Projektowanie oprogramowania
wykład 1

Dr inż. Gabriel Rojek

1

Literatura

1. Booch G., Jacobson I., Rumbaugh J., “UML przewodnik użytkownika” WNT 2002
2. Gamma E., Helm R., Johnson R., Vlissides J., “Wzorce projektowe. Elementy programowania obiektowego wielokrotnego użytku”, WNT 2005
3. Steve McConnel, "Kod doskonały. Jak tworzyć oprogramowanie pozbawione błędów", Wydanie II
4. Russ Miles, Kim Hamilton, "UML 2.0. Wprowadzenie", Helion, Gliwice 2007
5. Sommerville I., "Inżynieria oprogramowania", WNT 2003
6. Robert C. Martin, "Mistrz czystego kodu. Kodeks postępowania profesjonalnych programistów", Helion 2021

2

Określanie wymagań

- **Wymagania** dotyczące systemu **są opisem systemu pod względem wykonywanych przez niego funkcji** oraz przebiegu realizacji tych funkcji przez system.
- Wymagania stanowią zewnętrzny obraz systemu, tak jak przedstawia się on użytkownikom.
  - Nie powinny więc zawierać technicznych szczegółów dotyczących projektowania i implementacji kodu (np. informacji dotyczących klas, obiektów etc.)

3

Fazy określania wymagań

- Proces określania wymagań dla systemu informatycznego można podzielić na fazy:
  - faza ustalania wymagań (odkrywania wymagań),
  - faza specyfikacji wymagań (tworzenia opisu wymagań),
  - faza atestacji (walidacji, validation) wymagań.
- Powyższe fazy mogą być powtarzane wielokrotnie na różnych etapach określania wymagań, wraz z rosnącym zakresem i poziomem szczegółowości wymagań.

4

Klient & wykonawca

- Zakładać będziemy, że w określaniu wymagań uczestniczy:
  1. klient (znający dziedzinę zastosowań)
  2. wykonawca (odpowiedzialny za aspekty informatyczne, choć nie musi to być ostateczny wykonawca projektu)
- *Obie strony, muszą porozumieć się co do wielu elementów, przy czym klient często nie rozumie specyfiki funkcjonowania programów, a wykonawca nie zna specyfiki dziedziny zastosowań.*

5

Klasyfikacja wymagań

- **wymagania funkcjonalne** – dotyczące tego co ma realizować system; jakie ma spełniać funkcje, jakich dostarczać usług, jak zachowywać się w określonych sytuacjach.
- **wymagania niefunkcjonalne** – dotyczące tego jak system powinien realizować swoje zadania; np. wymagania dotyczące koniecznych zasobów, ograniczeń czasowych, niezawodności, bezpieczeństwa, przenośności, współpracy z określonymi narzędziami i środowiskami.
- Wymagania funkcjonalne powinny być:
  - kompletne – opisywać wszystkie usługi żądane od systemu
  - spójne – nie zawierać stwierdzeń sprzecznych

6

Odkrywanie wymagań

- Zrozumienie docelowego funkcjonowania systemu!
- Pomocnymi technikami w odkrywaniu wymagań są:
  - poznanie całości otoczenia systemu (poprzez obserwacje, zaznajomienie z odpowiednimi dokumentami, itp)
  - wykorzystanie istniejących systemów realizujących podobne funkcje
  - obserwacje i wywiady z przyszłymi użytkownikami systemu
  - stosowanie scenariuszy wykorzystania systemu (przypadków użycia, uses cases)
  - modelowanie systemu
  - tworzenie prototypów systemu

7

Przypadki użycia

- **Przypadek użycia oznacza interakcję z całym systemem lub jego podsystemem prowadzącą do pewnego konkretnego rezultatu.**
  - Pojedynczy przypadek użycia obejmuje zazwyczaj pewną ilość scenariuszy związanych z wariantami sposobu korzystania z systemu, zdarzeniami nietypowymi (rozszerzenia).
  - Dla określenia wymagań istotne znaczenie mogą mieć właśnie przypadki nietypowe (awarie sprzętu, błędy użytkowników), które będą testowały istotne cechy systemu pod kątem niezawodności i bezpieczeństwa działania.
- Przypadki użycia są określane mianem techniki opisywania wymagań tworzonego systemu informatycznego.

8

Przykład opisu **przypadku użycia**

Nazwa: *Dokonaj rezerwacji*
Inicjator: *Rezerwujący*
Cel: *Zarezerwować pokój w hotelu*
Główny scenariusz:
1. Rezerwujący zgłasza chęć dokonania rezerwacji
2. Rezerwujący wybiera hotel, datę, typ pokoju
3. System podaje cenę pokoju
4. Rezerwujący prosi o rezerwację
5. Rezerwujący podaje swoje potrzebne dane
6. System dokonuje rezerwacji i nadaje jej identyfikator
7. System podaje Rezerwującemu identyfikator rezerwacji i przesyła go mailem
Rozszerzenia:
1a. Pokój niedostępny.
a. System przedstawia inne możliwości wyboru
b. Rezerwujący dokonuje wyboru
1b. Rezerwujący odrzuca podane możliwości
a. Niepowodzenie

9

Wychwytywania przypadków użycia

- Analiza opisu systemu z punktu widzenia przyszłego użytkownika.
  - Wymagania konieczne (bez nich system nie będzie funkcjonował),
  - Wymagania opcjonalne (pożądane, mogą być porzucone jako pierwsze w chwili napotkania nieprzewidzianych trudności w realizacji systemu).

10

Notacja graficzna

- Do zapisu przypadków użycia stosuje się często notację graficzną, najczęściej **diagramy przypadków użycia UML**.
- *Kolejne slajdy wprowadzają w zakres języka UML i dotyczą nie tylko diagramu przypadków użycia.*

11

UML, wprowadzenie

- Unified Modeling Language
- UML **nie jest** sposobem na analizę i projektowanie systemów komputerowych.
- UML **jest jedynie** językiem modelowania używanym w procesie analizy i projektowania systemów komputerowych.
  - Tworzy się z jego pomocą **model** systemu, inaczej **abstrakcję** (w sensie *skrót, uproszczenie*) systemu.
- UML powstał w wyniku rozwoju notacji graficznych związanych z trzema metodologiami tworzenia oprogramowania obiektowego: OOAD Boocha, OOSE Jacobsona i OMT Rumbaugh. Stąd „unified” w nazwie języka!

12

Diagramy UML

- Model UML systemu jest wyrażany w szeregu diagramów przedstawiających rozmaite części i aspekty modelu.
- Jest wiele różnych diagramów, **najbardziej podstawowa klasyfikacja**:
  - Diagramy struktury – ujmują **statyczne** aspekty systemu,
  - Diagramy zachowania (behawioralne) – ujmują **dynamiczne** aspekty systemu.

13

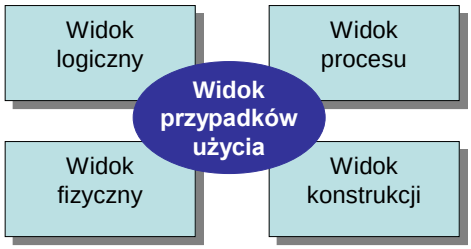
Diagramy struktury (statyczne) vs. zachowania (dynamicznie)

- diagramy struktury:
  - diagramy klas, obiektów, komponentów, struktur złożonych, pakietów, wdrożenia
- diagramy zachowania:
  - diagramy przypadków użycia, maszyny stanowej, czynności, diagramy interakcji (sekwencji, komunikacji, czasowe/harmonogramowania, przeglądowe diagramy interakcji)

14

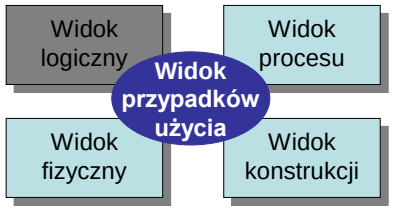
Perspektywy spojrzenia na system informatyczny

- Jednym ze sposobów rozbijania diagramów UML na perspektywy lub widoki jest **system widoków 4+1 Kruchtena.**



15

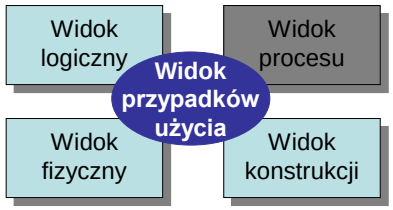
Widok logiczny



- Używany do modelowania części systemu oraz sposobów, w jaki one ze sobą współdziałają.
- Ten widok zazwyczaj tworzą diagramy:
  - Klas
  - Obiektów,
  - Maszyny stanowej,
  - Interakcji.

16

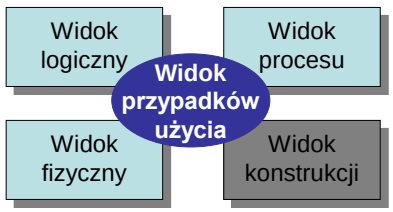
Widok procesu



- Opisuje procesy w systemie.
- Przydatny przy wizualizacji przypadków, jakie muszą zajść w systemie.
- Zawiera zazwyczaj diagram czynności.

17

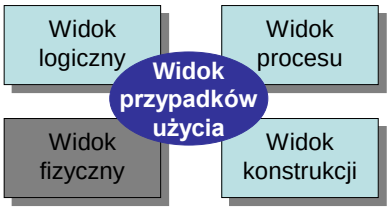
Widok konstrukcji



- Opisuje sposób, w jaki części systemu są zorganizowane w moduły oraz komponenty.
- Zawiera zazwyczaj diagramy:
  - Pakietów
  - Komponentów.

18

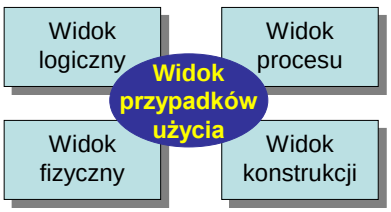
Widok fizyczny



- Wyjaśnia w jaki sposób projekt systemu opisany w trzech poprzednich widokach jest powoływany do życia w postaci zestawu rzeczywistych obiektów.
- Rzeczywiste wdrożenie systemu.
- Zawiera zazwyczaj diagramy wdrożenia.

19

Widok przypadków użycia



- Widok opisuje funkcjonalność modelowanego systemu **z perspektywy zewnętrznej**.
- Prezentuje przeznaczenie systemu.
- Wszystkie inne widoki bazują na widoku przypadków użycia (nazywa się przecież **4+1**).
- Zawiera zazwyczaj diagramy przypadków użycia, opisy i diagramy przeglądowe.

20

inżynieria wprzód / inżynieria odwrotna

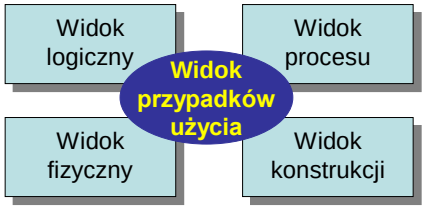
- Inżynieria wprzód - najpierw powstaje model UML systemu, a następnie na jego podstawie projektuje się i implementuje system.
- Inżynieria odwrotna (inżynieria wstecz) - budowa modelu UML na podstawie już istniejącego kodu.
  - Podejście pomocne, gdy chcemy np. dokonać modyfikacji kodu.

21

Diagram przypadków użycia UML

22

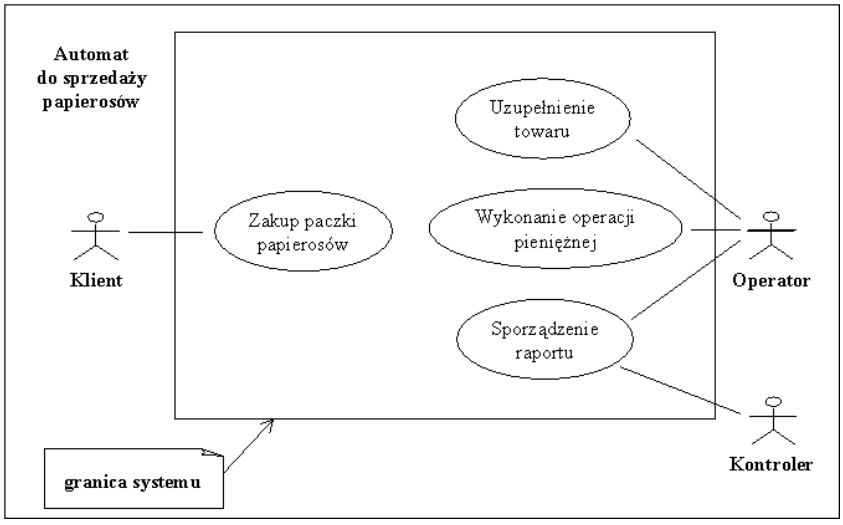
Przypadki użycia



- Przypadki użycia dotykają każdej innej części projektu systemu.
- Przypadki użycia nie określają wymagań niefunkcjonalnych systemu.

23

Diagram przypadków użycia - przykład



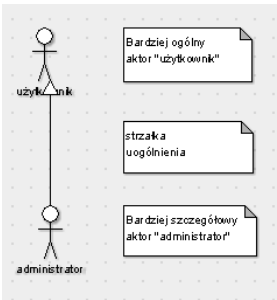
24

Aktorzy

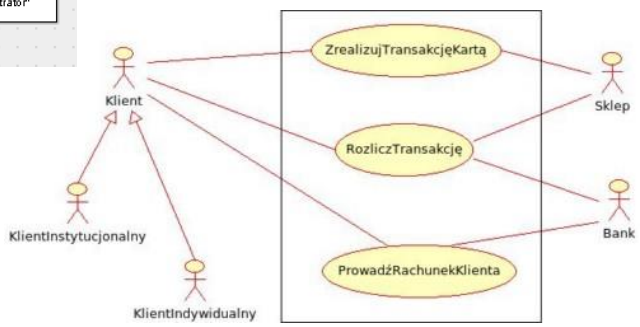
- Aktorzy mogą być:
  - Ludźmi wchodzącymi w interakcję,
  - Systemami zewnętrznymi,
  - Częściami systemu, które mają wpływ na funkcjonowanie systemu, ale same przez ten system nie mogą być zmieniane (jak np. zegar systemowy).

25

Powiązania pomiędzy aktorami



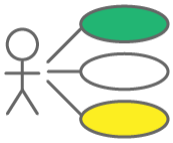
- Aby pokazać, że administrator może zrobić wszystko to, co zwyczajny użytkownik (a także kilka dodatkowych rzeczy), użyta została strzałka uogólnienia.



26

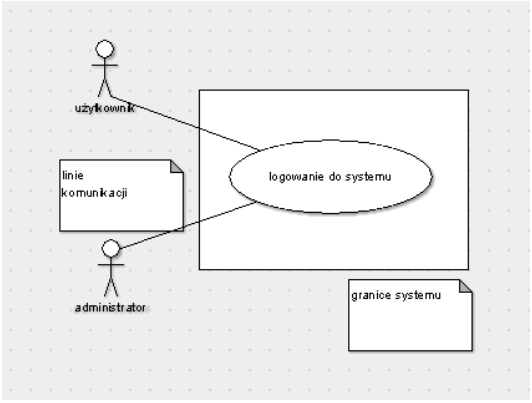
Przypadek użycia

- Jest przypadkiem, w którym dany system jest używany w celu spełniania jednego lub większej liczby wymagań użytkowników.
- Wychwytuje fragment funkcji udostępnianych przez system.
- Określają wymagania funkcjonalne systemu.



27

Linie komunikacji, granice systemu



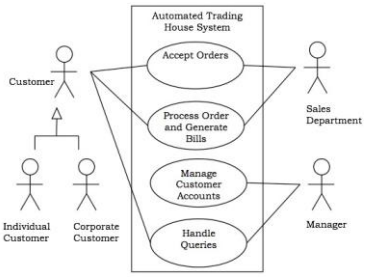
Czy „logowanie do systemu" jest dobrym przypadkiem użycia?

- Linie komunikacji sygnalizują, że aktor uczestniczy w przypadku użycia lub inicjuje go.

28

Opis przypadków użycia

- Diagram umożliwia jednym rzutem oka sprawdzić kto może wykonywać jakie czynności.
- Widać na nim jednak tylko nazwę przypadku.
  - Co ona dokładnie oznacza?
  - Jak to działa?
- Wszystkie potrzebne do projektowania algorytmów szczegóły należy przedstawić w opisie.



29

Pola w opisie przypadku użycia

- **Identyfikator** - symbol jednoznacznie identyfikujący wymaganie w formacie np. PU/X/99, gdzie: PU – przypadek użycia, X – symbol kategorii (np. od nazwy części systemu), 99 – kolejna liczba porządkowa
- **Nazwa** – nazwa przypadku użycia oznaczająca czynność – cel, jaki zostanie osiągnięty przez realizację tego przypadku, np. złożenie zamówienia
- **Aktor** – rola użytkownika wykonującego przypadek użycia
- **Zdarzenie inicjujące** – zdarzenie, które rozpoczyna wykonanie przypadku użycia
- **Warunki początkowe** – warunki, jakie muszą być spełnione, aby wykonać przypadek użycia; jeśli nie zostaną spełnione – przypadek nie rozpocznie się
- **Opis przebiegu interakcji** – poszczególne kroki wykonania przypadku użycia
- **Sytuacje wyjątkowe** – scenariusze opisujące odstępstwa od przebiegu głównego
- **Przebiegi alternatywne** – inny przebieg kroków prowadzący do tych samych warunków końcowych
- **Warunki końcowe** – efekty wykonania przypadku użycia widoczne dla użytkownika wykonującego ten przypadek oraz dla innych użytkowników
- **Powiązania** – lista identyfikatorów powiązanych przypadków użycia i wymagań
- **Częstotliwość wykonania** – określenie jak często dany przypadek użycia jest wykonywany w całym systemie – skala: rzadko, średnio, często lub konkretne szacunki liczbowe wraz z podaniem jednostki czasu

30

Opis przypadku użycia, przykłady

- **Warunki początkowe:**
  - Klient nie posiada nieopłaconych zamówień
  - Klient jest w statusie aktywnego klienta
  - Pracownik Biura Obsługi Klienta ma dostęp do rejonu, do którego należy Klient
- **Zdarzenie inicjujące:**
  - Wybranie opcji złożenia zamówienia

31

Opis przypadku użycia, przykłady

- **Przebieg w krokach:**
  1. System wyświetla formularz dodawania nowego zamówienia.
  2. Użytkownik wypełnia formularz i zatwierdza
  3. System pokazuje ceny wybranych produktów i podsumowanie zamówienia – wartość całkowitą
  4. Użytkownik zatwierdza podsumowanie zamówienia
  5. System wyświetla opcje płatności: dotpay, płatność kartą master card, SkyCash, Przelew tradycyjny
  6. Użytkownik wybiera opcję płatności
  7. System przekierowuje użytkownika do procesu płatności wybranej metody płatności
  8. Użytkownik dokonuje płatności

32

Opis przypadku użycia, przykłady

- **Przebiegi alternatywne**
  - użytkownik może podać kod rabatowy – jeśli zostanie on poprawnie zidentyfikowany, do zamówienia naliczany jest rabat o wysokości zgodnej z kategorią kodu rabatowego
  - Jeśli kod rabatowy wynosi 100%, nie pojawia się krok wykonywania płatności, zamówienie jest od razu przesyłane do realizacji – wykonują się wszystkie warunki końcowe
- **Sytuacje wyjątkowe**
  - Podane dane nie spełniają reguł walidacji – system wyświetla komunikat błędu przy błędnie wypełnionym polu, składanie zamówienia nie jest kontynuowane do czasu poprawienia błędów i ponownego zatwierdzenia
  - Płatność nie powiodła się – system wyświetla powiadomienie o niepowodzeniu płatności oraz wysyła e-mail (E/ZAM/01 w repozytorium treści e-maili), zamówienie nie zostaje złożone

33

Opis przypadku użycia, przykłady

- **Warunki końcowe:**
  - System wyświetla potwierdzenie złożenia zamówienia
  - Na e-mail klienta wysyłana jest wiadomość z potwierdzeniem zamówienia (E/ZAM/01 w repozytorium treści e-maili)
  - Zamówienie pojawia się na liście zamówień do realizacji
  - Zamówione produkty odejmowane są od stanu magazynowego
- **Częstotliwość wykonywania:**
  - Około 450 dziennie w całej Polsce

34

Use Case Example

Name: `Purchase ticket`

Participating actor: `Passenger`

Entry condition:

- `Passenger` standing in front of ticket distributor.
- `Passenger` has sufficient money to purchase ticket.

Exit condition:

- `Passenger` has ticket.

Event flow:

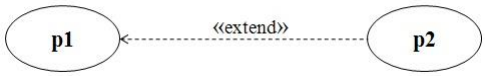
1. `Passenger` selects the number of zones to be traveled.
2. Distributor displays the amount due.
3. `Passenger` inserts money, of at least the amount due.
4. Distributor returns change.
5. Distributor issues ticket.

Anything missing?

Exceptional cases!

35

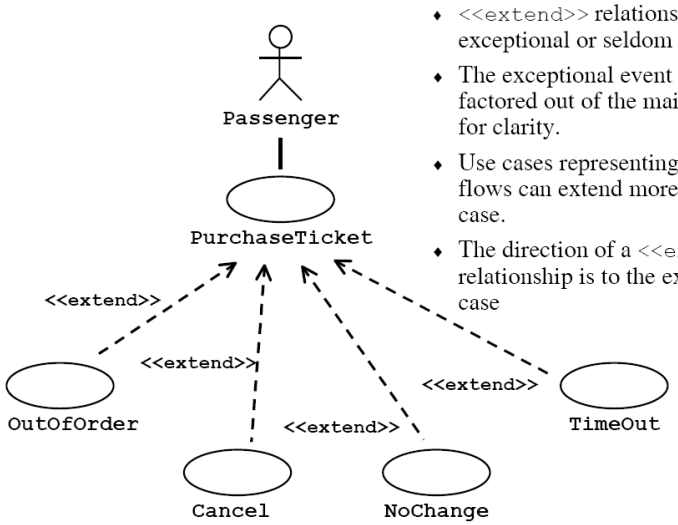
Związek rozszerzania



- Związek rozszerzenia (ang. extend) wskazuje, że dany przypadek użycia opcjonalnie rozszerza funkcjonalność bazowego przypadku użycia.
  - Funkcjonalność bazowego przypadku użycia jest rozszerzana o inny przypadek użycia po spełnieniu określonego warunku.
- Strzałka prowadzi od przypadku użycia, który czasami rozszerza inny przypadek użycia - wykorzystywane w przebiegach opcjonalnych (operacje nie zawsze wykonywane)

36

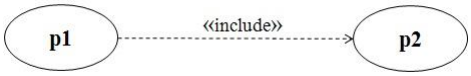
The <<extend>> Relationship



- <<extend>> relationships represent exceptional or seldom invoked cases.
- The exceptional event flows are factored out of the main event flow for clarity.
- Use cases representing exceptional flows can extend more than one use case.
- The direction of a <<extend>> relationship is to the extended use case

37

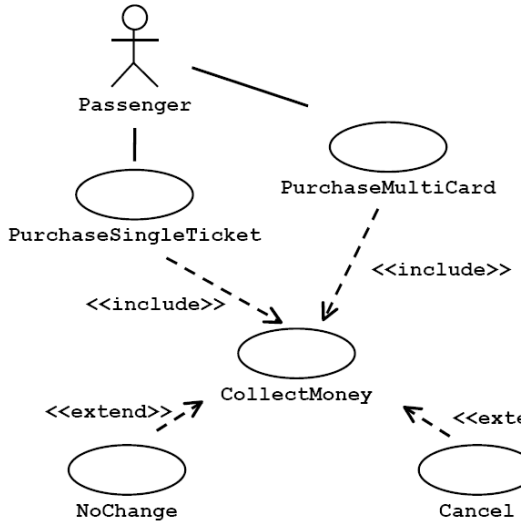
Związek zawierania



- Związek zawierania (ang. *include*) polega na rozszerzaniu funkcjonalności bazowego przypadku użycia o zachowanie innego przypadku użycia.
- Wskazuje na wspólny fragment wielu przypadków użycia; wykorzystywane w przebiegach podstawowych (operacje zawsze wykonywane).
- Istotny jest fakt, że związek zawierania zawsze skierowany jest grotem w stronę zawieranego przypadku użycia.

38

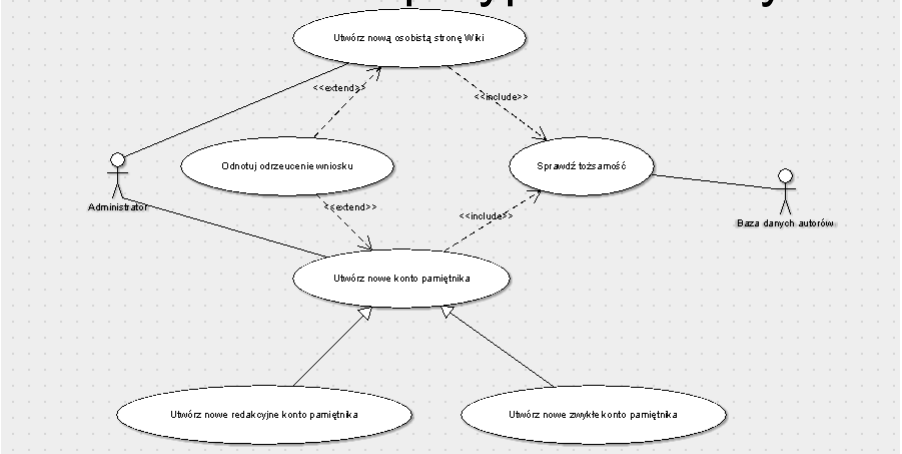
The <<include>> Relationship



- An <<include>> relationship represents behavior that is factored out of the use case.
- An <<include>> represents behavior that is factored out for reuse, not because it is an exception.
- The direction of a <<include>> relationship is to the using use case (unlike <<extend>> relationships).

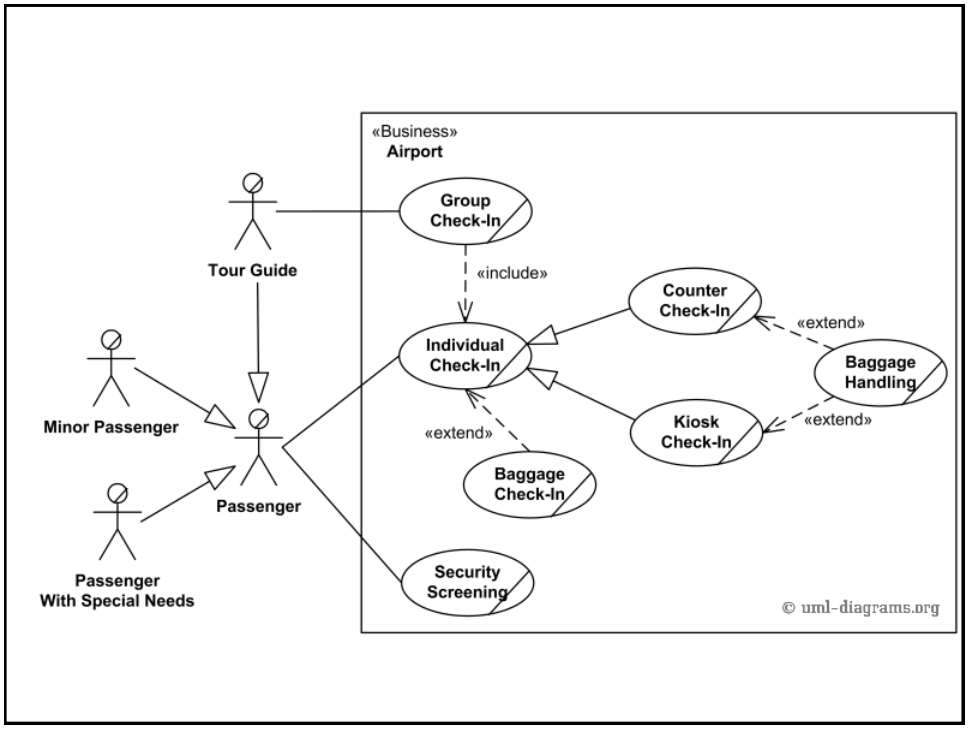
39

Dziedziczenie przypadków użycia

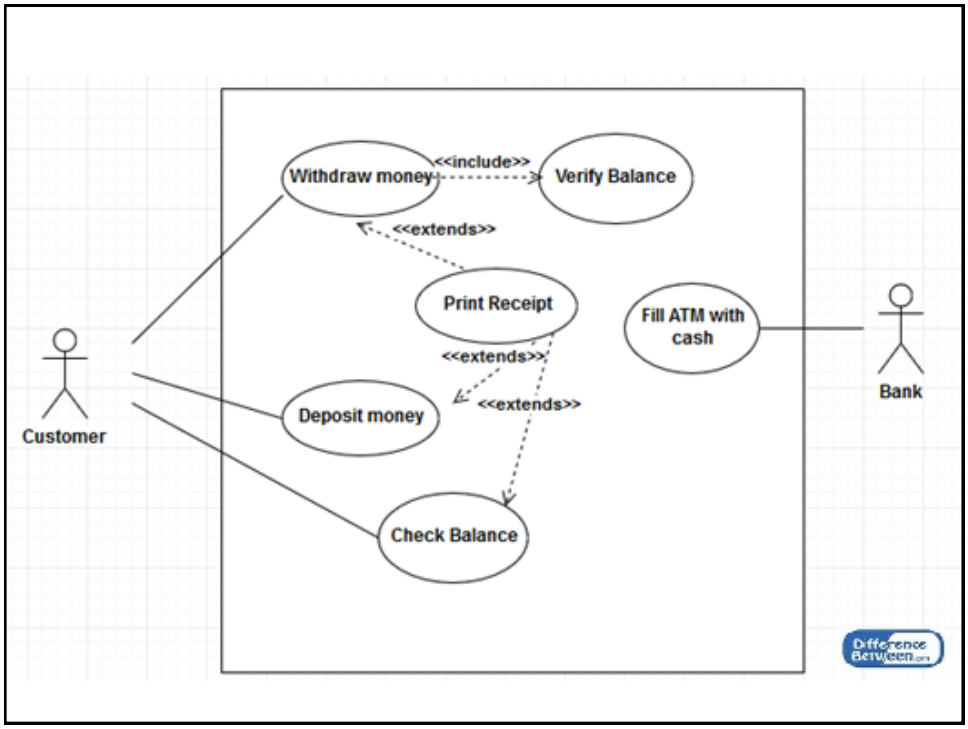


- Dziedziczenie pozwala na współdzielenie większości zachowania ogólnego przypadku użycia.

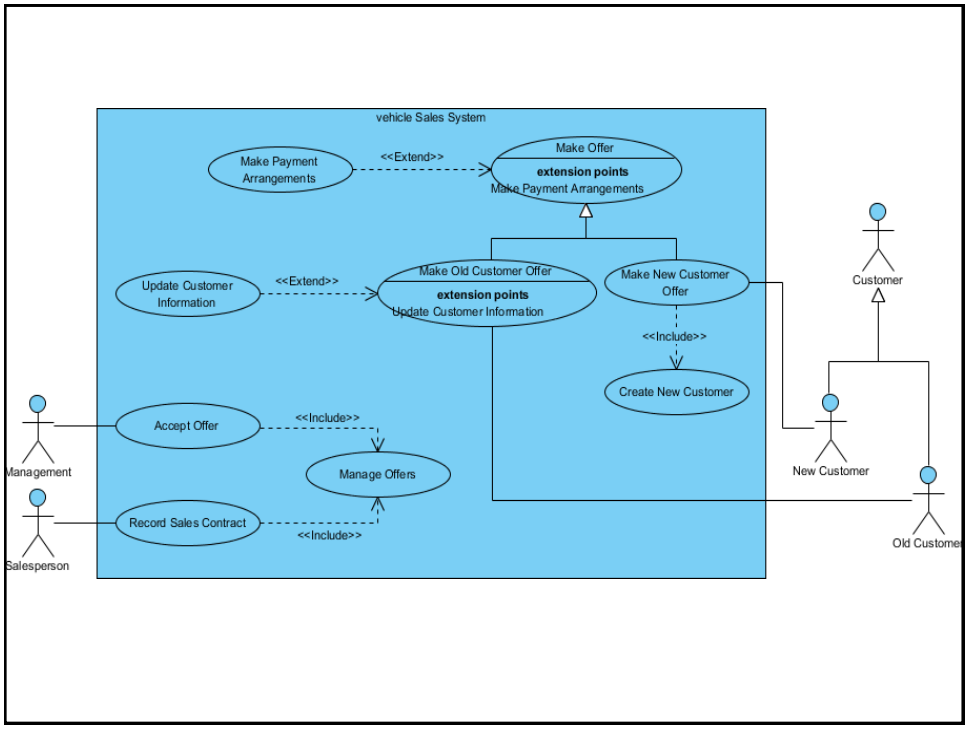
40


«Business»
Airport
Group Check-In
«include»
Individual Check-In
Counter Check-In
«extend»
Baggage Handling
Kiosk Check-In
«extend»
«extend»
Baggage Check-In
Security Screening
Tour Guide
Minor Passenger
Passenger
Passenger With Special Needs
© uml-diagrams.org

41


Withdraw money
<<include>>
Verify Balance
<<extends>>
Print Receipt
Fill ATM with cash
Bank
<<extends>>
<<extends>>
Deposit money
Customer
Check Balance

42


vehicle Sales System
Make Payment Arrangements
<<Extend>>
Make Offer
extension points
Make Payment Arrangements
Update Customer Information
<<Extend>>
Make Old Customer Offer
extension points
Update Customer Information
Make New Customer Offer
<<Include>>
Create New Customer
Customer
Management
Accept Offer
<<Include>>
Manage Offers
Salesperson
Record Sales Contract
<<Include>>
New Customer
Old Customer

43