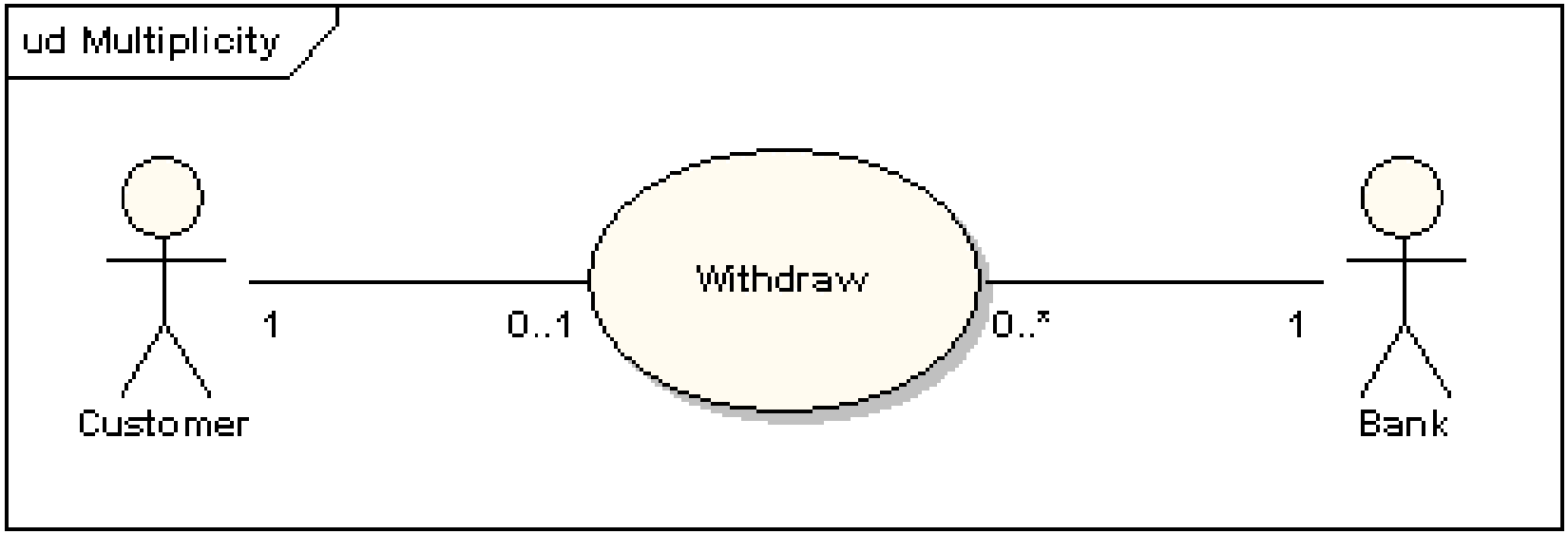
- Jednostka pracy
- Wysoki poziomom zewnętrznej obserwacji systemu
- Notacja – elipsa
- <<>> znak stereotypu oznaczającego właściwości związku

Związek użycia przypadku użycia <<use>>

•Np. aktor *Customer* używa przypadku użycia *Withdraw* (pobiera pieniądze np. z konta)

Przypadek użycia zawiera:

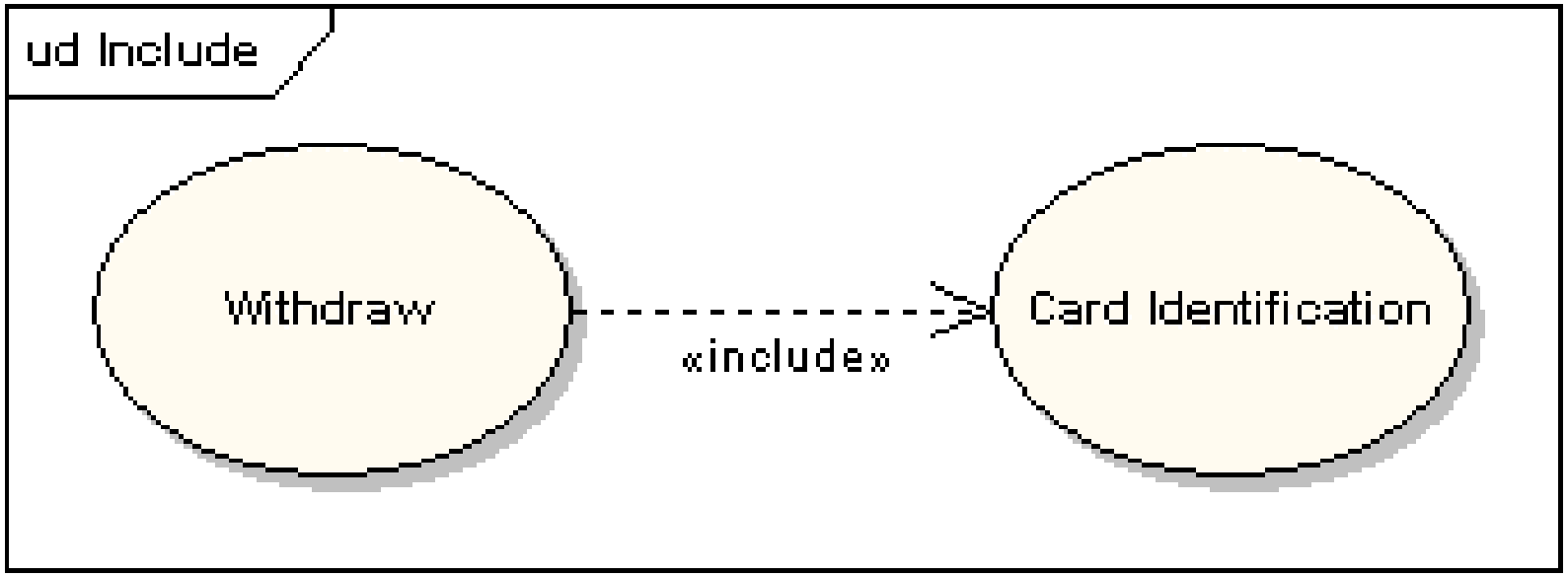
- **Nazwę i opis**
- **Wymagania** funkcjonalne spełniane dla użytkownika
- **Ograniczenia** – warunki przed- po- przypadku użycia oraz nie zmieniające się na skutek wykonania przypadku użycia
- **Scenariusze** – sekwencja zdarzeń między systemem i zewnętrznymi użytkownikami (opis tekstowy)
- **Diagramy scenariuszy** – diagramy sekwencji
- **Dodatkowe informacje** – np. identyfikacja karty płatniczej przed dokonaniem wyciągu z konta



Powiązania (Association) – liczność związku

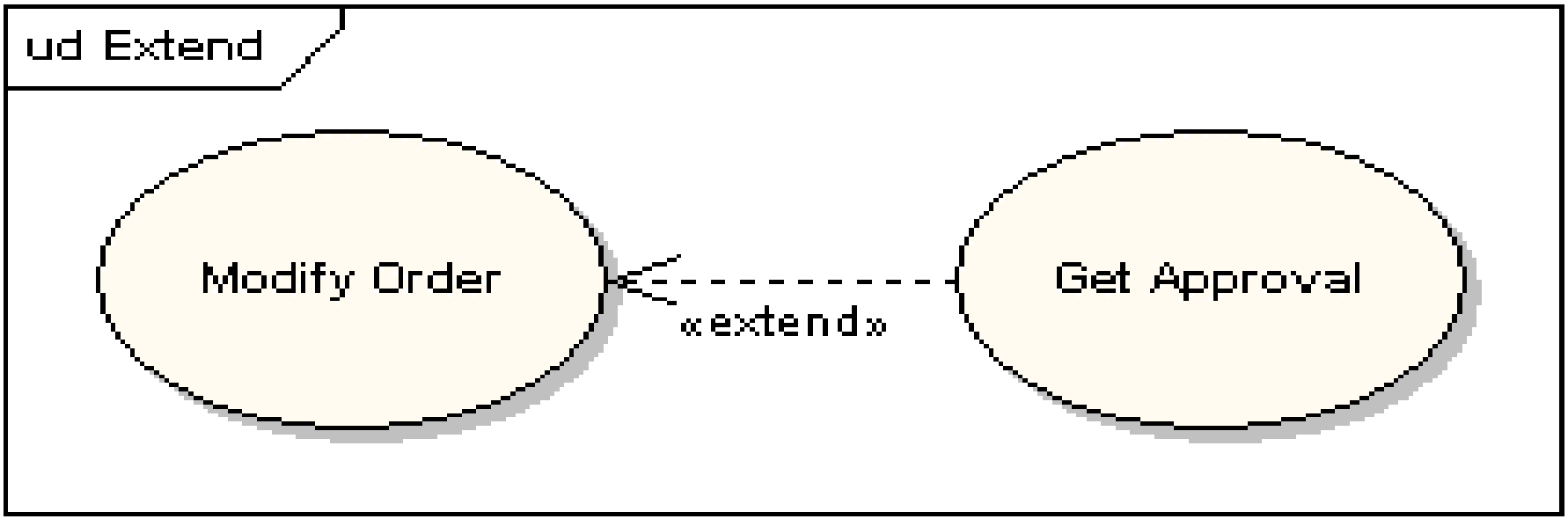
- Liczność instancji na końcu połączenia

Np. aktor *Klient (Customer)* ma tylko jedną sesję wypłacania pieniędzy w danym momencie (*Withdraw*) natomiast *Bank* może mieć ich wiele w tym samym czasie



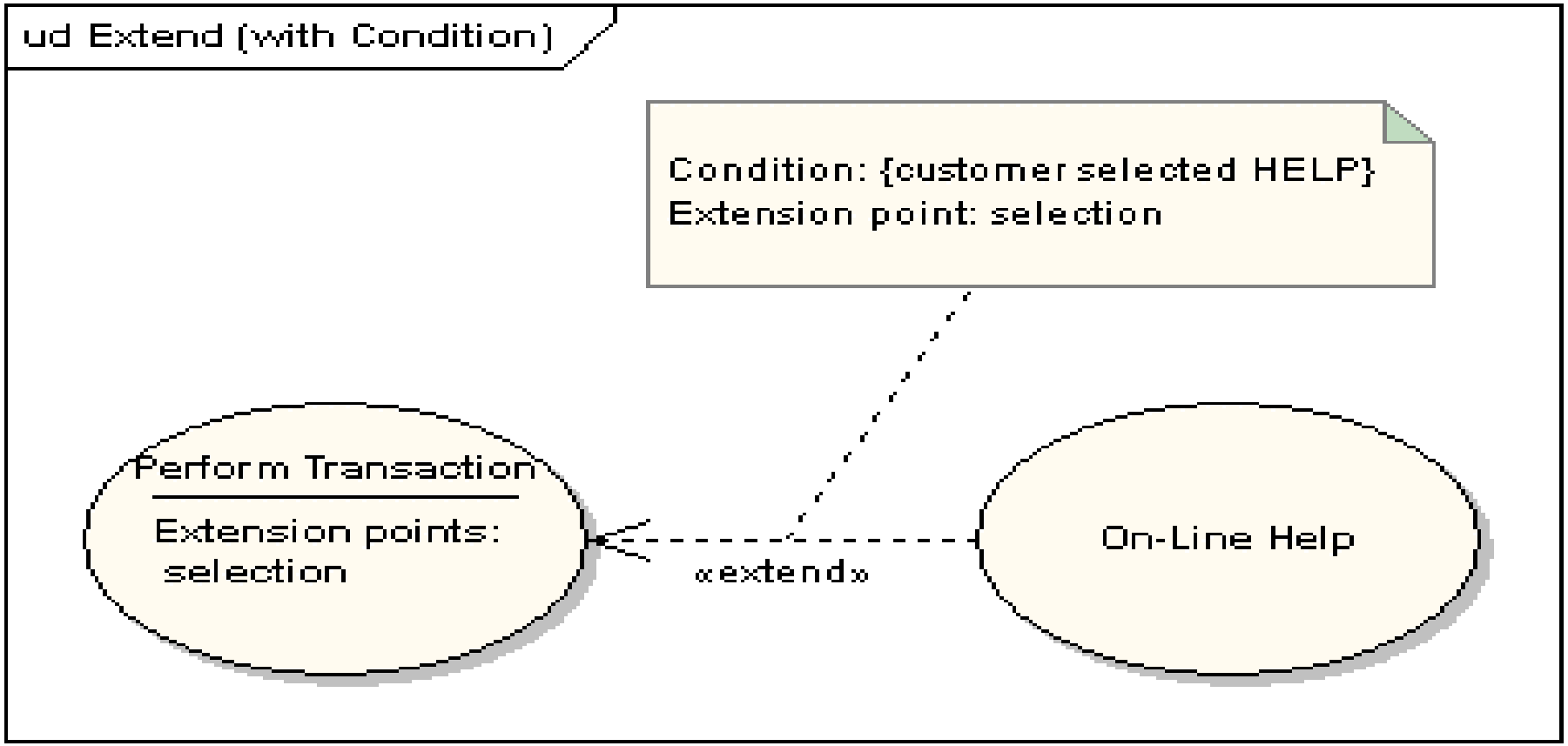
Zawieranie <<includes>>

Przypadek użycia zawiera jeden lub wiele innych przypadków użycia eliminując powtarzanie funkcjonalności systemu dzięki tej wielożywalności, czyli zawieraniu
np.Pobranie z konta (*Withdraw*) zawsze zawiera identyfikację karty (*Card identification*)



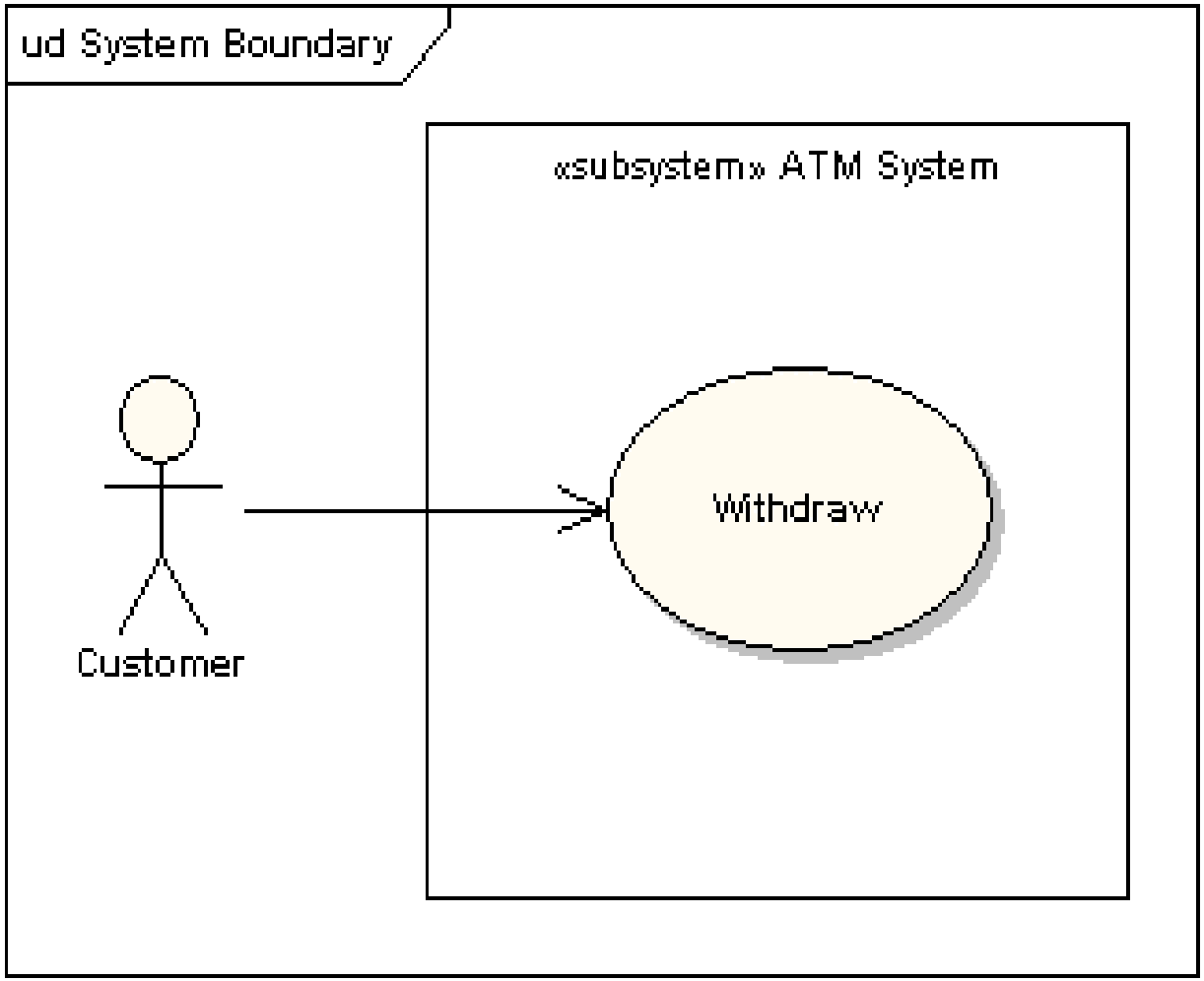
Rozszerzanie <<extends>>

Jeden przypadek użycia może być użyty do rozszerzenia właściwości drugiego przypadku użycia
Np. Przypadek użycia *Zezwolenie (Get Approval)* opcjonalnie rozszerza właściwości przypadku użycia *Modyfikuj zlecenie (Modify Order)*



Punkty rozszerzające (Extension Points)

np..Punkt, w którym rozszerzany przypadek użycia *Wykonanie tranzakcji* (*Perform Transaction*) jest rozszerzany przez rozszerzajacy przypadek użycia *Pomoc* (*On-Line Help*) zgodnie ze znaczeniem punktu rozszerzania np. przez *wybór* (*selection*)



Granice systemu (System Boundary)

Zazwyczaj aktorzy są na zewnątrz systemu, a przypadki użycia wewnątrz systemu.

Tworzenie modelu konceptualnego systemu informatycznego – część 1

1. **Elementy diagramów przypadków użycia (use-cases)**
2. **Wytyczne tworzenia diagramów przypadków użycia (use cases)**

(wg Booch G., Rumbaugh J., Jacobson I., UML przewodnik użytkownika)

Identyfikacja aktorów i przypadków użycia – przypadek prostego systemu

1. Należy wyznaczyć granice systemu w ramach środowiska
2. Należy zdefiniować wymagania systemu: należy podać użytkowników - aktorów systemu, zależności między aktorami typu dziedziczenie (***generalization***) lub powiązanie (***association***), oczekiwane funkcje systemu (przypadki użycia), powiązania między aktorami i przypadkami użycia oraz zależności między przypadkami użycia
3. Należy opisać każdy przypadek użycia np. wg szablonu podanego na slajdzie 9 (przykład na slajdzie 28)

(1) Wytyczne przy modelowaniu granic systemu

- Należy zidentyfikować aktorów działających wokół systemu. Oznacza to wyznaczenie grup użytkowników korzystających w określonym celu z projektowanego systemu (zarządzanie, pielęgnacja, usługi) - (analiza wspólności i zmienności)
- Należy uporządkować aktorów wg zależności typu powiązanie np. Klient korzysta z usług Sprzedawcy lub dziedziczenia: Komercyjny Klient dziedziczy przypadki użycia od Klienta (analiza wspólności i zmienności)
- Należy powiązać aktorów z przypadkami użycia za pomocą powiązań nadając im zidentyfikowane znaczenie za pomocą stereotypu wg podanych definicji

(2) Wytyczne przy modelowaniu wymagań stawianych systemowi (slajdy 21,28)

- Należy określić otoczenie systemu, czyli zidentyfikować aktorów
- Dla każdego aktora należy podać działania, jakie każdy aktor oczekuje od systemu (slajdy 22, 27, 30)
- Działania należy zapisać jako przypadki użycia
- Należy wyłączyć powtarzające się ciągi działań i zastąpić je jednym nowym połączonym relacją **<<include>>** lub **<<extends>>** lub **<<use>>** lub zwykłym powiązaniem typu ***association*** bez stereotypu (analiza wspólności i zmienności)
- Należy uwzględnić tych aktorów, przypadki użycia oraz zidentyfikowane powiązania między nimi
- Można dodać do każdego aktora i przypadku użycia notatkę opisującą wymagania niefunkcjonalne (np. sprzęt, język, korzystanie z Internetu)

(3) Wytyczne przy modelowaniu przypadków użycia

- Należy opisać główny i nadzwyczajne ciągi zdarzeń każdego przypadku użycia (slajd 28) podając: czynności i dane używane podczas działania przypadku użycia
- Należy zdefiniować testy systemu (slajd 29) w odniesieniu do wybranego aktora i powiązanego za nim jednego lub grupy przypadków użycia podając stan początkowy i końcowy oznaczający powodzenie testu przypadku użycia (np. *Wstawianie nowej książki jest możliwe tylko wtedy, gdy istnieje już jej tytuł w katalogu oraz posiada unikatowy numer. Po wstawieniu tej książki nie może być dwóch książek o tym samym numerze*)

Tworzenie modelu konceptualnego systemu informatycznego – część 1

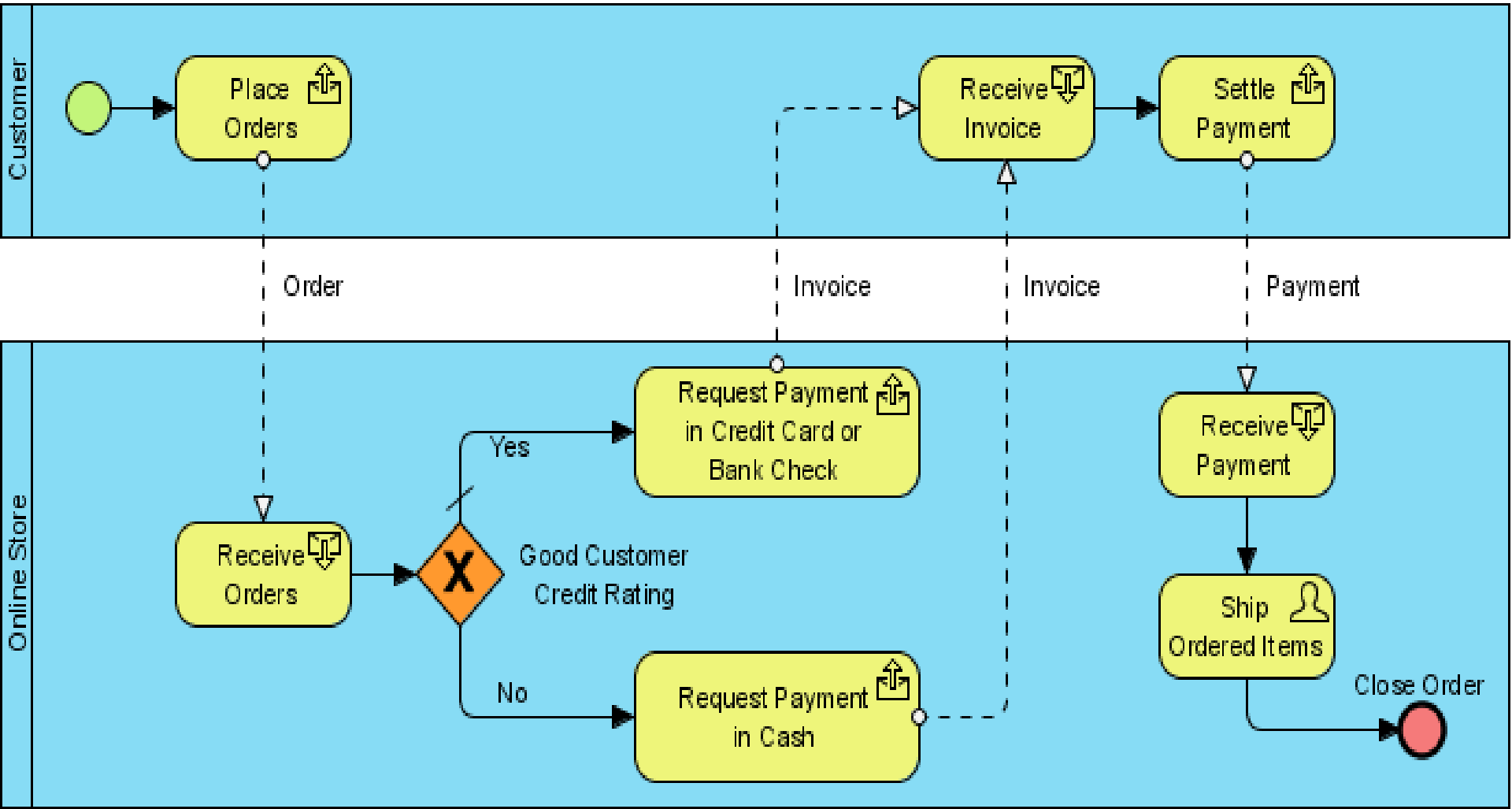
1. **Elementy diagramów diagramów przypadków użycia (use-cases)**
2. **Wytyczne tworzenia diagramów przypadków użycia (use-cases)**

   (wg Booch G., Rumbaugh J., Jacobson I., UML przewodnik użytkownika)

3. **Model konceptualny – identyfikacja i specyfikacja wymagań**

Model biznesowy

Przykład **modelu biznesowego** dotyczącego realizacji opłat z dokumentacji środowiska Business Process Visual Architect (BP-VA) do wizualnego modelowania diagramów procesów stosując notacje BPM (Business Process Modeling Notation - BPMN).



Proces identyfikacji i specyfikacji wymagań

Czynności	Produkty wyjściowe	Opis produktu wyjściowego
lista kandydujących wymagań	lista znamionowa	status, szacowany koszt, priorytety, poziom ryzyka implementacji itp.
zrozumienie kontekstu systemu	***Model dziedziny)**** (*domain model*)-najważniejsze obiekty systemu: „rzeczy” lub zdarzenia podawane przez ekspertów	**diagram najważniejszych klas dziedziny** (*domain classes*) z niewielką ilością operacji-metod (około 10-50 w notacji UML), reszta przewidywanych klas w glosariuszu (*glossary*);
	Model biznesowy* (*business model*) - wewnętrzny model procesu biznesowego organizacji, wyszczególniany przez klientów systemu (*customers*)	**„business use case” :** **a) Opis przypadków użycia („uses cases”) i aktorów („actors”)** odpowiadających procesowi biznesowemu oraz klientom procesu biznesowego **b) biznesowy model obiektowy** (*business object model*) składający się z wykonawców (*workers*), encji biznesowych (*business entities*), jednostek pracy (*work units*) realizujących „use case”,

funkcjonalne wymagania	***Model przypadków użycia* (identyfikacja przypadków użycia z modelu biznesowego)**	proces reprezentowania wymagań jako przypadków użycia oraz innych produktów specyfikowany za pośrednictwem języka UML: **1.** **opis tekstowy** realizujących zachowanie systemu przy działaniu poszczególnych przypadków użycia lub aktorów i przypadków użycia - czyli opis sekwencji akcji odpowiednich do modyfikacji, przeglądu, projektowania i testowania, **2.** **model przypadków użycia zawierający aktorów i przypadki użycia oraz powiązania** (np. dziedziczenia) między nimi oraz: **dodatkowo diagramy**: stanów, czynności, sekwencji akcji oraz współpracy, **przepływu zdarzeń** **3.** **opis architektury** przypadków użycia , **4.** **glosariusz** - definicje ważnych termów wyprowadzanych z modelu dziedziny lub modelu biznesowego, **5.** **prototyp interfejsu użytkownika**-interakcje między aktorami - ludźmi i oprogramowaniem
niefunkcjonalne wymagania	**uzupełniające wymagania lub indywidualne wymagania**	**1.** **specjalne wymagania** zawierające niefunkcjonalne wymagania w postaci opisu tekstowego **2.** **ograniczenia środowiska i implementacji** (np. typ komputera, typ plików, rodzaj systemu operacyjnego, typ oprogramowania Internetu), zależności, konserwacja, zdolność do poszerzania,

Tworzenie modelu konceptualnego systemu informatycznego – część 1

1. **Elementy diagramów diagramów przypadków użycia (use-cases)**
2. **Wytyczne tworzenia diagramów przypadków użycia (use-cases)**

   (wg Booch G., Rumbaugh J., Jacobson I., UML przewodnik użytkownika)
3. **Model konceptualny – identyfikacja i specyfikacja wymagań**
4. **Przykłady diagramów przypadków użycia (use-cases)**

Przykład 1: Katalog tytułów i książek - opis biznesowy „świata rzeczywistego” w języku klienta

1. **Opis zasobów ludzkich**
   Pracownik wypożyczalni może dodawać do katalogu tytułów nowe tytuły. Każdy tytuł jest reprezentowany przez następujące dane: tytuł, autor, wydawnictwo, ISBN oraz informacje o liczbie egzemplarzy i miejscu ich przechowywania i występuje w bibliotece jako pojedyncza informacja dla każdego tytułu. Pewna grupa tytułów opisuje książki nagrane na kasety, dlatego dodatkowo tytuł zawiera dane nagrania np nazwisko aktora. Każdy egzemplarz, niezależnie, czy jest książką czy kasetą, jest opisany odrębną informacją zawierajacą numer egzemplarza i ewentualnie (dotyczy to wyodrębnionych egzemplarzy) informację o liczbie dni, na które można wypożyczyć egzemplarz. Numery egzemplarzy mogą się powtarzać dla różnych tytułów. Pracownik bilioteki (bibliotekarz) może dodawać nowe tytuły i egzemplarze oraz je przeszukiwać, natomiast klient może jedynie przeszukiwać tytuły i sprawdzać egzemplarze wybranych tytułów.
2. **Przepisy**
   Pracownik ponosi odpowiedzialność za poprawność danych - odpowiada materialnie za niezgodność danych ze stanem wypożyczalni. Wypożyczalnia powin-na być przyjazna dla klienta biznesowego
3. **Dane techniczne**
   Klient może przeglądać dane wypożyczalni za pośrednictwem strony internetowej lub bezpośrednio za pomocą specjalnego programu (tak obsługuje już wdrożone programy). Pracownik biblioteki może dodatkowo modyfikować i usuwać dane o tytułach oraz egzemplarzach. Zakłada się, że klientów, jednocześnie przeglądających dane wypożyczalni może być ponad 1000 oraz wypożyczalnia może zawierać kilkadziesiąt tysięcy tytułów oraz przynajmniej dwukrotnie więcej egzemplarzy. Biblioteka składa się z kilku ośrodków w różnych miastach na terenie kraju (lista miast jest dołączona do umowy). Zaleca się stosowanie technologii Java.

Wymagania stawiane tworzonej aplikacji - na podstawie opisu „świata rzeczywistego” wykonanego przez eksperta problemu reprezentującego stronę zamawiającą czyli użytkownika i klienta, przekazana wykonawcy systemu

wg Hans-Erik Erikson, Magnus Penker: UML Toolkit

Wymagania funkcjonalne

- System powinien wspierać wypożyczanie książek
- Biblioteka wypożycza podane książki i czasopisma osobom zarejestrowanym, o ile je posiada
- Biblioteka dokonuje zakupu nowych książek, przy czym popularne książki kupuje w kilku egzemplarzach. Usuwa zniszczone książki i czasopisma.
- Bibliotekarz jest pracownikiem biblioteki, komunikuje się z wypożyczającym. Jego praca jest wspierana za pomocą systemu
- Wypożyczający może zarezerwować książkę lub czasopismo, które nie jest dostępne w danej chwili, W momencie, kiedy zamówione rzeczy są dostępne- albo po zwrocie lub dzięki zakupowi, można je wypożyczyć i usunąć rezerwację. Rezerwację można usunąć niezależnie.
- Biblioteka może łatwo utworzyć, zmienić i usunąć informację o tytułach, wypożyczających, wypożyczeniach i rezerwacjach

Wymagania niefunkcjonalne

- System powinien pracować w popularnych systemach (UNIX, Windows, OS/2) i powinien mieć nowoczesny graficzny interfejs użytkownika
- System powinien się rozwijać np. wprowadzenie możliwości zawiadamiania rezerwującego książkę o jej dostępności lub dłużnika o przekroczeniu terminu wypożyczenia przez Internet

AKTOR	**OPIS**	**PRZYPADKI UŻYCIA**
Bibliotekarz	*Bibliotekarz wypożycza, rezerwuje książki i przyjmuje zwroty książek oraz usuwanie rezerwacji. Jest on odpowiedzialny za utrzymywanie zasobów biblioteki (wstawianie i usuwanie: tytułów książek, egzemplarzy książek oraz danych wypożyczających)*	• zarzadzanie • wykonaj rezerwacje • usun rezerwacje • wypozycz pozycje • zwrot pozycji
Klient	*Klient nie może bezpośrednio korzystać systemu. Korzysta on z usług bibliotekarza*	•wykonaj rezerwacje • usun rezerwacje • wypozycz pozycje • zwrot pozycji

PU wypożycz pozycje
OPIS
CEL: obsługa bibliotekarza
WS (warunki wstępne): inicjalizacja przez uruchomienie programu (np. otwarcie strony WWW, start aplikacji)
WK (warunki końcowe): pojawienie się nowej danej w aplikacji lub komunikat o przyczynach braku wypożyczenia
PRZEBIEG:

1. Pokazanie menu i wybór tytułu z listy
2. Pokazanie menu i wybór wypożyczającego z listy
3. Identyfikacja tytułu - jeśli tytuł jest w systemie, przejdź do punktu 4 lub automatycznie zakończ wypożyczanie
4. Identyfikacja dostępności egzemplarza (pozycji) – jeśli jest przynajmniej jeden egzemplarz wolny, przejdź do punktu 5 lub zakończ automatycznie wypożyczanie
5. Identyfikacja wypożyczającego – jeśli jest, przejdź do punktu 6 lub zakończ automatycznie wypożyczanie
6. Rejestracja nowego wypożyczenia zawierającego informację o wypożyczającym i wypożyczonym egzemplarzu i wywołanie **PU usun rezerwacje**

Test PU: wypożycz pozycje

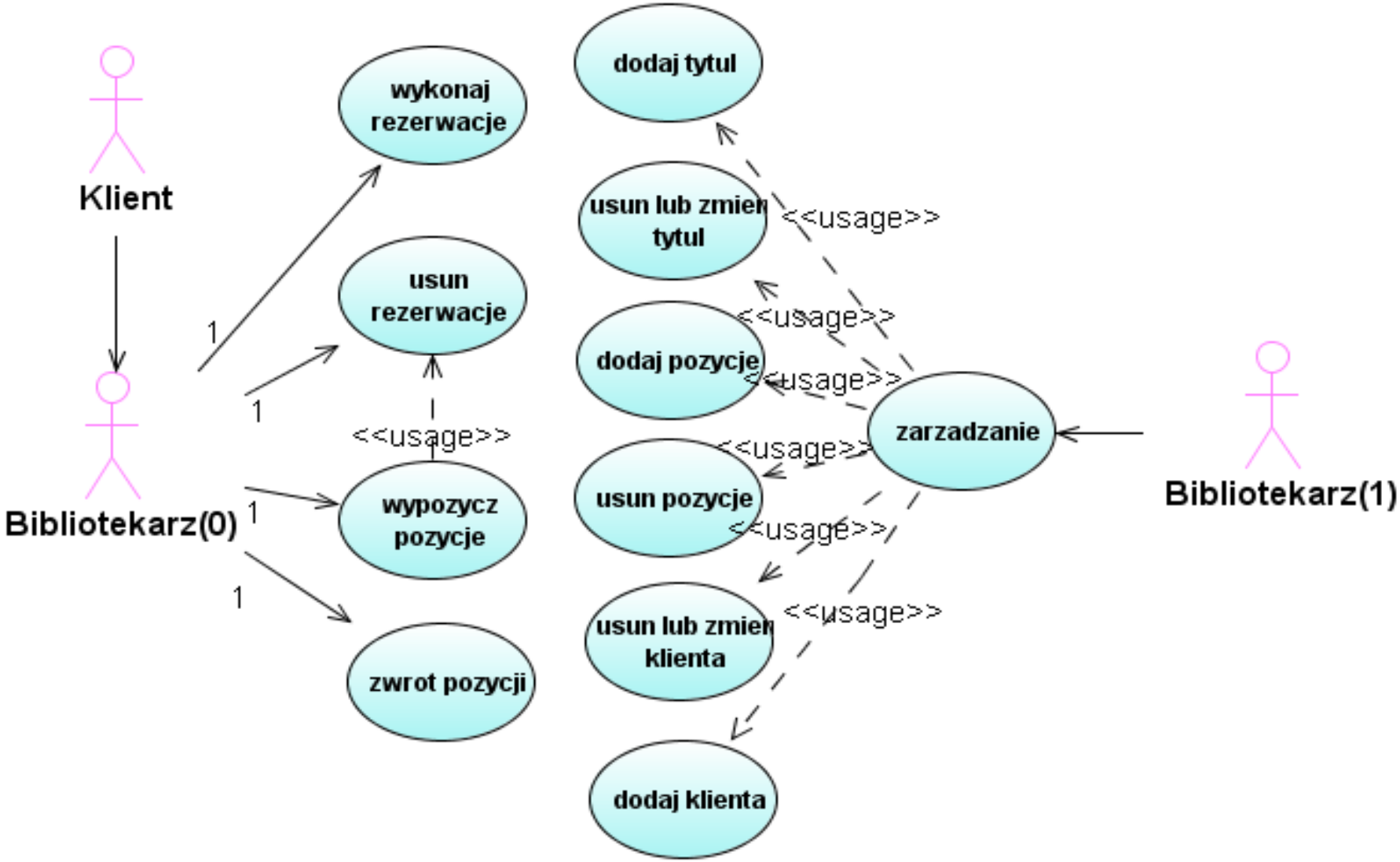
Dane wejściowe:

Dane klienta wypożyczającego książkę, tytuł wypożyczanej książki

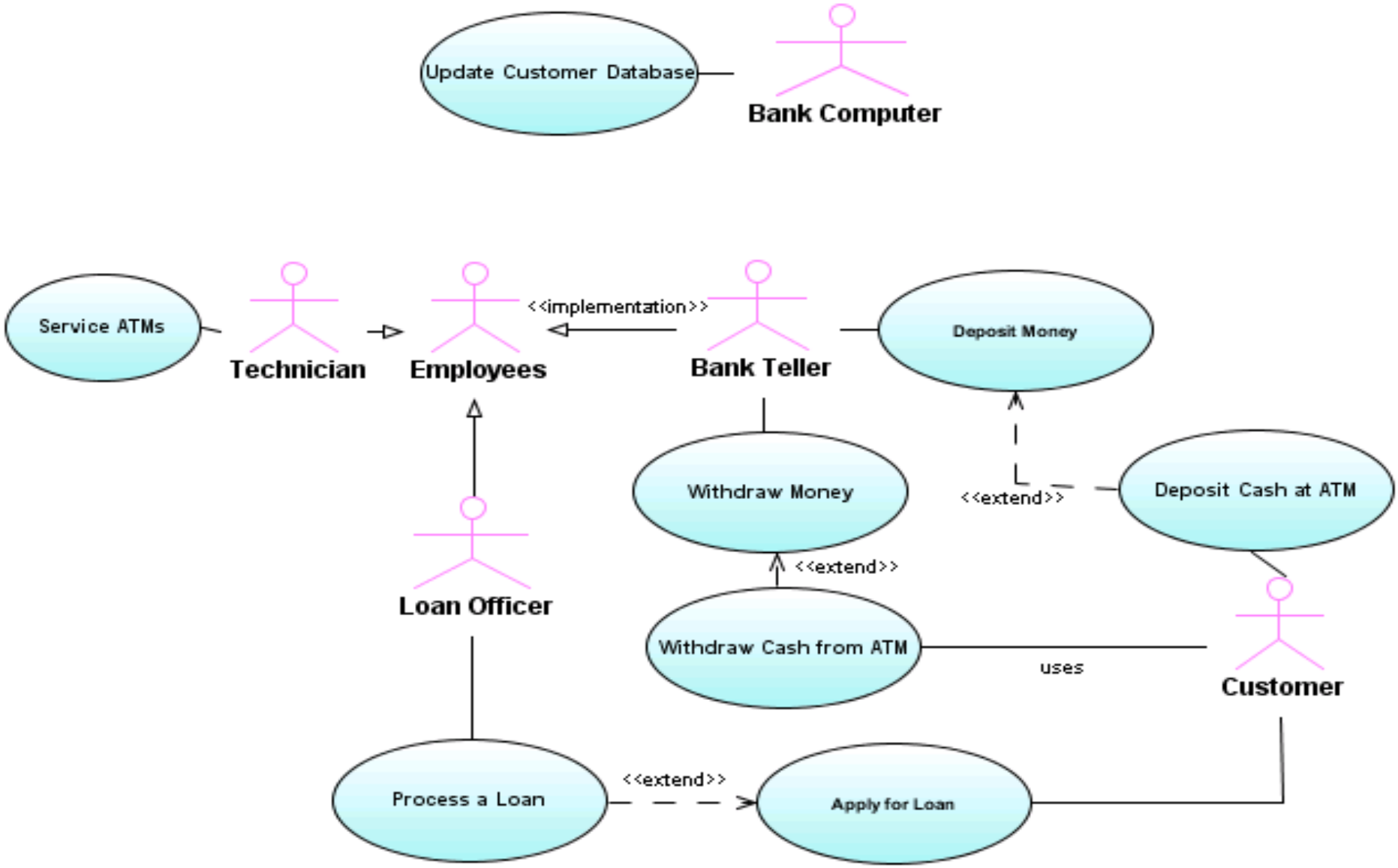
Dane wyjściowe:

- Powstanie nowego wypożyczenia, jeśli dane wejściowe były w systemie oraz była wolna książka o podanym tytule
- Usunięcie rezerwacji wypożyczonej książki, jeśli była w systemie dla danego klienta

Wypożyczalnia



Przykład 2. System bankowy: wg UML Modeling: Creating Use Case Diagrams (http://www.netbeans.org/products/uml/)



Opis diagramu przypadków użycia systemu bankowego

AKTOR	OPIS	PRZYPADKI UŻYCIA
Bank Computer		Update Customer DataBase
Employees	Aktor bazowy dla aktorów Technician, Employees, Bank Teller	
Technician		Service ATMs
Bank Teller		Deposit Money rozszerzony (extends) przez Deposit Cash from at ATM Withdraw Money rozszerzony (extends) przez Withdraw Cash from ATM
Loan Oficier		Process Loan
Customer		Withdraw Cash from ATM Deposit Cash from at ATM Apply for Loan rozszerzony (extends) przez Process Loan

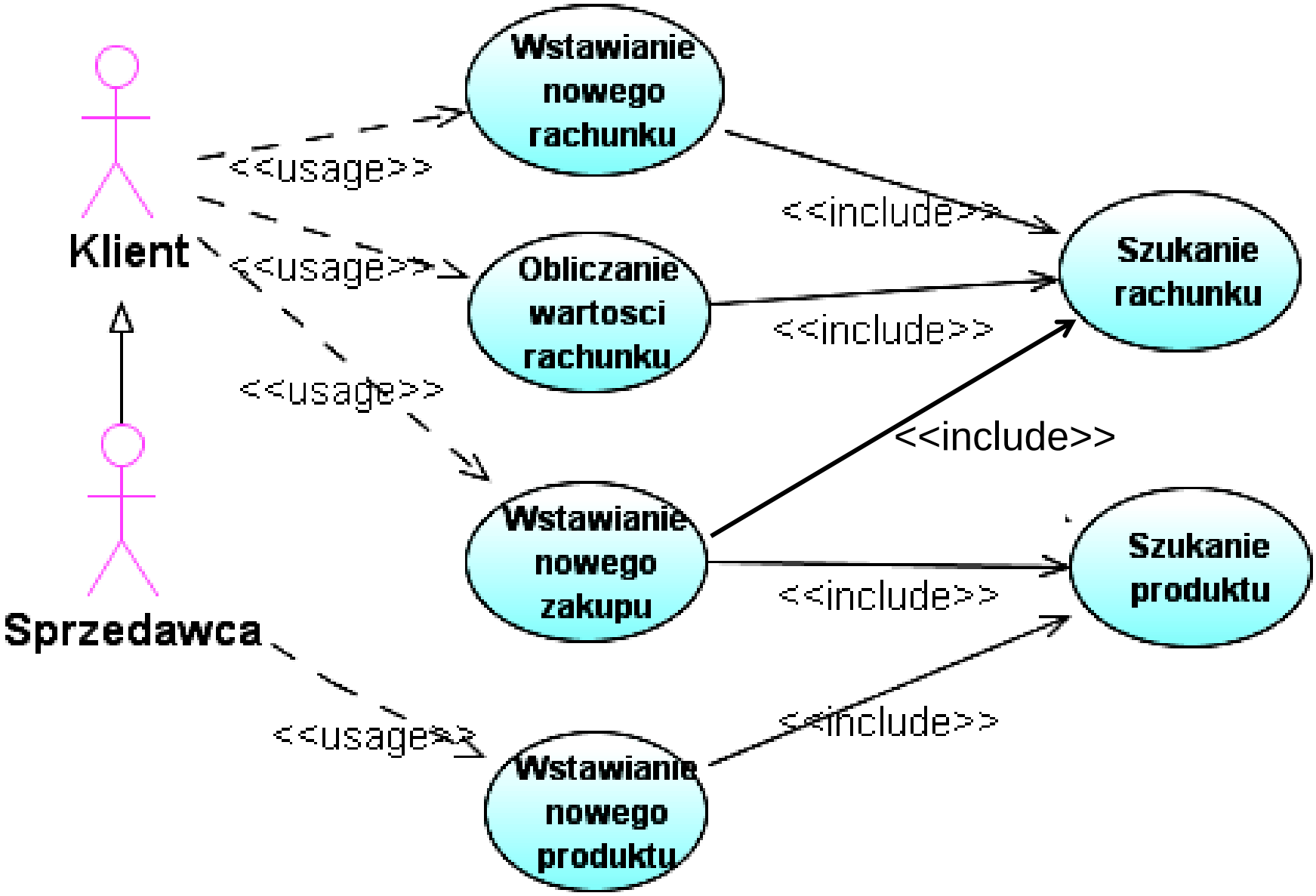
Przykład 3. System sporządzania rachunków

Lista wymagań funkcjonalnych

1. System zawiera katalog produktów
2. Można zakupić trzy typy produktów różniące się sposobem obliczania ceny detalicznej: netto, z podatkiem, z promocją,
3. Można wprowadzić wiele rachunków
4. Pozycje rachunku muszą zawierać produkty różne w sensie nazwy, ceny, podatku i promocji
5. Każda pozycja rachunku powinna podać swoją wartość brutto oraz dane produktu oraz ilość zakupionego produktu.
6. Na rachunku powinna znajdować się wartość łączna wszystkich zakupów oraz wartości zakupów należących do wybranych kategorii

Lista wymagań niefunkcjonalnych

1. Wstawianie produktów może odbywać się tylko przez uprawnione osoby
2. Wstawianie nowych rachunków oraz wstawianie nowych zakupów jest dokonywane przez klientów
3. Zakupy mogą być dokonane przez Internet przez aplikację uruchamianą przez przeglądarkę lub bez jej pośrednictwa


Wstawianie nowego rachunku
Klient
<<usage>>
<<usage>>
<<include>>
Szukanie rachunku
Obliczanie wartosci rachunku
<<include>>
<<usage>>
<<include>>
Wstawianie nowego zakupu
Szukanie produktu
<<include>>
Sprzedawca
<<usage>>
<<include>>
Wstawianie nowego produktu

AKTOR	OPIS	PRZYPADKI UŻYCIA
Klient	*Klient może dokonywać zakupów wybranych produktów przez Internet korzystając z przeglądarki lub z aplikacji*	• Wstawianie nowego rachunku powiązane przez <<include>> z PU Szukanie rachunku • Obliczanie wartosci rachunku powiązane przez <<include>> z PU Szukanie rachunku • Wstawianie nowego zakupu powiązane przez <<include>> z PU Szukanie rachunku oraz powiązane przez <<include>> z PU Szukanie produktu
Sprzedawca	*Sprzedawca może dodatkowo dodawać nowe produkty*	• Wstawianie nowego rachunku powiązane przez <<include>> z PU Szukanie rachunku • Obliczanie wartosci rachunku powiązane przez <<include>> z PU Szukanie rachunku • Wstawianie nowego zakupu powiązane przez <<include>> z PU Szukanie rachunku oraz powiązane przez <<include>> z PU Szukanie produktu • Wstawianie nowego produktu powiązane przez <<include>> z PU Szukanie produktu

Wnioski - model przypadków użycia

- Opis w języku klienta
- Zewnętrzna postać systemu
- Strukturyzacja za pomocą przypadku użycia czyli struktura postaci zewnętrznej systemu
- Używany głównie jako kontrakt między klientem i wykonawcami, określający co system powinien robić i czego nie powinien robić
- Może zawierać redundancję, sprzeczności
- Przedstawia funkcjonalność systemu, dołączając architekturę ważnej funkcjonalności
- Definiuje przypadki użycia analizowane w modelu analizy