Projektowanie Systemów Informatycznych 2011/2012

Kontakt

- e-mail: **dmirowska@wne.uw.edu.pl**
  - w tytule proszę wpisać: „**PSI**” i **nr grupy**, do której Student uczęszcza
  - mail powinien zawierać: **imię i nazwisko** Studenta
- dyżur: wtorek 9.45-11.15 s.3
- strona: coin.wne.uw.edu.pl/dmirowska

Po co modeluje się systemy?

- lepsze zrozumienie funkcjonowania systemu przez laików
- dostosowanie poziomu abstrakcji do potrzeb zamawiającego, ukrycie szczegółów
- „zamodelowany” system łatwiej i szybciej się programuje – ułatwienie pracy programistów

Po co modeluje się systemy?

- „zamodelowany” system łatwiej się modyfikuje np. programista, który go nie tworzył a chce wprowadzić modyfikacje bez modelu musi przejrzeć wszystkie źródła

oszczędność czasu

oszczędność pieniędzy

Projektowanie systemów

- Dwa podejścia do projektowania systemów:
- **strukturalne**
- **obiektowe**

wiele metodyk i notacji z nimi związanych

„wojny metodologiczne” w latach 80-tych

- Lata 80-te – dominacja podejścia **strukturalnego** – problemy unifikacyjne

rozwój innych podejść m.in. **obiektowego**

Podejście obiektowe

- Lata 90-te - **modele obiektowe** w centrum zainteresowania twórców i użytkowników systemów jako mogące sprostać wyzwaniom:

- gospodarce opartej na wiedzy: e-business, e-health, e-government, e-learning – powszechność aplikacji internetowych

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, *Język UML 2.0 w modelowaniu systemów informatycznych*, Helion, Gliwice 2005

Podejście obiektowe

- globalizacja gospodarki – integracja systemów informatycznych w telekomunikacji, bankowości, edukacji, transporcie, turystyce;
- powszechność i dostępność aplikacji m.in. internetowych dla potrzeb społeczeństwa informacyjnego;
- multimedialny charakter aplikacji wykorzystujący dźwięk, głos, grafikę, film.

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, *Język UML 2.0 w modelowaniu systemów informatycznych*, Helion, Gliwice 2005

Podejście obiektowe

- **Cel**: tworzenie oprogramowania będącego odbiciem fragmentu rzeczywistości
- Każdy MO opiera się na pewnych podstawowych pojęciach i kategoriach:
  - **obiekt** – egzemplarz klasy, każdy byt (rzecz lub pojęcie)
  - **klasa** – ogólna kategoria obiektów mających te same atrybuty i operacje,

J. Schmuller, *UML dla każdego*, Helion, Gliwice 2003

Podstawowe pojęcia obiektowości

- **dziedziczenie** – obiekt dziedziczy atrybuty i operacje swojej klasy, klasa może dziedziczyć od innej klasy,

- **polimorfizm** – operacja może w różnych klasach mieć tę samą nazwę i oznaczać w poszczególnych przypadkach inne działanie,

- **komunikat** – żądanie wykonania operacji przez współpracujące ze sobą obiekty,

- J. Schmuller, *UML dla każdego*, Helion, Gliwice 2003

Podstawowe pojęcia obiektowości

- **hermetyzacja** – różnicowanie dostępu do obiektu; obiekt ukrywa to co robi przed innymi obiektami i światem zewnętrznym,
- **interfejs** – „twarz” obiektu konieczna by zainicjować operację; ma go każdy obiekt, pozwala innym obiektom lub ludziom wykonywać jego określone operacje.

J. Schmuller, *UML dla każdego*, Helion, Gliwice 2003

Kto bierze udział w projektowaniu systemu?

- zamawiający
- użytkownicy
- projektanci
- programiści
- analitycy

UML…

- **U**nified **M**odeling **L**anguage – Zunifikowany Język Modelowania to:

  „***graficzny język wizualizacji, specyfikowania, tworzenia i dokumentowania składników systemów informatycznych***” (def. OMG)

- jest graficznym odzwierciedleniem tworzonego systemu, składającą się z logicznie powiązanych ze sobą diagramów; diagramy składają się na model systemu, który powinien być rozwijany wraz z rozwojem firmy i systemu

UML…

- pozwala pośredniczyć między tym, co przeciętny człowiek rozumie przez działanie programu a jego fizyczną realizacją w postaci kodu,
- dzięki UML modele są zrozumiałe dla klientów, zleceniodawców, analityków, programistów, użytkowników; kody zazwyczaj tylko dla programistów.
- jest czymś w rodzaju zapisu nutowego dla muzyków, czy matematycznych symboli dla wszystkich ludzi.

UML…

- jego podstawą teoretyczną jest obiektowość
- powstał w wyniku rozwoju metod i analizy projektowania obiektowego na przełomie lat 80’ i 90’ – spory nad standaryzacją metod,
- 1995 r. pierwsza wersja UML – **G. Booch, I. Jacobson, J. Rumbaugh,**

Historia UML

- 1996 r. - Konsorcjum UML (HP, IBM, Microsoft, Oracle) -> UML 1.0,
- 1997 r. Object Management Group zajmuje się rozwojem UML,
- 1.1, 1.2, 1.3, 1.4, 1.4.2, 1.5,
- 2005 r. wersja 2.0,
- 2009 r. wersja 2.2.

UML – dla przypomnienia

- jest graficznym odzwierciedleniem tworzonego systemu, składającym się z logicznie powiązanych ze sobą diagramów,
- diagramy składają się na model systemu,
- model powinien być rozwijany wraz z rozwojem firmy i systemu

Model i diagram

- model – abstrakcja zawierająca wszystkie elementy potrzebne do opisania modelowanego zjawiska lub zagadnienia.

- diagram – sposób obserwacji zjawiska; sposób patrzenia na część lub całość modelu; zbiór bytów.

Element modelowania pojawiający się tylko raz w modelu może pojawić się na kilku diagramach.

R.A. Maksimchuk, E.J. Nalburg, *UML dla zwykłych śmiertelników*, Mikom, Warszawa 2007

Rodzaje diagramów

- diagramy struktury – odnoszą się do statycznej struktury elementów w systemie
  - diagram klas,
  - diagram pakietów,
  - diagram komponentów,
  - diagram wdrożeniowy,
  - diagram struktur połączonych.

Rodzaje diagramów

- diagramy zachowań – odnoszą się do dynamicznego zachowania elementów w systemie
  - diagram przypadków użycia,
  - diagram stanów,
  - diagram czynności (aktywności).

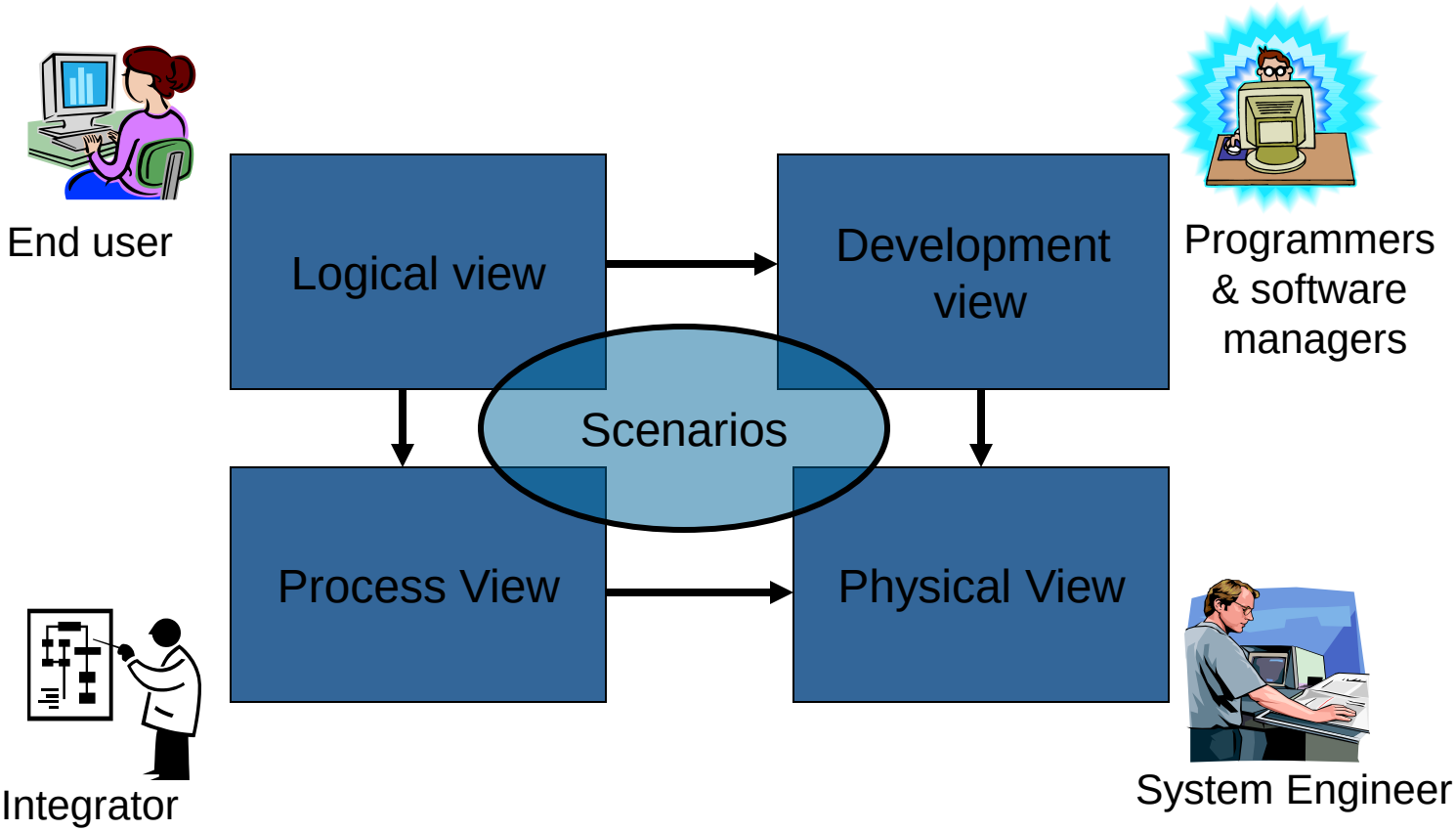
Rodzaje diagramów

- diagramy interakcji są częścią diagramów zachowań
- diagramy sekwencji,
- diagramy kooperacji (współdziałania, komunikacji),
- diagram przebiegów czasowych.

Podejście 4+1 do modelowania

- pozwala spojrzeć na system z 5 różnych perspektyw odzwierciedlających najważniejsze aspekty systemu
- różne diagramy ilustrują różne perspektywy/spojrzenia na system

Podejście 4+1 do modelowania



Mazeiar Salehie, *4+1 View Model of Software Architecture*, prezentacja
http://www.google.pl/url?sa=t&rct=j&q=4%2B1%20model%20of%20
architecture&source=web&cd=4&ved=0CEIQFjAD&url=http%3A%2F%2Fplg.
uwaterloo.ca%2F~holt%2Fcs%2F446%2F08%2Fslides%2Fmazeiar-kruchten-4%2B1.

Perspektywy i diagramy je ilustrujące

- Scenariuszy: funkcjonalność systemu z perspektywy zewnętrznej – d. przypadków użycia
- Logiczna: modelowanie poszczególnych części systemu i sposobu współpracy między nimi – d. klas, obiektów, interakcji, maszyny stanowej
- Konstrukcji: organizacja części systemu w moduły i komponenty – d. pakietów i komponentów
- Procesów: wizualizacja przypadków, które muszą zajść w systemie – d. czynności
- Fizyczna: jak jest wdrażany system – d. wdrożenia

Narzędzia CASE

- CASE: **C**omputer-**A**ided-**S**oftware/**S**ystem-**E**ngineering to:
- *„narzędzia wspierające wytwarzanie i utrzymywanie oprogramowania”*

  Częściej
- *„narzędzia wspierające, obejmujące wspomaganie specyfikacji wymagań, analizę, projektowanie i implementację systemów”*.

Definicje z: S. Szejko (red.), *Metody wytwarzania oprogramowania*, Mikom, Warszawa 2002

Rodzaje narzędzi CASE

- narzędzia wspomagające weryfikację, walidację i testowanie,
- narzędzia wspomagające zarządzanie projektami,
- wspomagające zarządzanie konfiguracją oprogramowania,
- wspomagające metodologie obiektowe,
- itd.

S. Szejko (red.), *Metody wytwarzania oprogramowania*, Mikom, Warszawa 2002

Zalety narzędzi CASE

- narzędzia CASE wspomagające proces tworzenia systemu prowadzą m.in. do:
- redukcji czasu i kosztów,
- wyższej jakości systemu,
- zapewnienia semantycznej poprawności diagramów i modelu,
- niektóre pozwalają na automatyczne generowanie szkieletowego kodu źródłowego,
- pozwalają na generowanie dokumentacji.

S. Szejko (red.), *Metody wytwarzania oprogramowania*, Mikom, Warszawa 2002

Narzędzia CASE dla użytkowników UML

- ArgoUML,
- Enterprise Architect,
- Objecteering,
- IBM Rational Rose,
- Poseidon for UML,
- Borland Together,
- StarUML
- inne

Uwaga!

- nie wszystkie programy obsługują wszystkie wersje UML,
- nie wszystkie programy pozwalają na generowanie kodu w preferowanym przez nas języku.
- Można sprawdzić na stronie:

http://www.oose.de/service/uml-werkzeuge.html

Diagramy przypadków użycia

- przedstawiają wykorzystanie systemu widziane z perspektywy zewnętrznej (aktorów) np. użytkowników lub systemów zewnętrznych,
- pokazują funkcjonalność systemu i jego interakcje ze światem zewnętrznym,
- opisują co robi system z punktu widzenia zewnętrznego obserwatora,
- przedstawiają **co** robi system, **nie jak** to robi ani z jakich zasobów systemowych korzysta,

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, *Język UML 2.0 w modelowaniu systemów informatycznych*, Helion, Gliwice 2005

Przypadek użycia

- pojedyncze **zadanie lub cel**,
- specyfikacja **ciągu akcji**, które system może wykonać przez interakcje z aktorami,
- nazwa PU – **polecenie wykonania funkcji** w systemie sformułowane w trybie rozkazującym, np.: Dokonaj rezerwacji,

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, *Język UML 2.0 w modelowaniu systemów informatycznych*, Helion, Gliwice 2005

Aktor

- **zbiór ról** odgrywanych przez użytkowników PU w trakcie interakcji z tym PU,
- może być **osobowy** lub **nieosobowy**,
- odzwierciedla role pełnione przez obiekty będące instancją klasy,
- może **inicjować** PU, **użytkować** realizowane przez PU funkcjonalności, **dostarczać dane**
- nazwa aktora – **rzeczownik lub wyrażenie rzeczownikowe w l.poj.**, np. Dział sprzedaży, Termin płatności, Bank, Recepcjonista

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, *Język UML 2.0 w modelowaniu systemów informatycznych*, Helion, Gliwice 2005

Zależności między elementami DPU

- Aktor może użytkować jeden lub więcej PU.
- Jeden PU może być użytkowany przez więcej niż jednego aktora.
- Każdy aktor musi być bezpośrednio powiązany z co najmniej jednym PU i każdy PU z co najmniej jednym PU.

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, *Język UML 2.0 w modelowaniu systemów informatycznych*, Helion, Gliwice 2005

Związki

- Przypadki użycia integruje się za pomocą związków:
  - asocjacji,
  - uogólnienia,
  - zależności,
  - realizacji.

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, *Język UML 2.0 w modelowaniu systemów informatycznych*, Helion, Gliwice 2005

Dokumentacja i scenariusze

- **scenariusz** – określony ciąg akcji dokumentujący zachowanie,
- scenariusze główne i alternatywne,
- precyzyjny opis funkcjonalności przypadku użycia dzięki scenariuszom głównym i alternatywnym

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, *Język UML 2.0 w modelowaniu systemów informatycznych*, Helion, Gliwice 2005

Funkcje DPU

- identyfikacją oraz dokumentacja wymagań,
- analiza dziedziny przedmiotowej,
- opracowanie projektu przyszłego systemu,
- zrozumiała platforma komunikacji,
- kontrakt co do zakresu i funkcjonalności systemu,
- podstawa testowania funkcji systemu na dalszych etapach.

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, *Język UML 2.0 w modelowaniu systemów informatycznych*, Helion, Gliwice 2005

Etapy tworzenia DPU

1. Identyfikacja aktorów,
2. Identyfikacja przypadków użycia,
3. Opracowanie związków,
4. Dokumentacja przypadków użycia.

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, *Język UML 2.0 w modelowaniu systemów informatycznych*, Helion, Gliwice 2005

Modelowanie architektury

- Architektura – sposób organizacji i integracji komponentów komputera lub systemu komputerowego.
- Architektura systemu - pokazuje jak powinien wyglądać system, by spełniał biznesowe potrzeby przedsiębiorstwa.
- a) logiczna
- b) fizyczna

R.A. Maksimchuk, E.J. Nalburg, *UML dla zwykłych śmiertelników*, Mikom, Warszawa 2007

Architektura logiczna

- część architektury, która nie zależy od stosowanej technologii,
- jest interpretacją tego jak powinna wyglądać architektura,
- nie pokazuje sposobu implementowania oprogramowania, jest metodą jego opisu.

R.A. Maksimchuk, E.J. Nalburg, *UML dla zwykłych śmiertelników*, Mikom, Warszawa 2007

Diagramy klas

- służą projektowaniu architektury logicznej,
- opisuje typy obiektów (elementów dziedziny przedmiotowej) w systemie i rodzaje statycznych relacji, które między nimi zachodzą

Obiekt i klasa

- obiekt jest instancją (wystąpieniem) klasy,
- klasa – uogólnienie zbioru obiektów mających takie same atrybuty i operacje, znaczenie i związki,
- klasa zawiera zestaw informacji istotnych z punktu widzenia kontekstu systemu.

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, *Język UML 2.0 w modelowaniu systemów informatycznych*, Helion, Gliwice 2005

Atrybuty i operacje

- wszystko co wiadomo o obiekcie jest opisane przez wartości jego atrybutów, a jego zachowanie w operacjach określających usługi, które oferuje,
- operacje wykonywane przez klasę lub na klasie przez inne klasy
- zainicjowanie operacji prowadzi do użytkowania/modyfikacji danych reprezentowanych przez wartości atrybutów.

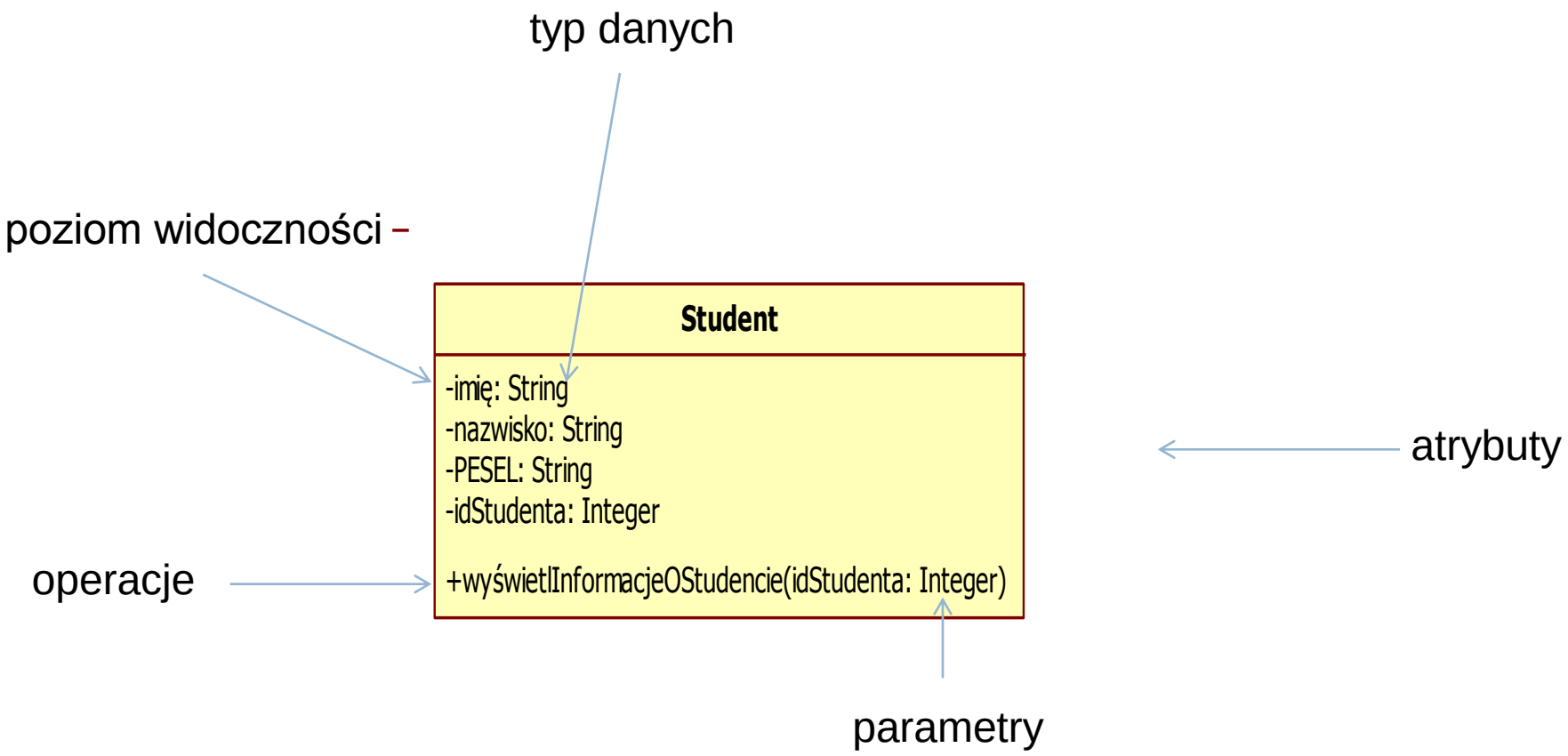
S. Wrycza, B. Marcinkowski, K. Wyrzykowski, *Język UML 2.0 w modelowaniu systemów informatycznych*, Helion, Gliwice 2005

Widoczność

- różne wymagania względem dostępu do operacji i atrybutów klas,
- możliwość pełniej ochrony danych czy ograniczenia dostępu do nich
- poziomy widoczności: publiczny, prywatny, chroniony, pakietowy
- reguła: atrybuty prywatne, operacje publiczne

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, *Język UML 2.0 w modelowaniu systemów informatycznych*, Helion, Gliwice 2005

Klasa

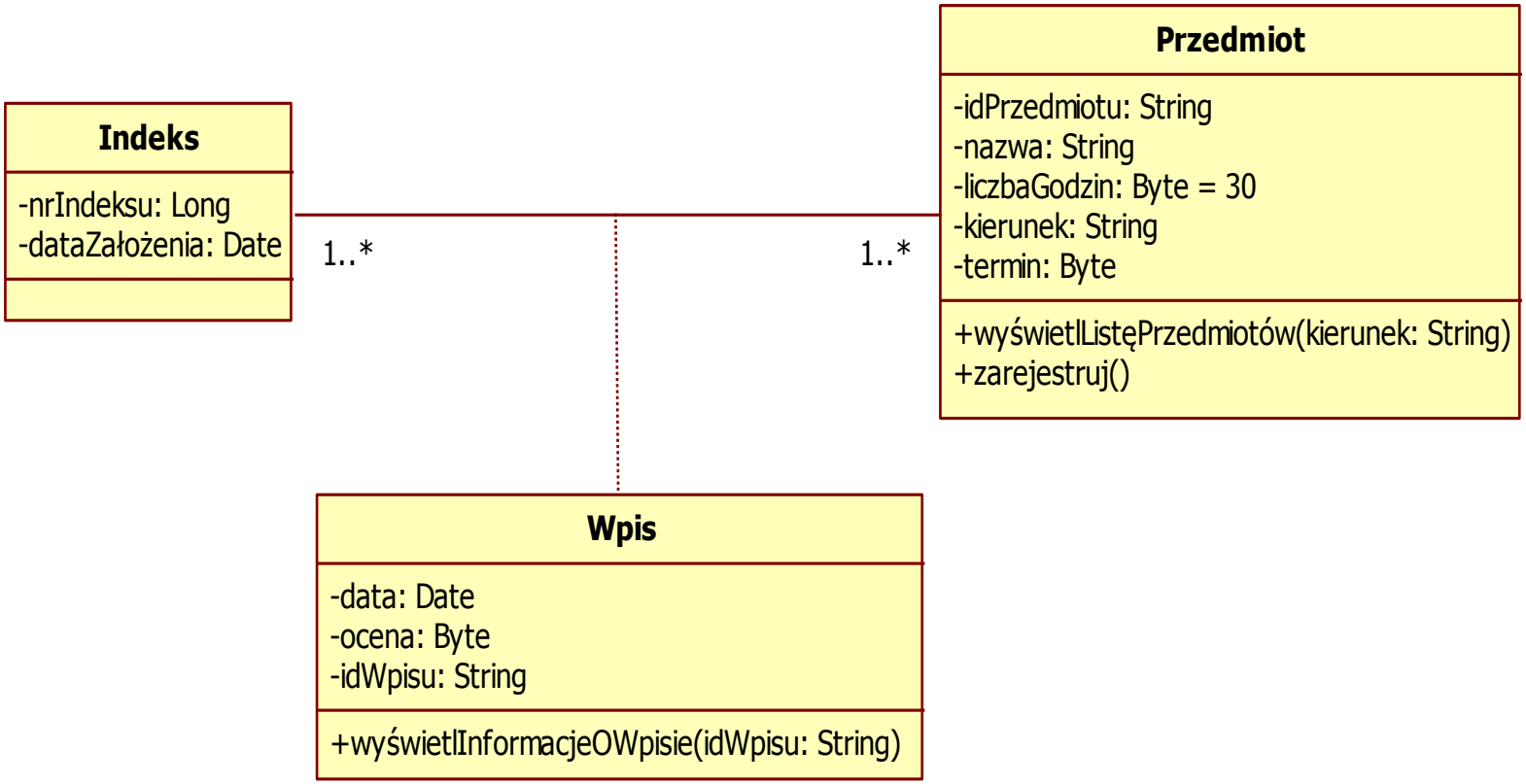

typ danych
poziom widoczności
Student
-imię: String
-nazwisko: String
-PESEL: String
-idStudenta: Integer
+wyświetlInformacjeOStudencie(idStudenta: Integer)
atrybuty
operacje
parametry

Asocjacje i klasy asocjacyjne

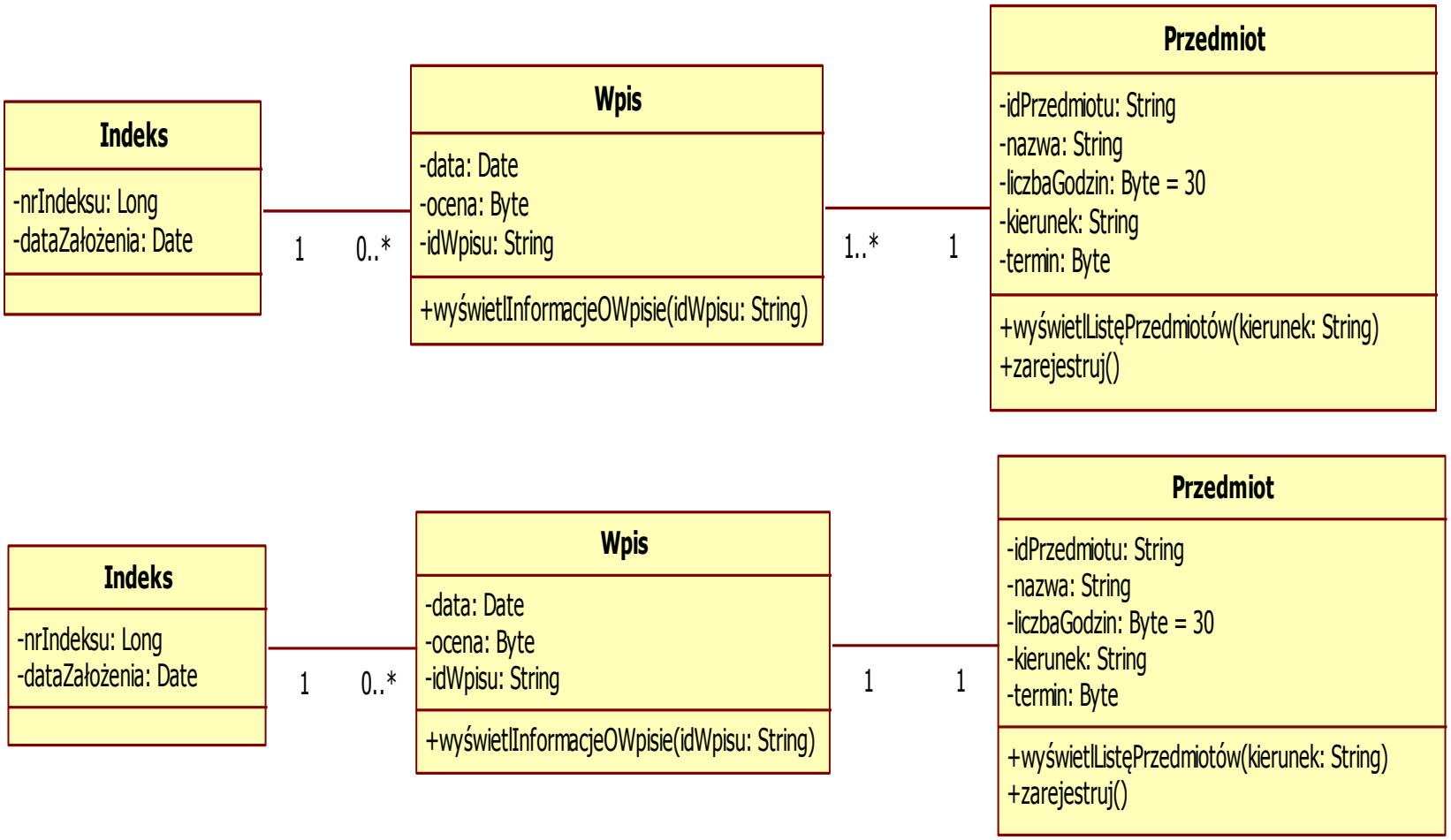
- asocjacje: nazwy, role, nawigacja, liczebność
- klasa asocjacyjna – pozwala na opisanie asocjacji w postaci klasy,
- każda asocjacja może mieć tylko jedną instancję klasy asocjacyjnej

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, *Język UML 2.0 w modelowaniu systemów informatycznych*, Helion, Gliwice 2005

Klasa asocjacyjna



Ominięcie klasy asocjacyjnej

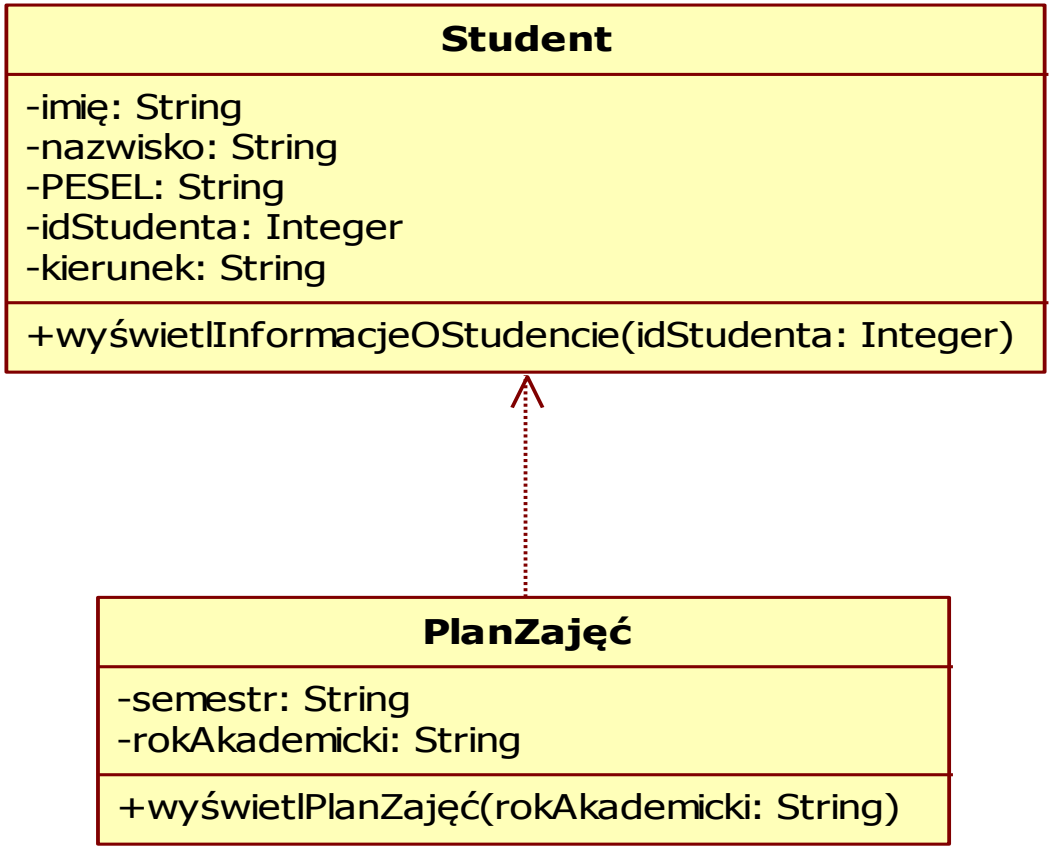


Zależności

- zależności – niezależna (docelowa) klasa wykorzystuje klasę zależną (źródłową); jeśli zmieni się docelowa, to zmieni się źródłowa; jeśli zmieni się źródłowa, to docelowa nie zmieni się

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, *Język UML 2.0 w modelowaniu systemów informatycznych*, Helion, Gliwice 2005

Zależności


Student
-imię: String
-nazwisko: String
-PESEL: String
-idStudenta: Integer
-kierunek: String
+wyświetlInformacjeOStudencie(idStudenta: Integer)
PlanZajęć
-semestr: String
-rokAkademicki: String
+wyświetlPlanZajęć(rokAkademicki: String)

Agregacja

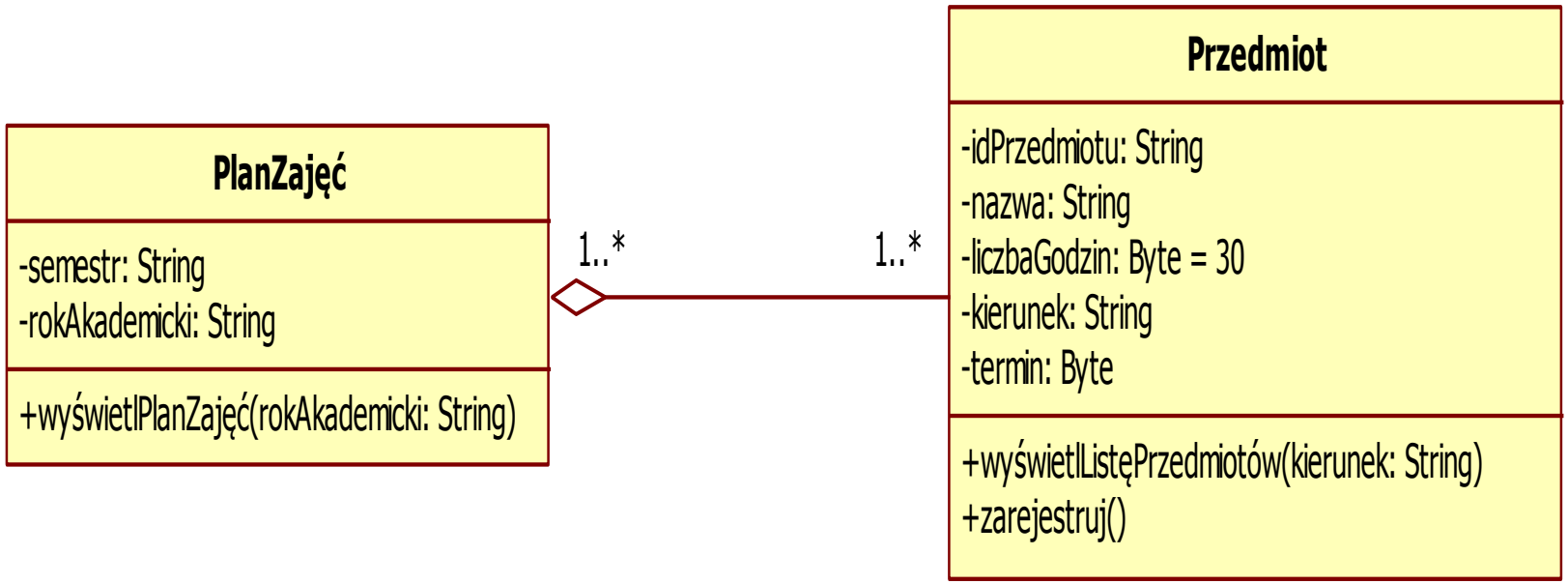
- związek całość-część między klasami

a) całkowita (kompozycja) – obiekty segmenty nie mogą samodzielnie funkcjonować bez agregatu (całości), wypełniony romb

b) częściowa – usunięcie agregatu nie powoduje usunięcia segmentów, pusty romb

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, *Język UML 2.0 w modelowaniu systemów informatycznych*, Helion, Gliwice 2005

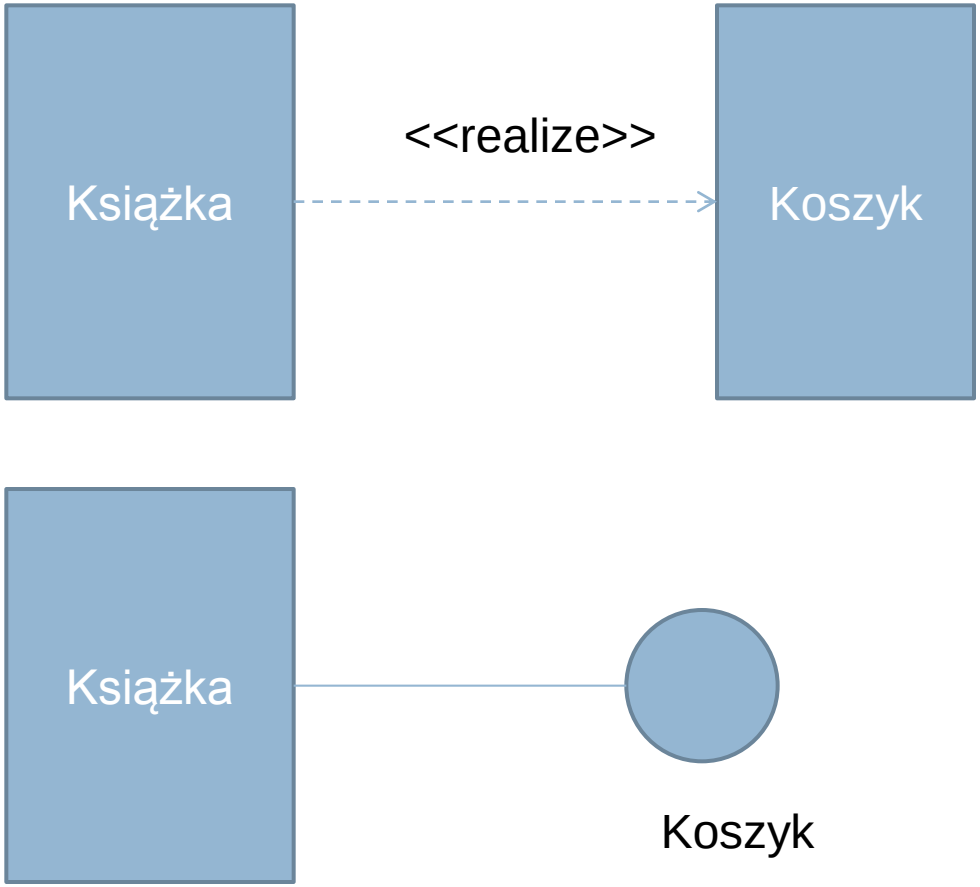
Agregacja


PlanZajęć
-semestr: String
-rokAkademicki: String
+wyświetlPlanZajęć(rokAkademicki: String)
1..*
1..*
Przedmiot
-idPrzedmiotu: String
-nazwa: String
-liczbaGodzin: Byte = 30
-kierunek: String
-termin: Byte
+wyświetlListęPrzedmiotów(kierunek: String)
+zarejestruj()

Interfejsy i realizacja

- jedna strona związku wskazuje na klasyfikator określający kontrakt, a druga wywiązanie się z niego,
- związek między interfejsem i klasą,
- interfejs - zestaw operacji, które określają pewien aspekt zachowania klasy i związany z tym zbiór operacji, które klasa udostępnia innym klasom,
- interfejs jest zbiorem operacji wykonywanych przez klasę, interfejs jest realizowany przez klasę

Interfejsy i realizacja



Uogólnienie i klasy abstrakcyjne

- klasa abstrakcyjna – nie ma konkretnych instancji obiektów, stanowi uogólnienie konkretnych obiektów znajdujących się na niższych poziomach hierarchii,
- nazwy klas abstrakcyjnych piszemy kursywą,
- klasy potomkowie dziedziczą atrybuty i operacje klas przodków

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, *Język UML 2.0 w modelowaniu systemów informatycznych*, Helion, Gliwice 2005

Proces tworzenia diagramu klas

1. Identyfikacja i nazwanie klas
2. Połączenie z wykorzystaniem asocjacji
3. Identyfikacja i nazwanie atrybutów i operacji
4. Wyspecyfikowanie asocjacji
5. Opracowanie innych związków
6. Opcjonalnie: opracowanie diagramów obiektów

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, *Język UML 2.0 w modelowaniu systemów informatycznych*, Helion, Gliwice 2005

Modelowanie analityczne

- model analityczny – etap pośredni między wymaganiami a projektem, pozwala na dekompozycję niebanalnych przypadków użycia, sprecyzowanie środowiska, w którym system będzie pracował
- zasada uproszczonego modelowania
- pominięcie środowiska programistycznego, tylko wymagania funkcjonalne
- element opcjonalny i pośredni

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, Język UML 2.0 w modelowaniu systemów informatycznych, Helion, Gliwice 2005, s. 343-345

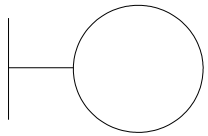
Stereotyp

- mechanizm rozszerzalności wzbogacający zbiór standardowych kategorii modelowania uml
- wprowadza się za jego pomocą nowe elementy, które bazują na już istniejących
- duża liczba stereotypów standardowych, już zdefiniowanych

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, Język UML 2.0 w modelowaniu systemów informatycznych, Helion, Gliwice 2005, s. 27-28

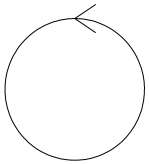
Klasy analityczne

- w realizacji przypadków użycia biorą udział klasy:



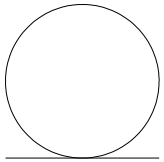
boundary

graniczna



control

sterująca



entity

przechowująca

Klasa graniczna

- zazwyczaj pośredniczy między aktorem i systemem
- jest na styku systemu, oddziela go od otoczenia
- reprezentuje formatki ekranowe, strony www, terminale, interfejsy
- może być interfejsem użytkownika, systemu, urządzenia

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, Język UML 2.0 w modelowaniu systemów informatycznych, Helion, Gliwice 2005, s. 346

Klasa sterująca

- reprezentuje procesy, które zawierają logikę aplikacji określając przetwarzanie informacji
- pośredniczy między wszystkimi rodzajami klas analitycznych
- umożliwia koordynację, operowanie na innych klasach i sterowanie nimi
- na etapie projektowym jej obiekty mogą zostać przekształcone na samodzielne klasy lub ich operacje

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, Język UML 2.0 w modelowaniu systemów informatycznych, Helion, Gliwice 2005, s. 346

Klasa przechowująca

- reprezentuje przechowywane dane
- nie może samodzielnie inicjować interakcji
- związana jest z wieloma przypadkami użycia
- na etapie projektowym może być implementowana jako samodzielna klasa lub jako atrybut

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, Język UML 2.0 w modelowaniu systemów informatycznych, Helion, Gliwice 2005, s. 346

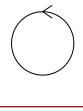
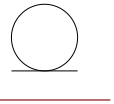
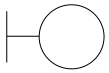
Klasy analityczne

- w każdej relacji aktora z systemem pośredniczy obiekt klasy boundary
- na jeden przypadek użycia przypada zazwyczaj jedna klasa sterująca

Kolejność tworzenia modelu analitycznego

1. diagram przypadków użycia
2. diagram klas analitycznych
3. diagram sekwencji na klasach analitycznych

Dozwolone połączenia

Może łączyć się z	Aktor 	Graniczna 	Sterująca 	Przechowująca 
Aktor 	-	+	-	-
Graniczna 	+	-	+	-
Sterująca 	-	+	+	+
Przechowująca 	-	-	+	-

Diagramy interakcji

- interakcja – wymiana bodźców, impulsów i komunikatów między instancjami klasyfikatorów w systemie,
- zazwyczaj dotyczą jednego przypadku użycia,
- pokazują jak współpracują ze sobą obiekty w systemie,
- zawierają obiekty i wymieniane komunikaty,
- diagramy sekwencji i komunikacji

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, Język UML 2.0 w modelowaniu systemów informatycznych, Helion, Gliwice 2005

Diagramy sekwencji

- pokazuje interakcje między obiektami w postaci sekwencji komunikatów, które między sobą wymieniają,
- wyznaczone zostają miejsce i kolejność wykonania operacji

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, Język UML 2.0 w modelowaniu systemów informatycznych, Helion, Gliwice 2005

Rodzaje diagramów sekwencji

- konceptualny – szybkie i ogólne naszkicowanie zakresu i zawartości i interakcji;
- implementacyjny – wyższy poziom precyzji, obejmuje przepływy główne i alternatywne, przekazywane programistom jako dokumentację projektową;
- wystąpieniowy – wystąpienie diagramu w odniesieniu do konkretnego scenariusza

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, Język UML 2.0 w modelowaniu systemów informatycznych, Helion, Gliwice 2005

Podstawowe elementy

- obiekt,
- linia życia – powiązana z konkretnym obiektem, pokazuje długość życia tego obiektu,
- komunikat – specyfikacja wymiany informacji między obiektami, zawierająca polecenie wykonania określonej operacji, umieszcza się je pomiędzy liniami życia

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, Język UML 2.0 w modelowaniu systemów informatycznych, Helion, Gliwice 2005

Komunikaty

- porządkowane są według kolejności ich występowania ,
- im później występuje wysłanie komunikatu, tym niżej umieszczony jest na diagramie,
- można je numerować,
- można wysyłać komunikaty to obiektów nie znajdujących się w bezpośrednim sąsiedztwie z nadawcą

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, Język UML 2.0 w modelowaniu systemów informatycznych, Helion, Gliwice 2005

Linia życia i aktywność

- linia życia reprezentuje okres życia obiektu, w niektórych momentach życia obiekt jest w stanie czuwania, a w innych jest aktywowany poprzez komunikaty, przejmując sterowanie interakcją i podejmując zlecone działania,
- po wykonaniu zleconych operacji i wysłaniu komunikatów wynikowych obiekt przechodzi z powrotem w stan czuwania

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, Język UML 2.0 w modelowaniu systemów informatycznych, Helion, Gliwice 2005

Ośrodek sterowania

- ośrodek sterowania – specyfikacja wykonywania czynności w ramach interakcji; przyjmuje postać prostokąta na linii życia obiektu,
- inicjowany aktywacją a przejście w stan czuwania dezaktywacją,
- na jednej linii życia może być kilka niezależnych ośrodków sterowania

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, Język UML 2.0 w modelowaniu systemów informatycznych, Helion, Gliwice 2005

Rodzaje komunikatów

- komunikaty mogą różnić się pod względem funkcjonalności,
- specyfika rodzajów komunikatów wyrażana jest graficznie
- komunikaty kompletne – znany jest nadawca i odbiorca
- komunikaty niekompletne – jedna z instancji nie jest znana

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, Język UML 2.0 w modelowaniu systemów informatycznych, Helion, Gliwice 2005

Komunikaty kompletne

- synchroniczny ,
- asynchroniczny,
- zwrotny – nie należy go nadużywać,
- opcjonalny,
- oczekujący

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, Język UML 2.0 w modelowaniu systemów informatycznych, Helion, Gliwice 2005

Komunikaty niekompletne

- utracony – odbiorca jest nieznany; stosowany w modelowaniu złożonych interakcji na wczesnych etapach kiedy nie można jednoznacznie określić odbiorcy komunikatu przesyłanego między fragmentami interakcji,
- znaleziony – nadawca jest nieznany; wysyłany spoza danego diagramu (impuls typu dym, hałas, ogień)

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, Język UML 2.0 w modelowaniu systemów informatycznych, Helion, Gliwice 2005

Tworzenie i niszczenie

- poszczególne instancje mogą za pośrednictwem operacji tworzyć i niszczyć obiekty,
- obiekt utworzony w wyniku przesłania komunikatu <<create>> i umieszczane poniżej pierwotnie istniejących obiektów,
- obiekt zniszczony wraz z odebraniem komunikatu <<destroy>> oraz oznaczony zdarzeniem niszczącym X

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, Język UML 2.0 w modelowaniu systemów informatycznych, Helion, Gliwice 2005

Samowywołanie

- oznacza sytuację, w której dana instancja wywołuje własną operację;
- zagnieżdżenie – określa logicznie powiązany ciąg komunikatów wywołujących się wzajemnie w ustalonej kolejności

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, Język UML 2.0 w modelowaniu systemów informatycznych, Helion, Gliwice 2005

Fragment wyodrębniony

- wyodrębniony obszar interakcji charakteryzujący się specyficznymi właściwościami określonymi przez operator interakcji,
- umożliwia bardziej precyzyjne pokazanie istoty interakcji,
- obramowanie z nagłówkiem,
- składa się z jednego lub wielu operandów interakcji

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, Język UML 2.0 w modelowaniu systemów informatycznych, Helion, Gliwice 2005

Operatory interakcji

- alt – alternatywa,
- opt – opcja,
- loop – pętla,
- par – współbieżność,
- assert – formuła,
- strict – ścisłe uporządkowanie,
- seq – słabe uporządkowanie,
- neg – funkcjonalność nieprawidłowa

Operandy

- operandy interakcji – subfragmenty fragmentu wyodrębnionego, oddzielone seperatorami,
- fragment wyodrębniony składa się z jednego lub wielu operandów interakcji,
- opt, loop, neg – tylko jeden operand,
- alt – może być wiele, ale realizowany tylko jeden w zależności od spełnienia warunku,
- alt, opt – warunek w obszarze operandu
- par – wszystkie operandy wykonywane jednocześnie,

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, Język UML 2.0 w modelowaniu systemów informatycznych, Helion, Gliwice 2005

Diagram komunikacji (kooperacji/kolaboracji)

- **Diagram sekwencji** – przebieg interakcji w czasie; uporządkowanie w czasie
- **Diagram komunikacji** – otoczenie i organizacja obiektów biorących udział w interakcji; uporządkowanie w przestrzeni
- jest rozszerzeniem diagramu obiektów,
- poza asocjacjami między obiektami pokazuje również wymieniane między nimi komunikaty

J. Schmuller, UML dla każdego, Helion, Gliwice 2003

Diagramy komunikacji

- diagramy komunikacji i diagramy sekwencji są **izomorficzne**, to znaczy jednoznacznie wzajemnie przekształcalne

S. Wrycza, B. Marcinkowski, K. Wyrzykowski, Język UML 2.0 w modelowaniu systemów informatycznych, Helion, Gliwice 2005

Komunikaty

- strzałki umieszczane nad liniami powiązań zwrócone w stronę odbiorcy komunikatu,
- muszą być numerowane,
- musi być wskazany rodzaj komunikatu

Źródła

- Fowler M., Scott K., *UML w kropelce*, LTP, Warszawa, 2002,
- Maksimchuk R.A., Naiburg E.J., *UML dla zwykłych śmiertelników*, Mikom, Warszawa, 2007,
- Szejko S. (red.), *Metody wytwarzania oprogramowania,* Mikom, Warszawa, 2002,
- Schmuller J., *UML dla każdego*, Helion, Gliwice, 2003,
- Wrycza S., Marcinkowski B., Wyrzykowski K., *Język UML 2.0 w modelowaniu systemów informatycznych*, Helion, Warszawa 2005.