MODEL ARCHITECTURAL VIEW

Ama Fariza, S.Kom, M.Kom

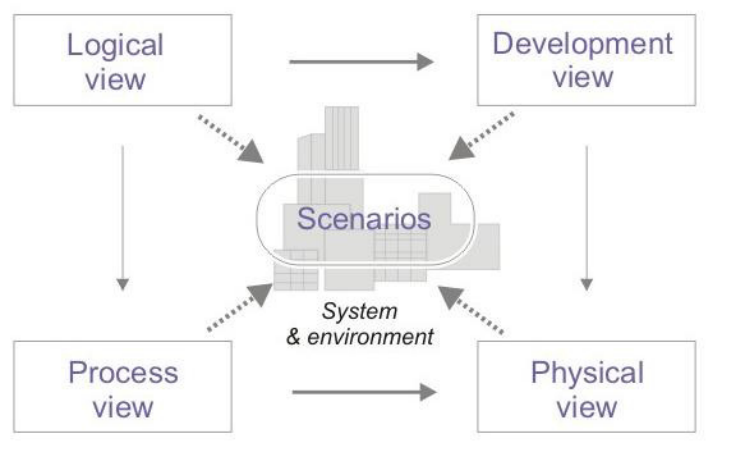
MATERI

- Model architectural view
- Dimensi dari view integration
- Jenis view
- Pedoman yang fleksibel untuk desain class
- Pedoman yang fleksibel untuk desain perilaku

MODEL ARCHITECTURAL VIEW #1

3

- View digunakan untuk menggambarkan sistem dari sudut pandang stakeholder misalnya end-user, developer dan project manager.
- Ada 4 view dari model yaitu logika, pengembangan, proses dan fisik



MODEL ARCHITECTURAL VIEW #2

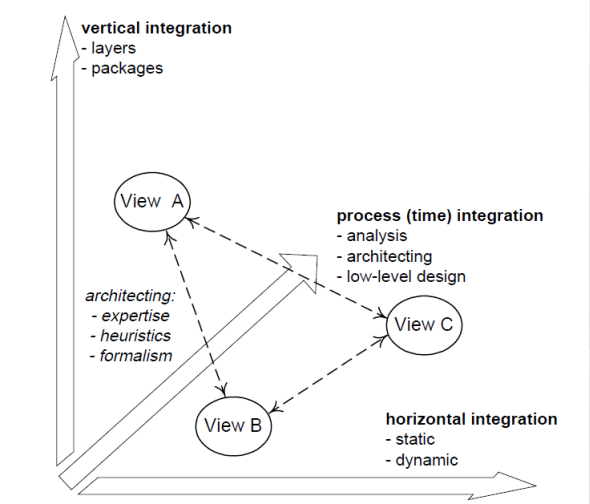
4

- ***Logical view :*** menyangkut fungsionalitas sistem yang disediakan untuk end-user. Diagram UML yang digunakan adalah **diagram class, diagram sequence**
- ***Development view:*** menggambarkan sistem dari perspektif programmer dan menyangkut manajemen software. Disebut juga view implementasi. Diagram UML yang digunakan adalah **diagram Komponen**, termasuk **diagram Package**
- ***Process view:*** berhubungan dengan aspek dinamis dari sistem, menjelaskan proses sistem dan bagaimana berkomunikasi dan fokus pada perilaku sistem pada saat eksekusi. Proses view menyangkut konkurensi, distribusi, integrator, performansi, skalabilitas dll. Diagram UML yang menyatakan proses view adalah **diagram aktifitas**
- ***Physical view:*** menggambarkan sistem dari sisi engineer. Menyangkut topologi dari komponen software pada layer fisik, sebagaimana koneksi fisik antar komponen. Disebut juga **deployment view.** Diagram UML yang digunakan adalah **diagram deployment**
- ***Scenario:*** deskripsi arsitektur diilustasikan menggunakan kumpulan dari use case atau skenario yang menjadi view kelima. Skenario menggambarkan urutan interaksi antar obyek dan antar proses. Digunakan untuk mengenali elemen arsitektur dan menjelaskan dan validasi desain arsitektur. Juga berlaku sebagai titik awal testing dari prototipe arsitektur. Diagram UML yang digunakan untuk menyatakan **scenario view** adalah **diagram case**

DIMENSI DARI VIEW INTEGRATION #1

5

- View dibagi dalam 3 dimensi utama yaitu **horisontal, vertikal dan proses.**
- Dimensi vertikal dan horisontal juga disebut sebagai **layer dan partisi**, sedangkan dimensi proses sebagai **life-cycle** (time)



COMPANY

DIMENSI DARI VIEW INTEGRATION #2
Dimensi vertical view

6

- Mencerminkan **dekomposisi sistem**
- **Layer tertinggi** menunjukkan sistem dalam bentuk abstrak, layer terendah menunjukkan sistem lebih detail
- Setiap layer menyatakan sistem yang lengkap, meskipun tidak dikerjakan hanya dengan satu diagram
- Satu layer tidak menyatakan relasi yang sama (dalam dan antar diagram) dengan layer lain
- Dalam UML, dekomposisi sistem dicapai melalui **diagram class/obyek** dan dihubungkan dengan paket, diagram lain misalnya **diagram state** tidak menyatakan dekomposisi sistem, tetapi masih digunakan setiap layer untuk mengulang konstruksi pemodelan yang sama dalam bentuk detail dan abstrak

COMPANY

DIMENSI DARI VIEW INTEGRATION #3
Dimensi vertical view

7

- Begitu sistem didekomposisi dalam subsistem, layer tambahan dari subsistem tidak menyatakan keseluruhan sistem, tetapi bagian dari subsistem → Artinya subsistem lain harus dinyatakan dengan cara yang sama oleh layer dan sub layer masing2 yang menyatakan sistem yang lengkap kembali
- Bisa terjadi ketidaksesuaian antara (sub) layer jika partisi dari layer tersebut tidak terpisah penuh dan tidak konsisten diaplikasikan
- Dalam beberapa kasus, subsistem tidak perlu perbaikan lebih lanjut karena sudah cukup detail dimana subsistem (layer) masih diperlukan perbaikan lanjutan → jika hal ini terjadi, perbaikan subsistem sudah cukup juga digunakan (misalnya mengakses, dipanggil dll) oleh semua layer lebih rendah lain sehingga keseluruhan masih menyatakan sistem yang lengkap.
- Hal ini juga menunjukkan bahwa level detail dicapai oleh layering yang sering tidak jelas didefinisikan dan bervariasi dalam setiap layer
- Aspek penting adalah apakah penjumlahan bagian2nya menyatakan gambaran lengkap dan pembagian logika (partisi) dari sistem ke dalam subsistem dilakukan konsisten pada semua level abstraksi

COMPANY

DIMENSI DARI VIEW INTEGRATION #4
Dimensi vertical view

8

- Desain fisik seharusnya lanjutan dari desain logika
- Hal ini bukan berarti bahwa layer fisik harus menggunakan jenis view yang sama atau kumpulan fitur yang sama dari layer logika
  - Contohnya desain fisik (dibandingkan desain logika) hanya menggunakan elemen pemodelan yang dapat langsung diimplementasikan, sehingga jika layer sistem secara fisik digambarkan dalam diagram class dan akan diimplementasikan dalam C (atau bahasa selain OO) daripada beberapa elemen dari diagram class, seperti pewarisan, seharusnya tidak digunakan dalam desain fisik lagi
- Partisi ke dalam subsistem masih valid dan seharusnya sama baik view logika dan view fisik
- Sehingga layer tidak harus menggunakan jenis view yang sama (misalnya diagram class)

COMPANY

DIMENSI DARI VIEW INTEGRATION #5
Dimensi horizontal view

- Batasan dari integrasi vertikal adalal kumpulan view digunakan setiap layer harus menyatakan seluruh sistem secara lengkap, meskipun setiap layer mungkin dinyatakan dengan kumpulan view yang berbeda
- Dengan menggunakan jenis view berbeda atau dekomposisi berbeda (meskiput tidak direkomendasi), sehingga dalam layer horisontal, view tidak dibutuhkan untuk memodelkan sistem yang lengkap (atau subsistem tergantung dari layer)

DIMENSI DARI VIEW INTEGRATION #6
Dimensi horizontal view

- View horisontal dibagi dalam view statis dan dinamis
- Perbedaan keduanya dihugungkan dengan waktu eksekusi yang terduga
- View dinamis menyatakan sistem (atau partisi) pada titik waktu tertentu atau waktu interval
  - Contohnya, diagram obyek menunjukkan obyek yang terdapat pada waktu tertentu; diagram sequence menunjukkan keterganungan antara beberapa obyek dalam suatu interval waktu
  - View dinamis biasanya menyatakan contoh dari state atau interaksi sistem dan komponen sistem selama eksekusi
- Diagram class adalah representasi statis dari semua ketergantungan sistem dan komponen sistem selama eksekusi
- Kategori statis menyatakan aspek yang lebih umum dari perilaku sistem dan konstruksinya selalu dapat diaplikasikan

DIMENSI DARI VIEW INTEGRATION #7
Integrasi process view

- Bentuk integrasi ini sering diabaikan meskipun sangat penting
- Integrasi sepanjang waktu (atau integrasi proses) menyatakan integrasi artifak produk selama siklus
- Aktifitas ini juga sering disebut sebagai versi kontrol atau manajemen konfigurasi tetapi juga merupakan dimensi lain dari permasalahan view integration
- Jika kita mempunyai dua pendekatan desain alternatif, setiap desain mempunyai kelebihan dan kekurangan yang unik. Dengan jelas, kita yakin bahwa masing2 alternatif menyatakan permasalahan yang sama dan mengerjakan dengan lengkap. Sehingga masuk akal untuk membandingkan dua pendekatan desain tersebut untuk menjamin kelengkapannya

JENIS VIEW

- **Diagrammatic view**
  - Diagram Class/Obyek
  - Diagram sequence
  - Diagram state
  - Diagram interface
  - Diagram collaboration
- **Textual view**
  - Object Constraint Language (OCL)
  - Bahasa pemrograman(C++)
  - Atribut dan properti lain dari elemen model UML

PEDOMAN YANG FLEKSIBEL UNTUK DESAIN CLASS #1

13

- **Petunjuk fleksibel untuk desain class:**
  - Petunjuk yang menuntun untuk desain clas yang lebih lanjut dan reusabel. Program berorientasi obyek yang berpasangan dan kohesif. Normalisasi class untuk kohesif
- **Mekanisme mengembangkan UML :**
  - penggunaan properti, batasan dan stereotype untuk mengembangkan notasi UML.
  - Stereotype standar, termasuk interface.
  - Penggunaan interface untuk menyatakan peran
- **Konsep dan Notasi untuk Diagram Interaksi:**
  - konsep dan notasi untuk diagram collaboration dan diagram sequence.
  - Menandakan iterasi, pencabangan dan pembuatan dan penghapusan obyek setiap jenis diagram.
  - Relasi dari diagram interaksi untuk diagram class.

COMPANY

PEDOMAN YANG FLEKSIBEL UNTUK DESAIN CLASS #2

14

- **Konsep dan Notasi untuk Diagram State Transisi:**
  - jika menggunakan diagram state transisi.
  - Notasi untuk diagram, termasuk state komposit, state history dan konkurensi state machine
- **Pendekatan desain perilaku:**
  - pendekatan untuk desain perilaku class dengan 'top-down' vs 'bottom-up'.
  - Peninjauan use case: 3 pendekatan untuk mengenal permasalahan use case.
  - Proses 'top-down' dari kebutuhan menentukan skenario, kemudian mengubah skenario ke diagram interaktif, metode obyek dan state machine.
  - Pendekatan 'bottom-up' terkonsentrasi pada tanggung jawa class

COMPANY

PEDOMAN YANG FLEKSIBEL UNTUK DESAIN PERILAKU #1

15

- **Petunjuk fleksibel untuk desain perilaku:**
  - Petunjuk untuk mengalokasikan dan mendesain perilaku sehingga desain lebih fleksibel
  - Kunjungan berpasangan
  - Menghindari kontrol terpusat
  - Penggunaan berlebihan dari metode asessor
- **Arsitektur sistem:**
  - Arsitektur terlayer
  - Kosep paket dan notasi UML nya
  - Menentukan layer an subsistem sebagai paket
  - Bagaimana dekomposisi sistem ke dalam UML
  - Diagram komponen

COMPANY

PEDOMAN YANG FLEKSIBEL UNTUK DESAIN PERILAKU #2

16

- **Konkurensi dan sinkronisasi:**
  - Thread dan proses
  - Mengolah akses konkuren ke obyek
  - Pendekatan terjadwal
  - Mengenalkan konkurensi dalam diagram interaktif
- **Distribusi dan persistensi:**
  - Distribusi fisik dan diagram deployment
  - Penekanan distribusi di atas diagram interaksi
  - Pengenalan ke obyek
  - File vs database relasional vs database OO
- **Pengenalan ke Java:**
  - Pengenalan pendek ke bahasa pemrograman Java sebagai contoh dari bahasa pemrograman OO
- **Ungkapan desain Low-Level:**
  - Petunjuk untuk desain class interface
  - Ungkapan untuk desain low-level dari atribut class, asosiasi dan operasi

COMPANY