UML. Geneza i ewolucja. Przegląd cech języka. Najważniejsze diagramy.

mgr inż. Stanisław Lota

Powtórka...

1. Porównaj system informatyczny z informacyjnym. Który z nich dostarcza więcej informacji i dlaczego?
2. Omów składniki systemu informatycznego oraz krótko je opisz.
3. Omów typy systemów informatycznych.
4. Porównaj systemy CRM oraz ERP. Do czego one służą?

UML

Zunifikowany język modelowania (*Unifield Modeling Language* – UML) jest standardowym językiem modelowania graficznego, używanym do modelowania procesów biznesowych, oprogramowania oraz architektury systemów.

UML został stworzony przez grupę ***Object Management Group*** **(OMG),**

Jest on językiem graficznym zaprojektowanym tak, aby osiągnąć jak największą elastyczność i możliwość dostosowania do konkretnych zadań, ale też **nie służy jedynie do modelowania rozwiązań zorientowanych obiektowo.**

UML

Unified Modeling Language jest językiem do:

- obrazowania (komunikacja między członkami zespołu)
- specyfikowania
- tworzenia (inżynieria wprzód i wstecz)
- dokumentowania

elementów powstałych podczas budowania systemu informatycznego.

Diagramy UML to schematy przedstawiające zbiór bytów i związków między nimi.

Model…przypomnienie

Model jest uproszczeniem rzeczywistości.

Modelowanie prowadzi do lepszego zrozumienia systemu.

Opanowanie złożonego systemu wymaga opracowania wielu wzajemnie powiązanych modeli.

W wypadku systemów informatycznych czynność tę mogą ułatwić różne spojrzenia na architekturę systemu w miarę rozwijania go.

UML

UML wskazuje sposób opracowywania i czytania poprawnych modeli.

UML nie mówi nic o tym, jakie modele należy przygotować i kiedy należy to uczynić.

Czynności związane z procesami tworzenia oprogramowania opisują **metody projektowania.**

UML

UML jest oficjalnie zdefiniowany przez Object Management Group w tzw. **metamodelu UML – Meta-Object Facility (MOF). Jak inne specyfikacje bazujące na Meta-Object Facility, metamodel UML i modele UML mogą być serializowane (zapisywane) w języku XML Metadata Interchange (XMI), opartym na standardzie XML.**

UML nie jest

Językiem programowania, chociaż istnieje możliwość generowania kodu z niektórych diagramów

Narzędziem, choć zawiera specyfikacje dla narzędzi

Metodyką, nie definiuje procesu tworzenia oprogramowania, chociaż istnieją metodyki bazujące na UML

Nie jest sposobem na analizę i projektowanie systemów komputerowych.

Umiejętność czytania oraz tworzenia diagramów klas UML jest podstawą w przypadku zawodu programisty.

UML

Głównym przeznaczeniem UML jest budowa systemów informatycznych. Z powodzeniem stosowano go już w:

- tworzeniu systemów informacyjnych przedsiębiorstw,
- usługach bankowych i finansowych,
- przemyśle obronnym i lotniczym,
- rozproszonych usługach internetowych,
- telekomunikacji,
- transporcie
- sprzedaży detalicznej
- elektronice w medycynie,
- nauce.

Trochę historii

Języki programowania obiektowego zaczęły pojawiać się między 1975, a 1989 rokiem. W tym czasie metodycy programowania poszukiwali metod analizy i projektowania zgodnych z podejściem obiektowym.

W latach 1989-1994 liczba języków projektowania obiektowego wzrosła do prawie 50, a wielu użytkowników miało kłopoty ze znalezieniem odpowiedniej metody dla siebie.

Trochę historii

W połowie lat 90-tych XX w**. Grady Booch, Ivar Jacobson oraz James Rumbaugh** rozpoczęli wspólne prace nad zunifikowanym językiem modelowania systemów informatycznych, który wspierałby paradygmat obiektowy.

W czerwcu 1996 roku opracowana została dokumentacja języka UML w wersji 0.9.

Następnie powstało konsorcjum odpowiedzialne za rozwój UML, w które zaangażowali się tacy giganci jak HP, IBM, Oracle i Microsoft. Wynikiem współpracy był UML 1.0.

W styczniu 1997 roku **UML 1.0 przekazano grupie Object Management Group (OMG), która do dzisiaj zajmuje się jego rozwojem.**

Trochę historii

Pod “skrzydłami” OMG powstały wersje 1.1, 1.2, 1.3, 1.4, 1.4.2 (ta została poddana standaryzacji ISO/IEC 19501) i ostatnią wersję z gałęzi 1.x oznaczoną numerem 1.5. W czerwcu 2005 roku OMG opublikowała wersję 2.0 UML . Następnie wydała wersje 2.1.1 i 2.1.2, 2.2, 2.3, 2.4,2.5.

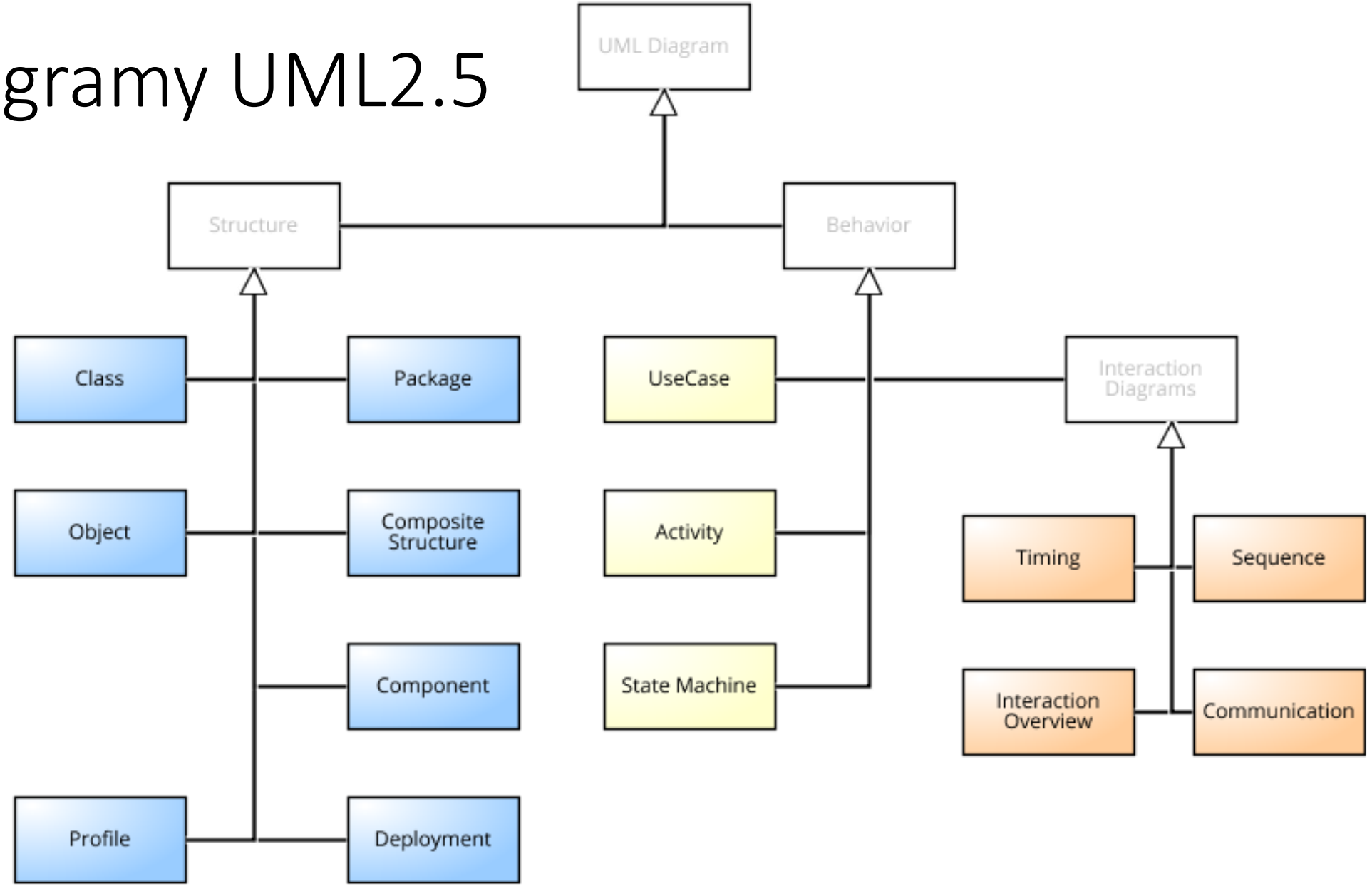
Diagramy UML 2.5

Diagramy UML to schematy przedstawiające zbiór bytów i związków między nimi.

Wyróżnić możemy podział na 14 diagramów głównych oraz 3 abstrakcyjne (struktur, zachowań i interakcji).

Istnieją niestety pewne niejednoznaczności co do stosowanego polskiego tłumaczenia diagramów, np. ang. timing diagram jest tłumaczony jako diagram czasowy, zależności czasowych, harmonogramowania, uwarunkowań czasowych czy diagram przebiegów czasowych.

Diagramy UML2.5



Diagramy struktur

Diagram klas (class diagram)

Obiektów (object diagram)

Komponentów (component diagram)

Wdrożenia (deployment diagram)

Struktur złożonych (composite structure diagram) UML 2.0

Pakietów (package diagram) UML 2.0

Profili (profile diagram) UML 2.2

Diagramy zachowań

Czynności (activity diagram)

Przypadków użycia (use case diagram)

Maszyny stanów (state machine diagram) (dla UML 1.x Stanów, statechart diagram)

Interakcji (diagram abstrakcyjny)

Komunikacji (communication diagram) (dla UML 1.x Współdziałania, collaboration diagram)

Sekwencji (sequence diagram)

Czasowy, harmonogramowania (timing diagram) UML 2.0

Sterowania interakcji (interaction overview diagram) UML 2.0

Jakie diagramy?

Projektując system informatyczny, rozpoczyna się przeważnie od tworzenia diagramów w następującej kolejności:

- Przypadków użycia
- Sekwencji
- Klas
- Aktywności

Są to najczęściej wykorzystywane diagramy. Pozostałe bywają pomijane, zwłaszcza przy budowaniu niedużych systemów informatycznych.

Narzędzia

Narzędzia do modelowania w UML czyli oprogramowanie, które pozwala tworzyć modele pomocne przy programowaniu, ale także analizie procesów biznesowych.

WWW:

https://www.draw.io/

https://creately.com/lp/uml-diagram-tool

https://yuml.me

PC:

Microsoft Visio

StarUML

Sparx Enterprise Architect

Diagram przypadków użycia

Diagram przypadków użycia (ang. *use case diagram*) jest diagramem, który przedstawia funkcjonalność systemu wraz z jego otoczeniem.

Diagramy przypadków użycia pozwalają na graficzne zaprezentowanie własności systemu tak, jak są one widziane po stronie użytkownika.

Diagramy przypadków użycia służą do zobrazowania usług, które są widoczne z zewnątrz systemu.

Diagram przypadków użycia

Diagram ten pokazuje co robi system, a nie jak to robi. Przedstawia usługi, które system świadczy aktorom, lecz bez wskazywania konkretnych rozwiązań technicznych.

Diagram ten sam w sobie zazwyczaj nie daje nam zbyt wielu informacji, dlatego też zawsze potrzebna jest do niego dokumentacja w postaci dobrze napisanego przypadku użycia.

Diagram przypadków użycia

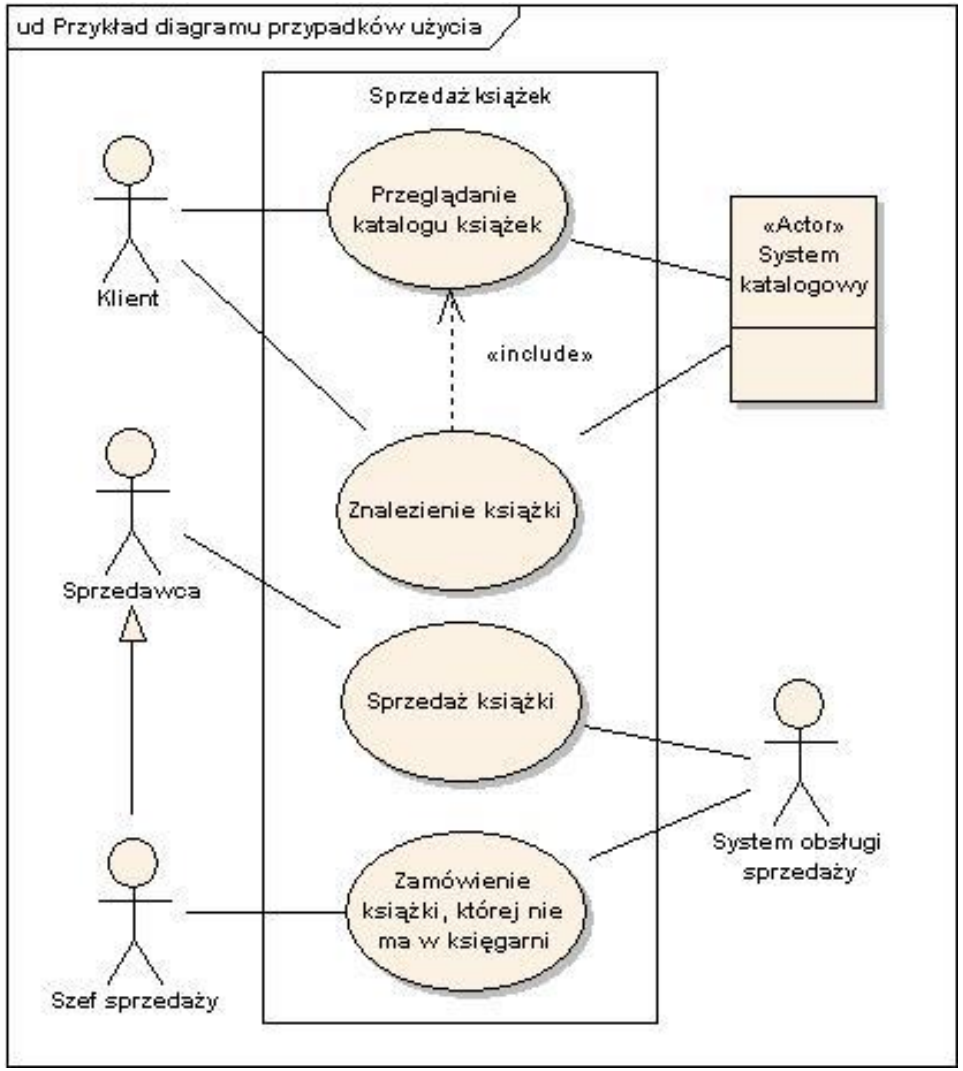


Diagram przypadków użycia

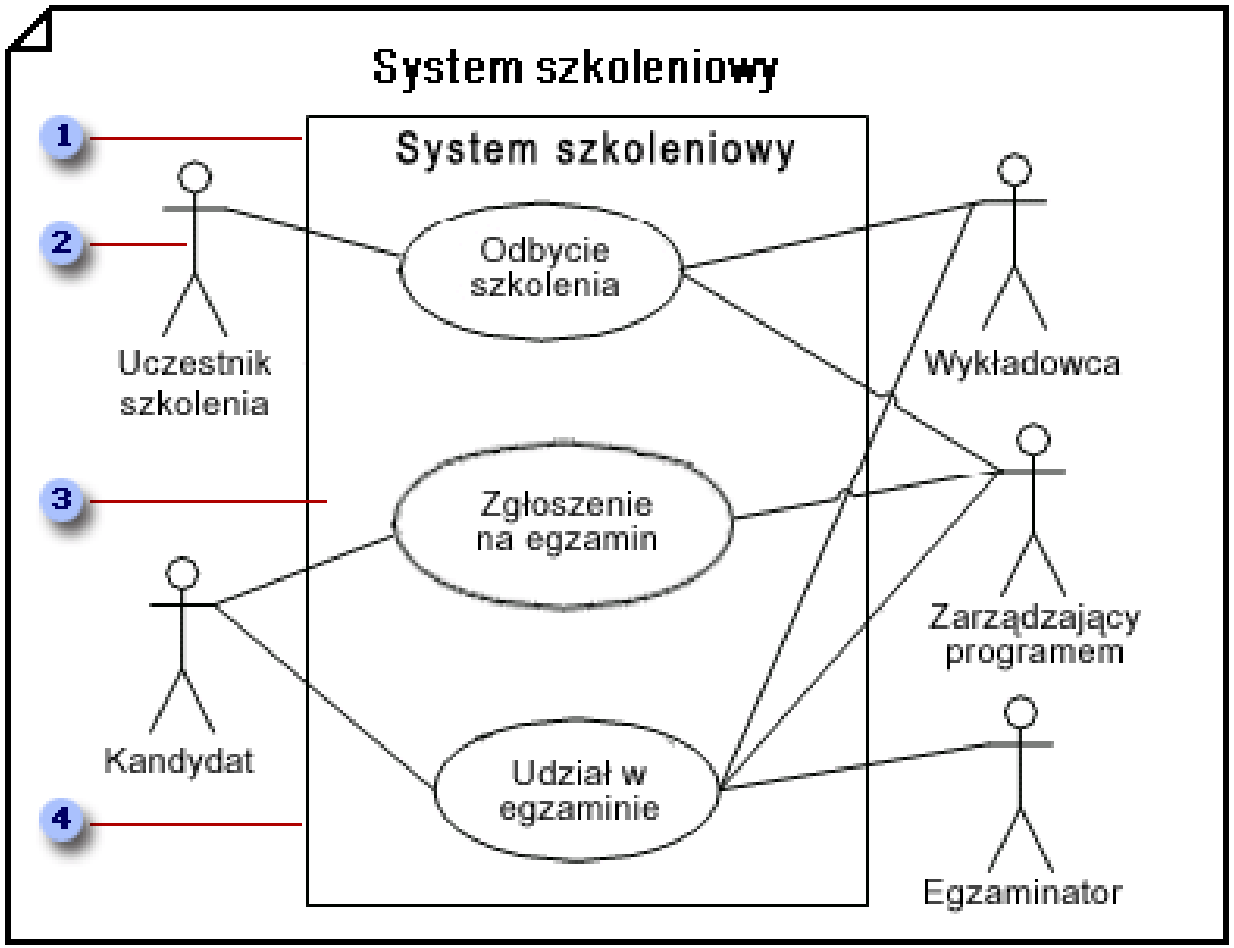


Diagram przypadków użycia

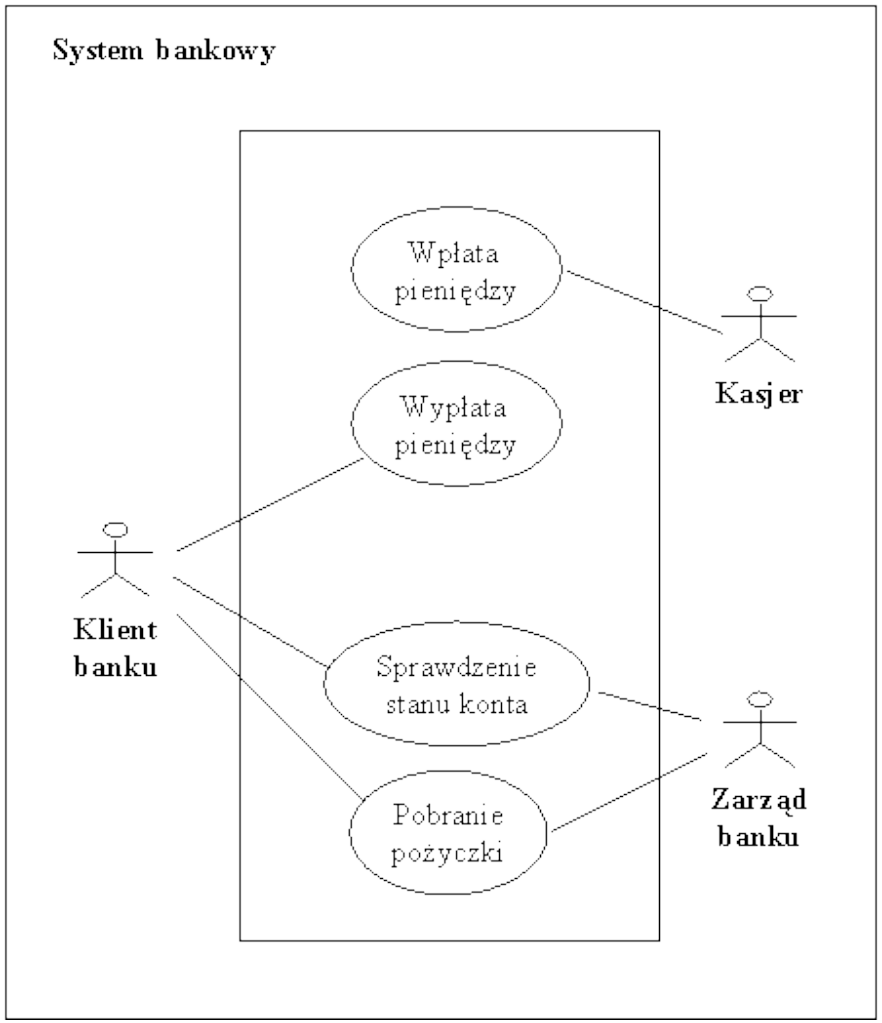


Diagram przypadków użycia

- identyfikacja oraz dokumentacja wymagań,
- umożliwia analizę obszaru zastosowań, dziedziny przedmiotowej,
- pozwala na opracowanie projektu przyszłego systemu,
- stanowi przystępną i zrozumiałą platformę współpracy i komunikacji twórców systemu, inwestorów i właścicieli,
- jest rodzajem umowy, kontraktu pomiędzy udziałowcami co do zakresu i funkcjonalności przyszłego systemu,
- stanowi podstawę testowania funkcji systemu na dalszych etapach jego cyklu życia.

Diagram przypadków użycia

Diagram przypadków użycia składa się z następujących kategorii pojęciowych:

przypadków użycia,

aktorów,

związków.

Diagram przypadków użycia

Przypadek użycia (ang. use case) – specyfikacja ciągu akcji i ich wariantów, które system (lub inna jednostka) może wykonać poprzez interakcję z aktorami tego systemu.

W związku z tym przypadek użycia jest kompleksowym działaniem realizowanym w projektowanym systemie w konsekwencji określonej aktywności aktora. Zakres danego działania determinowany jest przez wszystkie wzajemnie powiązane ze sobą przypadki użycia. Pojedynczy przypadek użycia to reprezentant spójnej jednostki funkcjonalności, którą dostarcza system.

Diagram przypadków użycia

Aktor (ang. actor) – spójny zbiór ról odgrywanych przez użytkowników przypadków użycia w czasie interakcji z tym przypadkiem użycia. **Aktorzy (czyli użytkownicy) systemu mogą być ludźmi lub zewnętrznymi systemami.**

Aktorem osobowym może być osoba, zespół, dział, instytucja, organizacja, zrzeszenie organizacji lub organizacja wirtualna. Nazwy aktorów osobowych często pokryte są z nazwami funkcji jakie pełnią w organizacji, projekcie lub przedsięwzięciu bądź nazwą stanowiska jakie piastują.

Diagram przypadków użycia

Natomiast aktorem bezosobowym może być system zewnętrzny (podsystemy, bazy danych), urządzenie lub czas. W tym drugim przypadku warto je oznaczać używając notacji prostokątnej.

Każdy przypadek użycia jest powiązany z jednym lub wieloma aktorami, ciągła linia ze strzałką oznacza, że pewien aktor jest szczególnym przypadkiem innego (szef sprzedaży jest także sprzedawcą).

Diagram przypadków użycia

Każdy aktor, który jest na diagramie przypadków użycia musi być bezpośrednio powiązany z co najmniej jednym przypadkiem użycia. Podobnie każdy przypadek użycia musi być użytkowany co najmniej przez jednego aktora (niejednokrotnie są to powiązania pośrednie).

Według standardu UML możemy wyróżnić cztery rodzaje związków:

- asocjacja wskazuje na komunikację dwukierunkową pomiędzy przypadkiem użycia a aktorem.
- uogólnienie polega na tym, że pewien przypadek użycia może być szczególną odmianą innego, już istniejącego przypadku użycia.
- oraz zależność i realizację.

Diagram sekwencji

Diagram sekwencji(ang. sequence diagram) służy do prezentowania interakcji pomiędzy obiektami wraz z uwzględnieniem w czasie komunikatów, jakie są przesyłane pomiędzy nimi.

Diagramy sekwencji dobrze ukazują dynamiczne aspekty realizacji scenariuszy, w których dochodzi do złożonych oddziaływań pomiędzy obiektami. Podstawowymi oddziaływaniami są wymiany komunikatów oznaczane następującymi symbolami:

- przesłanie komunikatu asynchronicznego – strzałka zwykła
- wywołanie funkcji – strzałka z wypełnionym grotem
- powrót z funkcji – strzałka rysowana linią przerywaną

Symbole oddziaływań można uzupełniać nazwami i specyfikacjami (np. sygnatury wywołań, typy przesyłanych argumentów).

Diagram sekwencji

Zastosowaniem diagramów sekwencji jest modelowanie zachowania systemu w kontekście scenariuszy przypadków użycia.

Diagramy sekwencji pozwalają uzyskać odpowiedź na pytanie, jak w czasie przebiega komunikacja pomiędzy obiektami.

Dodatkowo diagramy sekwencji stanowią podstawową technikę modelowania zachowania systemu, które składa się na realizację przypadku użycia.

Diagram sekwencji

Na diagramach sekwencji w łatwy sposób ilustrować można tworzenie i niszczenie obiektów (poprzez odpowiednie umieszczenie obiektów i odpowiednie symbole).

Diagramy sekwencji mogą zawierać dodatkowe konstrukcje umożliwiające zapis takich aspektów realizacji programu jak instrukcje warunkowe czy pętle. Realizuje się to poprzez umieszczenie na diagramie sekwencji odpowiednich regionów – ramek interakcji.

Diagram sekwencji

Diagramy sekwencji mogą także zawierać informacje o ograniczeniach czasowych realizowanych działań (istnieją też specjalne diagramy do tego celu – timing diagrams – które pomijamy ze względu na fakt, iż odpowiadają zachowaniu rzadko obecnemu w typowym oprogramowaniu).

Diagram sekwencji

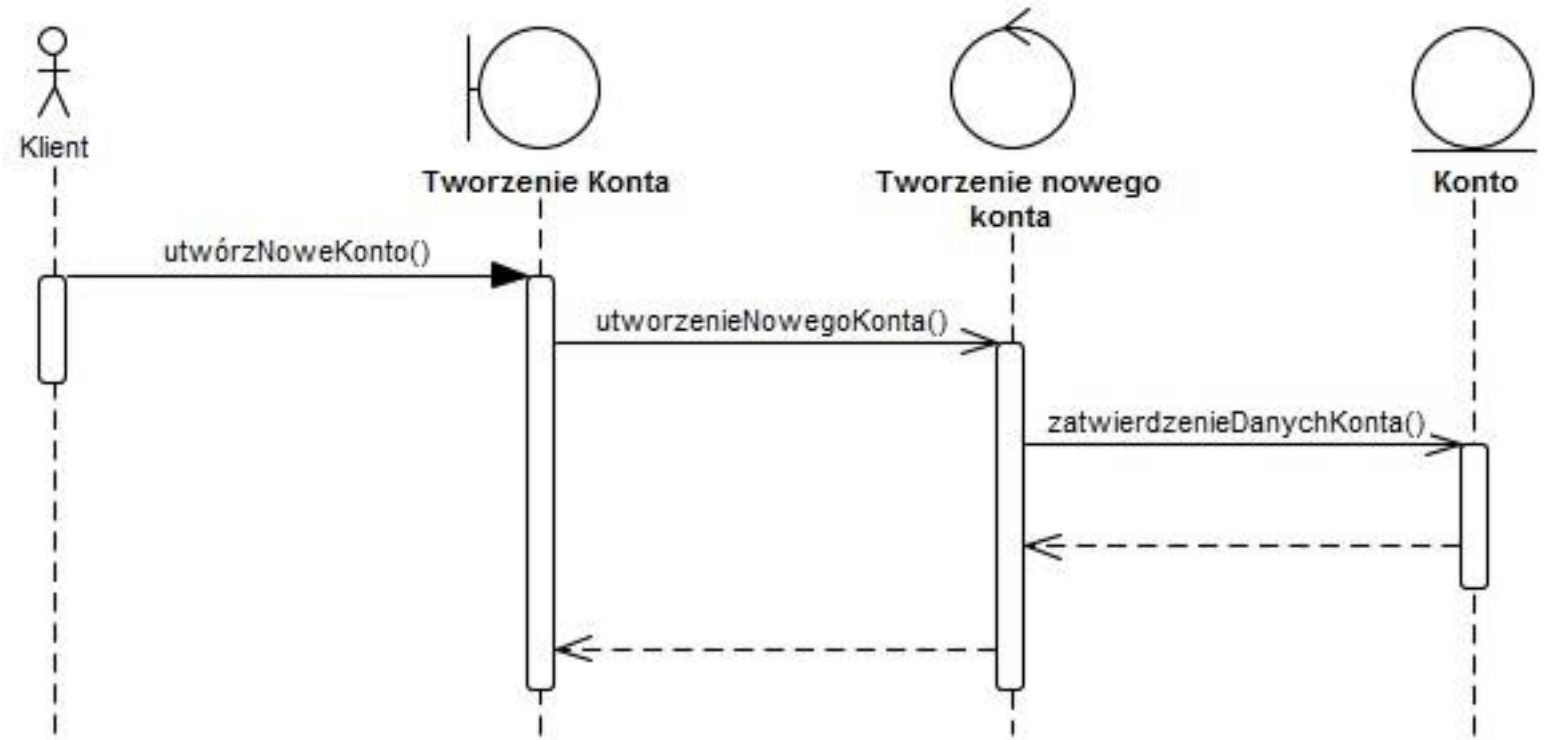
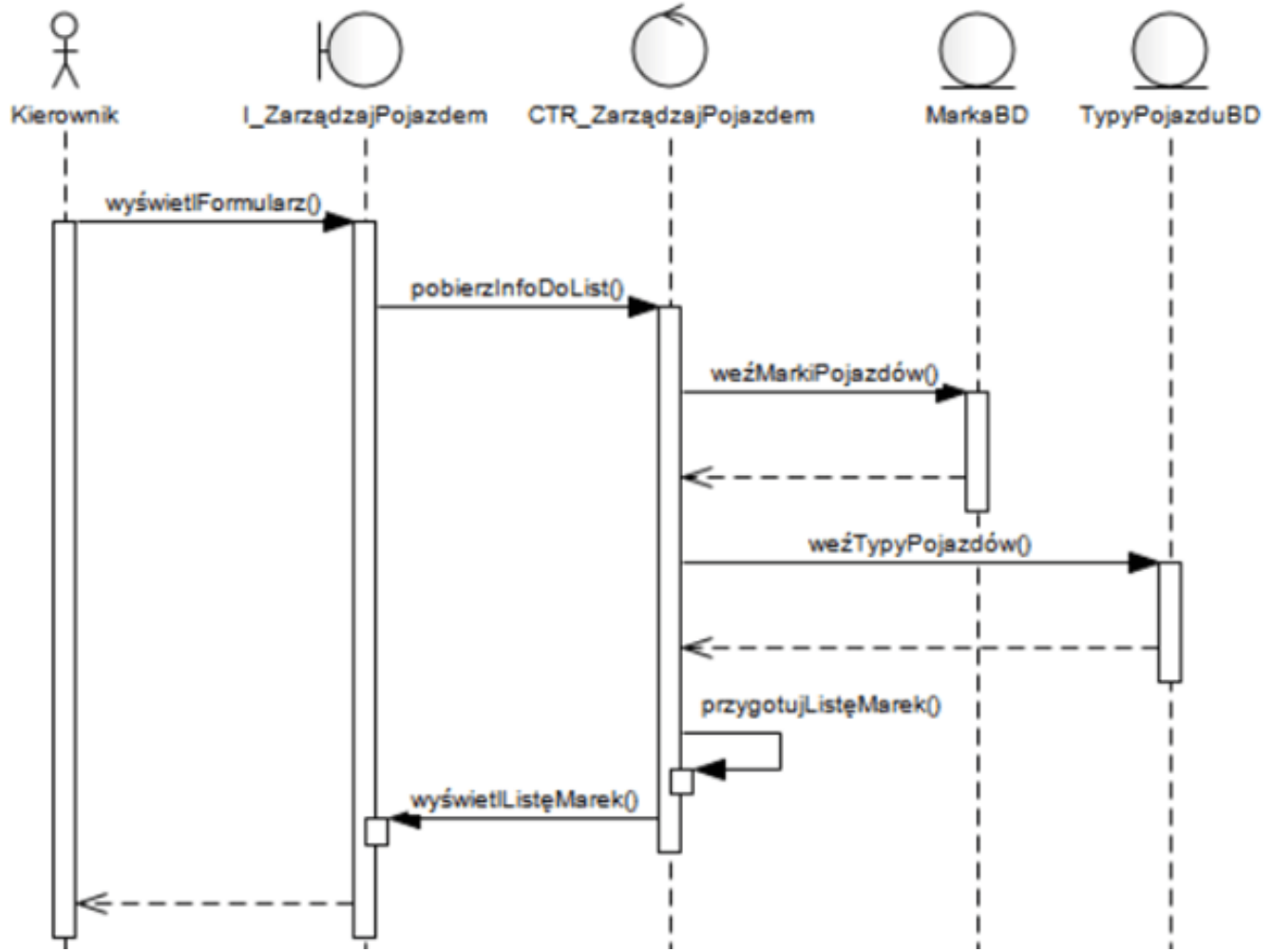


Diagram sekwencji



Diagramy klas

Diagram klas(ang. class diagram) obrazuje pewien zbiór klas, interfejsów i kooperacji oraz związki między nimi.

Diagram klas jest ściśle powiązany z projektowaniem obiektowym systemu informatycznego lub wręcz bezpośrednio z jego implementacją w określonym języku programowania.

Diagramy klas UML mogą pokazywać strukturę całego systemu bądź tylko jego części (i najczęściej tak jest).

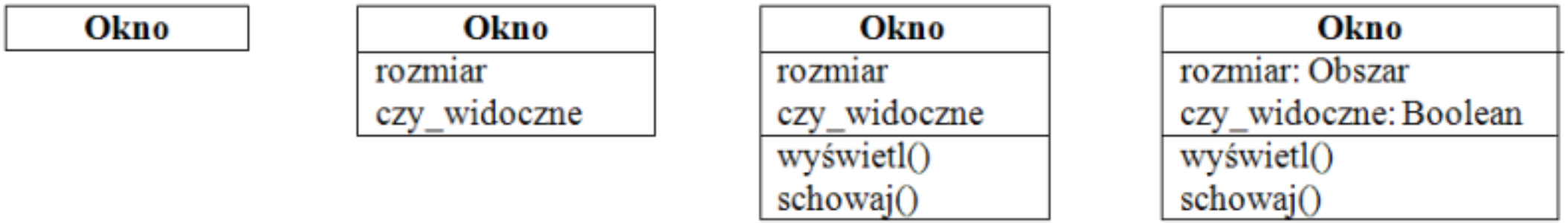
Diagramy klas

Doskonale obrazują one poglądową strukturę programu. Definiują **jakie metody i pola powinna zawierać dana klasa**. Pokazują one także część ich implementacji. Oczywiście diagram przedstawia **tylko typy obiektów** bez ich instancji – za to odpowiedzialny jest **diagram obiektów UML**.

Należy pamiętać, że **informacje zawarte w diagramie klas UML są poglądowe**. Mają ułatwić implementację klas oraz **ukazać zasadę funkcjonowania systemu**.

Diagramy klas

Klasa w modelu UML programu obiektowego jest reprezentowana przez prostokąt z umieszczoną wewnątrz nazwą klasy. Oddzielona część prostokąta pod nazwą klasy może zawierać atrybuty klasy, czyli metody (funkcje), właściwości (properties) lub pola (zmienne). Każdy atrybut pokazywany jest przynajmniej jako nazwa, opcjonalnie także z typem, wartością i innymi cechami.



Diagramy klas

Jeżeli w **diagramie klas UML** brakuje jakiegoś składnika, nie oznacza to, że taki nie istnieje w systemie. **Poprawne jest także całkowite ukrycie składników klasy**, pozostawiając tylko jej nazwę, ale tylko w wypadku gdy nie jest to klasa silnie decydująca o modelu systemu (o wysokim priorytecie).

Diagramy klas

Metody klasy mogą znajdować się w osobnej części prostokąta. Każda metoda jest pokazywana przynajmniej jako nazwa, a dodatkowo także ze swoimi parametrami i zwracanym typem.

Atrybuty (zmienne i właściwości) oraz metody mogą mieć też oznaczoną widoczność (zakres znaczenia ich nazw) jak następuje:

"+" dla public – publiczny, dostęp globalny

"#" dla protected – chroniony, dostęp dla pochodnych klasy (wynikających z generalizacji)

"–" dla private – prywatny, dostępny tylko w obrębie klasy (przy atrybucie statycznym) lub obiektu (przy atrybucie zwykłym)

"~" dla package – pakiet, dostępny w obrębie danego pakietu, projektu.

Diagramy klas

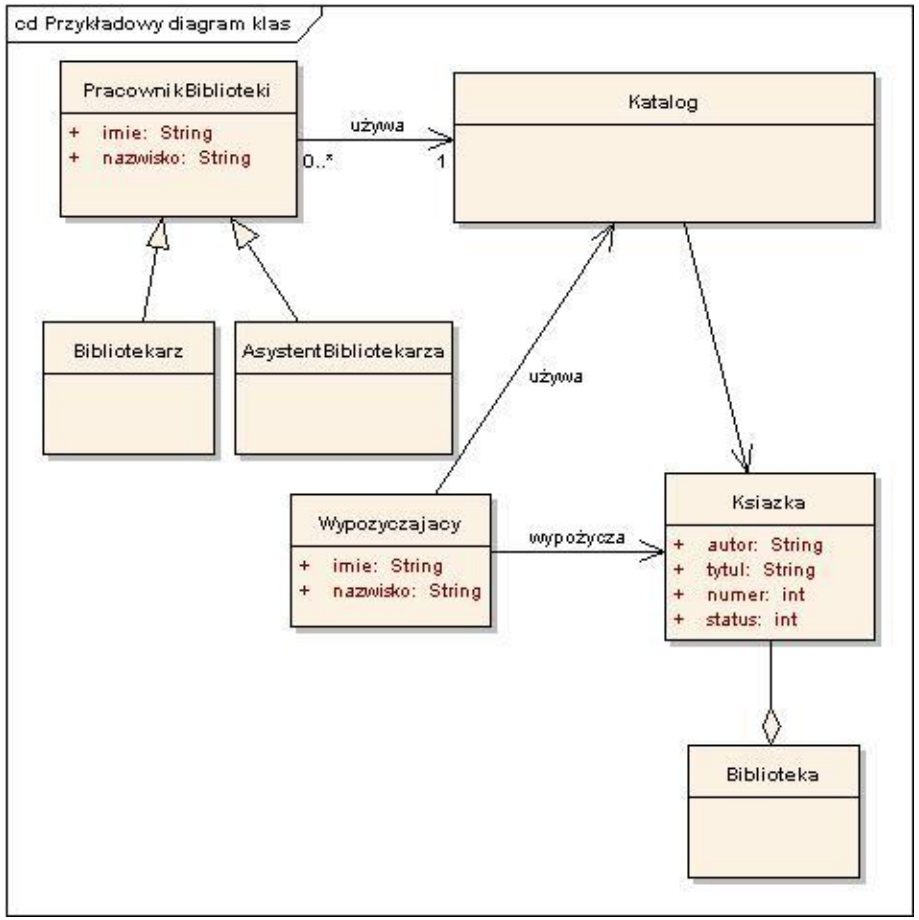


Diagram klas

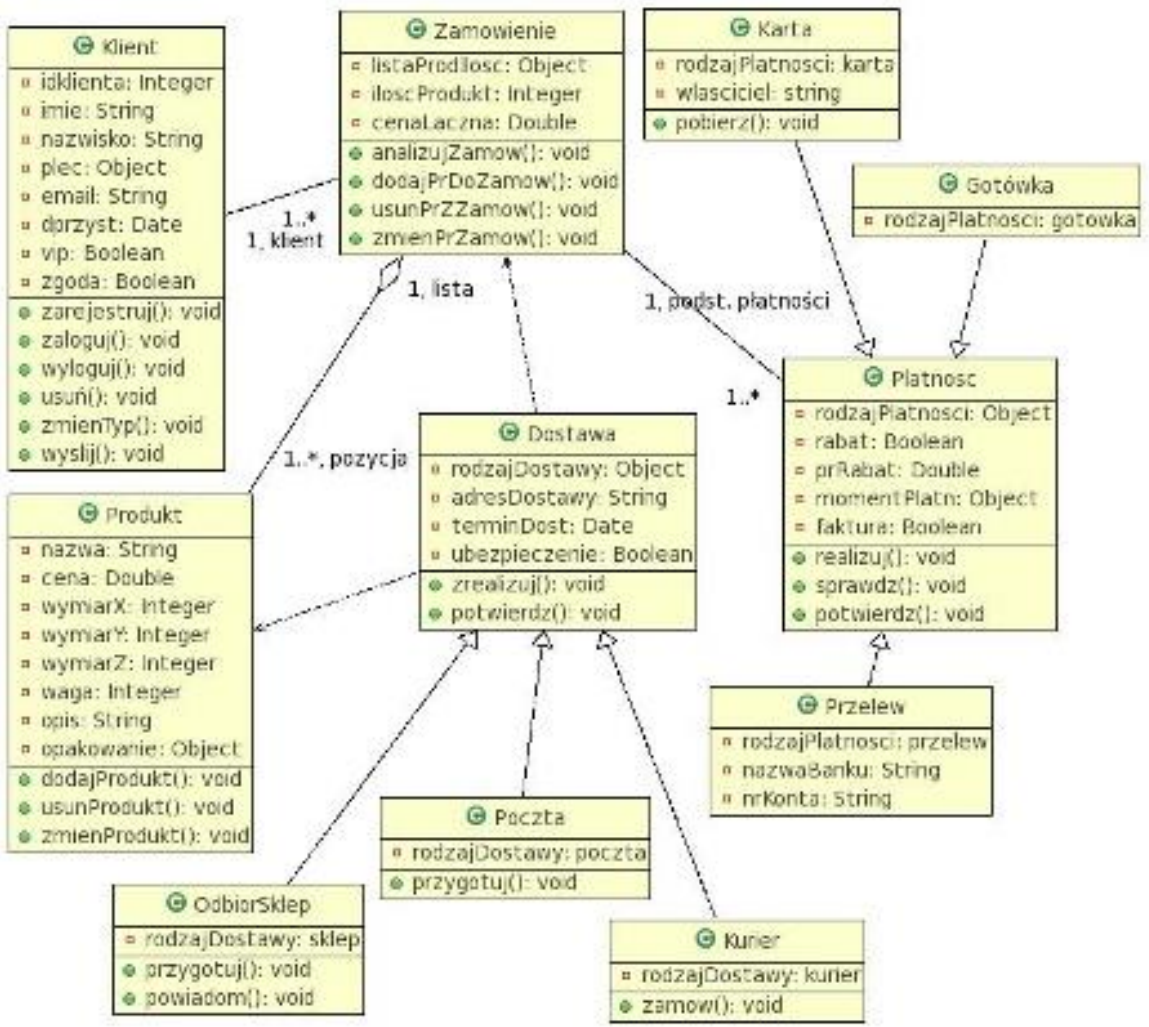


Diagram klas

Związki pomiędzy klasami są oznaczane liniami, które mogą, opcjonalnie, mieć strzałkę wskazującą kierunek relacji.

Końce związków mogą także być oznaczone krotnościami: mówią one o tym, ile obiektów danej klasy może brać udział w danej relacji. Wyróżnia się agregację, asocjację, generalizację oraz kompozycję.

Diagram klas

Agregacja reprezentuje związek typu całość-część. Występuje tutaj relacja posiadania — co oznacza, że elementy częściowe mogą należeć do większej całości, jednak również mogą istnieć bez niej. Na diagramie agregację oznacza się za pomocą linii zakończonej pustym rombem

Asocjacja wskazuje na trwałe powiązanie pomiędzy obiektami danych klas (np. firma zatrudnia pracowników). Na diagramie asocjację oznacza się za pomocą linii, która może być zakończona strzałką (oznaczającą kierunek powiązania klas). Nazwę cechy wraz z krotnością umieszcza się w punkcie docelowym asocjacji.

Diagram klas

Generalizacja – związek opisujący dziedziczenie po klasach. Na diagramie generalizację oznacza się za pomocą niewypełnionego trójkąta symbolizującego strzałkę (skierowaną od klasy pochodnej do klasy bazowej).

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. Na diagramie, kompozycję oznacza się za pomocą linii zakończonej wypełnionym rombem.

Diagram aktywności

Diagram aktywności(ang. activity diagram) zwany czasami diagramem czynności w języku UML służy do modelowania czynności i zakresu odpowiedzialności elementów bądź użytkowników systemu.

Diagram aktywności nie opisuje szczegółów jak te czynności są wykonywane a jedynie pokazuje w jakiej kolejności i kiedy. Dzięki swojej budowie pokazuje, jakie są akcje, a także jak są one połączone.

Diagram aktywności

Diagramy aktywności stosuje się w modelowaniu:

- wysokopoziomowych procesów biznesowych,
- systemów oraz podsystemów,
- scenariuszy przypadków użycia,
- procesów systemowych charakteryzujących się dużą liczbą równoległych
- czynności i sytuacji decyzyjnych,
- algorytmów.

Diagram aktywności

Diagramy aktywności składają się z następujących podstawowych elementów:

- czynności / akcji
- przepływów sterowania
- początku
- końca
- zakończenia przepływu.

Diagram aktywności

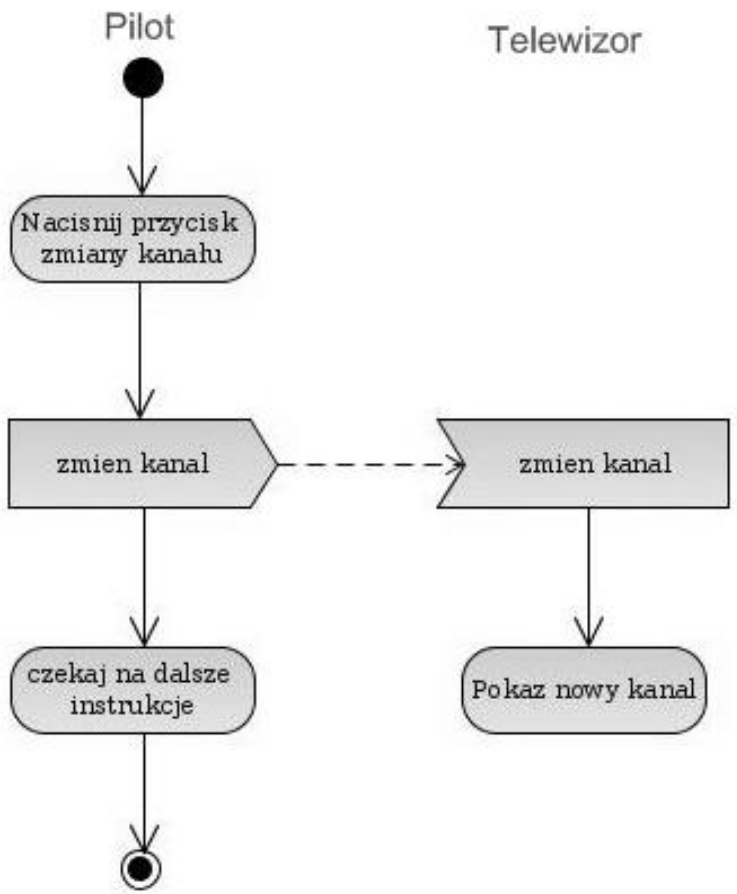


Diagram aktywności

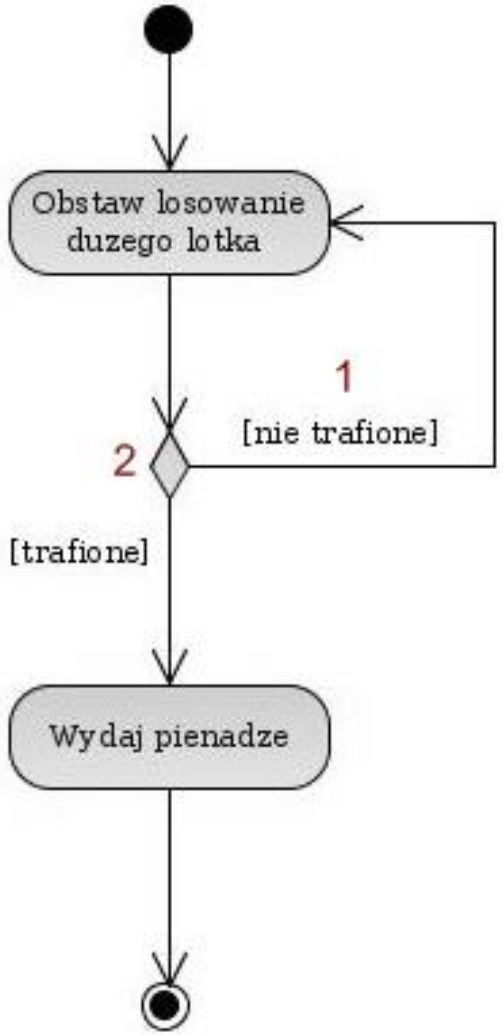


Diagram aktywności dla wyjścia z wykładu?