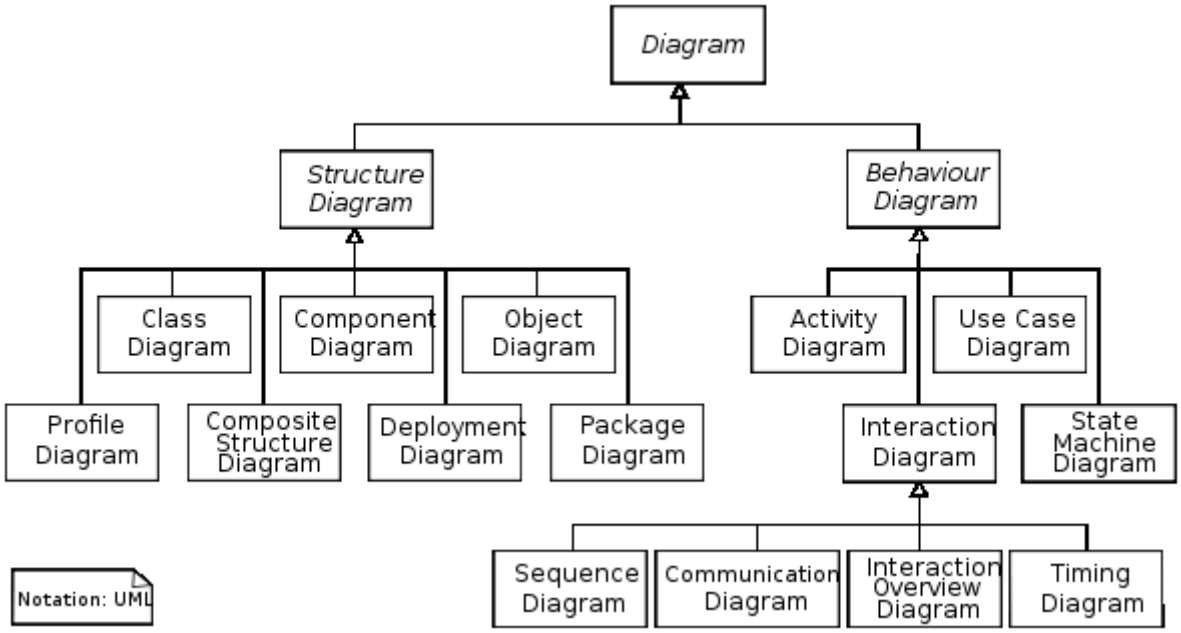
Projektowanie obiektowe oprogramowania
Wykład 1/2 - UML
Wiktor Zychla 2012

1 Wprowadzenie



- Diagramy struktury i diagramy zachowań (dynamiki)

2 Diagramy klas

2.1 Klasy i składowe

- Składowa prywatna, publiczna, chroniona, stała, statyczna, kolekcja, atrybut pochodny
- Metoda prywatna, publiczna, chroniona, internal, abstrakcyjna, statyczna, konstruktor, parametry

2.2 Asocjacje

- Zależności (strzałka przerywana) – brak informacji o rodzaju zależności, może być:
    - Tworzy
    - Wykorzystuje (zmienna lokalna)
    - Wykorzystuje (parametr metody)
    - Nadklasa lub interfejs
- Asocjacja: atrybut wpisany vs asocjacja – kiedy używać
- Kolekcja, generowanie kodu
- Agregacja vs kompozycja
    - Instancja reprezentująca część może należeć tylko do jednej instancji złożonej

    - Część musi zawsze należeć do jakiejś całości
    - Czas życia części jest powiązany z czasem życia całości
- Klasa asocjacyjna – modelowanie charakteru asocjacji (np. wiele od wielu) (tylko dwa, wiele „jak zrobić?")

2.3 Dziedzieczenie

- Realizacja – implementacja interfejsu
- Generalizacja, specjalizacja – dziedziczenie (tylko w zależności od kierunku)

3 Diagramy obiektów

- Migawka systemu
- Advanced / Instance Classifier
- Advanced / Set Run State

4 Diagramy stanów

- Stany i przejścia (akcje) – stany to bloczki, a akcje to strzałki
- Stany – nazwane rzeczownikowo/przymiotnikowo (oczekiwanie/przetwarzanie, oczekujący/aktywny/przydzielony)
- Akcje – nie nazywają się
- Przykładowy schemat
    - Stany – oczekiwanie, przetwarzanie
    - Wariant – nazwany
    - Zrównoleglanie – wysyłanie, fakturowanie
    - Stan kompozytowy

5 Diagramy czynności

- Czynności vs akcje
    - Czynności – długotrwałe, podzielne, ogólne
    - Akcje – krótkotrwałe, niepodzielne, szczegółowe – nazwane czasownikowo (wprowadź/wybierz/zatwierdź/wydrukuj/aktualizuj/weryfikuj)
- Odwrotność diagramu stanów jeśli chodzi o semantykę bloków vs strzałek – tam bloczek = stan, strzałka = akcja; tu bloczek = akcja, strzałka – wyznacza następstwo akcji
    - Sygnały (zdarzenia) – wyślij, odbierz
    - Wariant – „if"
    - Zdarzenia – send/receive
    - Regiony – na przykład „przerywalny", pojawia się zdarzenie „przerwij", anulowanie
    - Partycje – podział na aktorów

6 Diagramy komunikacji

- Obiekty, ustalonych typów
- Przepływają między nimi komunikacji w ustalonej kolejności

- Komunikaty można interpretować jako wywoływane metody
- Przykładowy kod:

```
public B b = new B();
public C c = new C();
public D d = new D();

public class A {
	public void fooA() {
		b.fooB();
		c.fooC();
	}
}
public class B {
	public void fooB() {
		c.fooC();
	}
}
public class C {
	public void fooC() {
		b.fooB();
		d.fooD();
	}
}
public class Actor {
	public void XXXX() {
		a.fooA();
	}
}
```

7 Diagram komunikacji vs diagram sekwencji

	plusy	Minusy
diagram sekwencji	• Wyraźna kolejność komunikatów • Dużo szczegółowych opcji notacji	• Ograniczone miejsce (diagram rozwija się od lewej do prawej
diagram komunikacji	• Dobrze organizuje przestrzeń („w ogóle jej nie organizuje”)	• Mało opcji notacji • Mniej czytelny

- Izomorfizm diagramów komunikacji i diagramów sekwencji – przykład

8 Diagramy sekwencji

- Linie życia, paski aktywacji/ośrodki sterowania (execution specification)
- Typy obiektów
  - Boundary – widok
  - Control – kontroler
  - Entity – model
- Związek między diagramem sekwencji a diagramem klas – ustalanie typu obiektu
- Komunikat – wartość zwrotna
  `wartość = komunikat( p1:P1, p2:P2, … ) : typ`
- lub przerywana strzałka zwrotna (EA – niekoniecznie)
- Singleton – jedynka w rogu, metoda statyczna – stereotyp „class”, „metaclass”
- Komunikat odnaleziony – „od nikogo”

- Komunikat do „this”
- Create/destroy
- Ramki, można zagnieżdżać
    - Loop – pętla
    - Alt – if-then-else
    - Opt – if
    - Par - współbieżność
    - Ref – odwołanie do innej, nazwanej ramki
    - Sd – nazwana ramka