

Refining the Requirements Model

Reuse Pada Pengembangan Software

- Pengembangan perangkat lunak telah berkonsentrasi pada menciptakan solusi baru
- Baru-baru ini, penekanan telah bergeser
- Banyak perangkat lunak kini dirakit dari komponen yang sudah ada
- Menggunakan kembali komponen dapat menghemat uang, waktu dan usaha

Reuse Pada Pengembangan Software

Mencapai reuse masih sulit

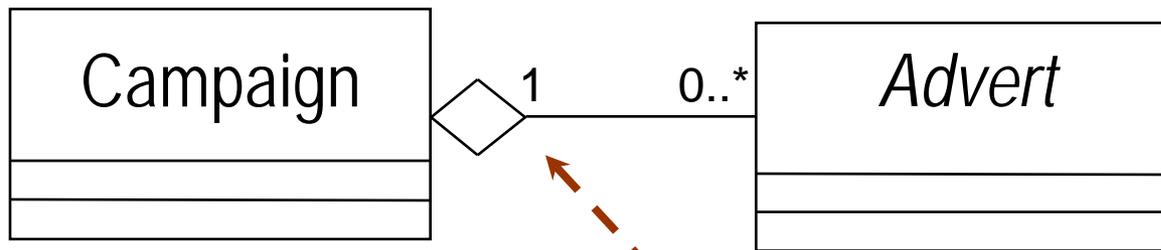
- Reuse tidak selalu tepat - tidak bisa menganggap komponen yang ada memenuhi kebutuhan baru
- Model organisasi membuat sulit untuk mengidentifikasi komponen-komponen yang sesuai
- NIH (Not-Invented-Here) sindrom
- Persyaratan dan desain yang lebih sulit dari kode untuk menggunakan kembali

Reuse: Kontribusi OO

- Encapsulation membuat komponen lebih mudah untuk digunakan pada sistem yang awalnya tidak mereka rancang
- Agregasi dan komposisi dapat digunakan untuk merangkum komponen
- Generalisasi memungkinkan penciptaan kelas khusus yang baru bila diperlukan

Aggregasi dan Komposisi

- Jenis khusus dari asosiasi, baik kadang-kadang disebut whole-part
- Sebuah kampanye terdiri dari beberapa iklan



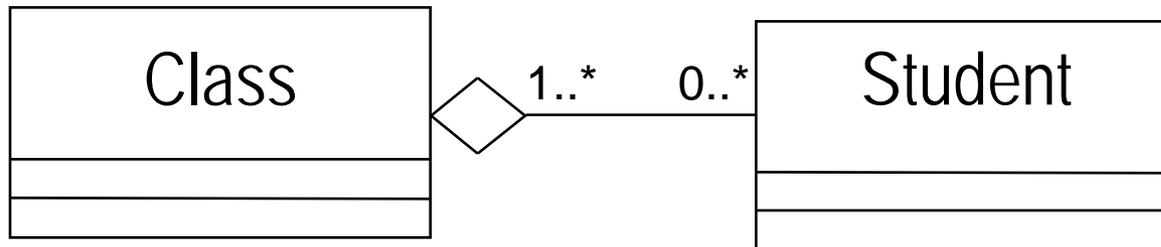
*Unfilled diamond
signifies aggregation*

Aggregasi dan Komposisi

- Aggregation whole-part relationship yang esensial
- Semantics bisa sangat tidak tepat
- Composition 'stronger':
 - Setiap bagian mungkin milik satu bagian yang lebih besar pada satu waktu tertentu
 - Jika bagian yang besar itu rusak maka akan merusak semuanya

Aggregasi dan Komposisi

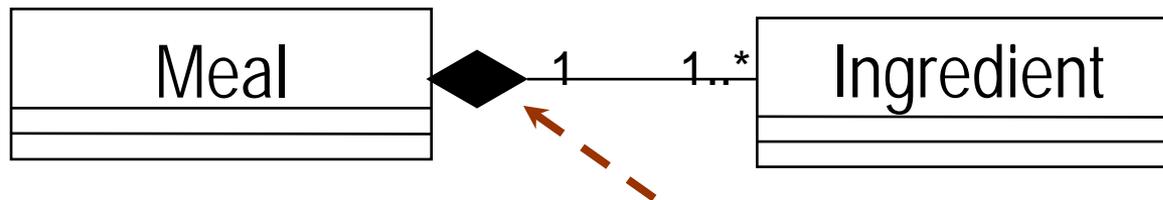
- contoh:



- - Siswa bisa dalam beberapa kelas
 - Jika kelas dibatalkan, siswa masih tetap ada!

Aggregasi dan Komposisi

- Contoh lainnya:



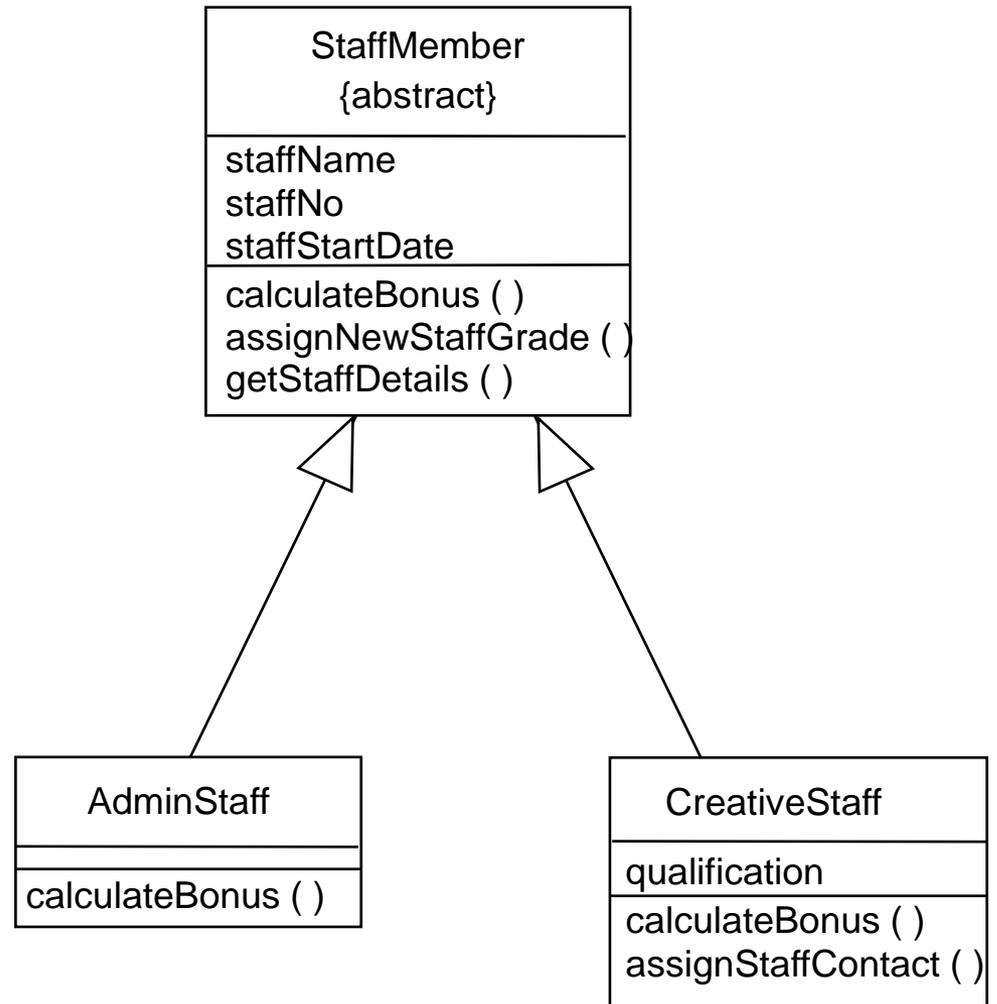
Filled diamond signifies composition

- ini (Mungkin) Komposisi:
 - Bahan-bahan makanan ini berada dalam satu menu makanan
 - Jika makan malam ini jatuh maka kita akan kehilangan seluruh bahan makanan tersebut

Menambahkan Struktur Generalisasi

- Menambah struktur generalisasi ketika :
 - Dua kelas yang sama di sebagian besar rincian, tetapi berbeda dalam beberapa hal
 - Mungkin berbeda:
 - Dalam perilaku (operasi atau metode)
 - Dalam data (atribut)
 - Dalam asosiasi dengan kelas lain

Menambahkan Struktur



Software Development Patterns

A pattern:

- “describes a problem which occurs over and over again in our environment, and then describes the core of a solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.”

Alexander et al. (1977)

Software Development Patterns

A pattern:

- “menjelaskan masalah yang terjadi berulang-ulang di lingkungan kita, dan kemudian menjelaskan inti dari solusi untuk masalah tersebut, sedemikian rupa bahwa Anda dapat menggunakan solusi ini jutaan kali, tanpa pernah melakukannya dengan cara yang sama dua kali.”

Alexander et al. (1977)

Software Development Patterns

- pattern memiliki:
 - *A context* = satu set keadaan atau prasyarat untuk masalah terjadi
 - *Forces* = masalah yang harus diatasi
 - Sebuah konfigurasi perangkat lunak yang memecahkan masalah

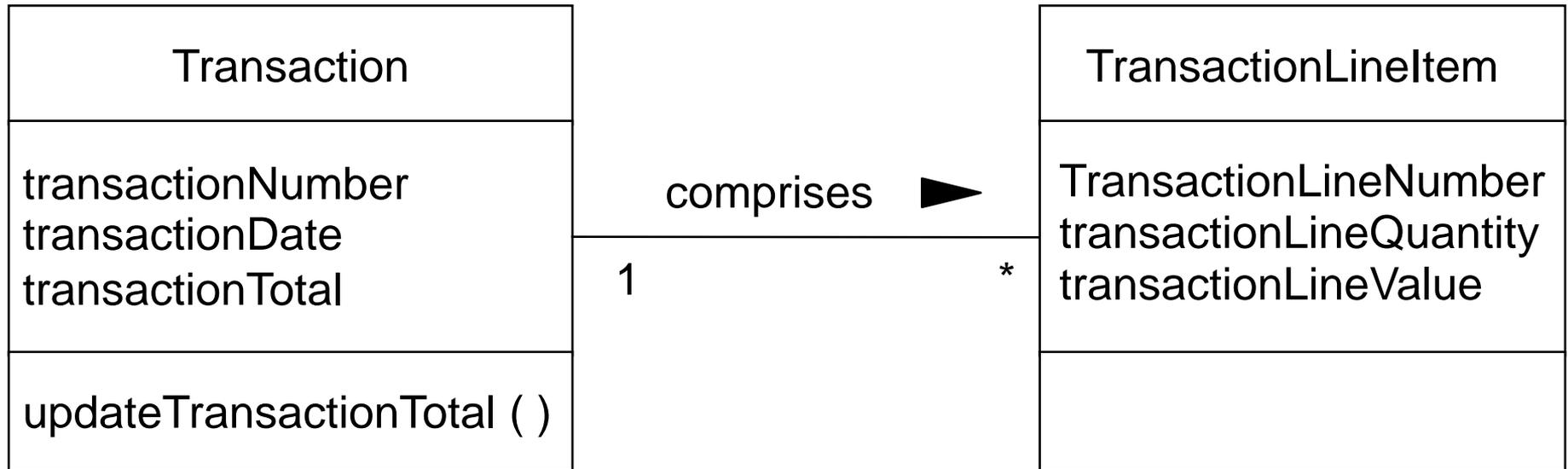
Software Development Patterns

- Pola yang ditemukan pada banyak titik dalam siklus hidup pengembangan sistem:
 - Pola analisis adalah kelompok konsep berguna dalam persyaratan pemodelan
 - Pola arsitektur menggambarkan struktur komponen utama dari sistem perangkat lunak
 - Pola Desain menggambarkan struktur dan interaksi komponen perangkat lunak yang lebih kecil

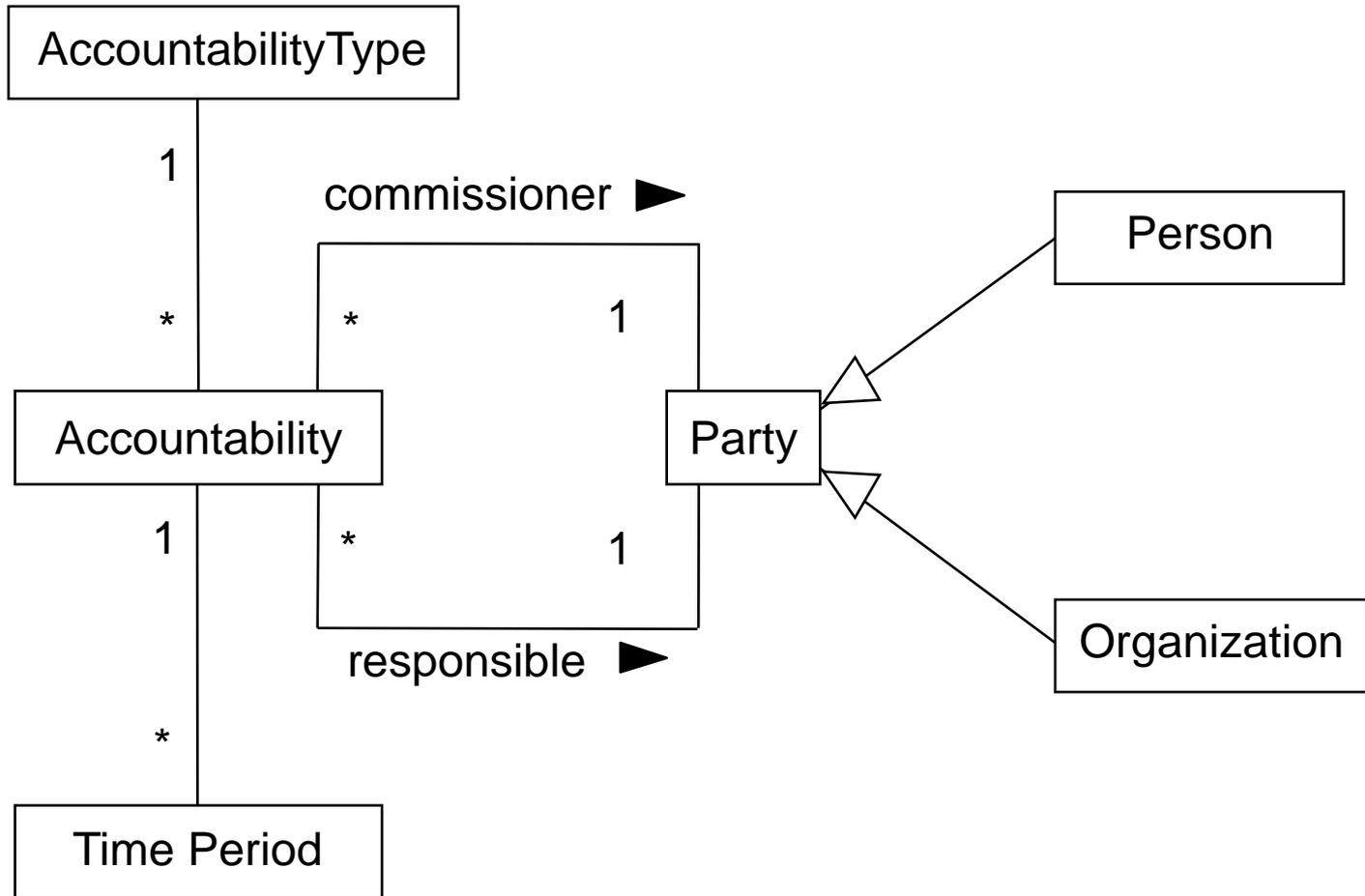
Software Development Patterns

- Pola telah diterapkan secara luas dalam pengembangan perangkat lunak:
 - Pola Organisasi menggambarkan struktur, peran dan interaksi dalam organisasi pengembangan perangkat lunak itu sendiri
- Anti-pola praktik dokumen kurang baik
- Mushroom Management adalah sebuah organisasi anti-pola

Simple Analysis Pattern



Accountability Analysis Pattern



Referensi

- Bennett, McRobb and Farmer (2005)
- Ambler (2003)
- Cheesman and Daniels (2001)
- Coad (1997)
- Fowler (1997)

(For full bibliographic details, see Bennett, McRobb and Farmer)