PENGEMBANGAN SISTEM DINAMIS (DEVELOPING DINAMIC SYSTEM)

Ama Fariza, S.Kom, M.Kom

MATERI

- Pendahuluan
- Diagrammatic View yang didukung oleh UML
- Dari permulaan ke transisi
- Meng-arsitektur software

PENDAHULUAN #1

3

- Pada bab ini akan digambarkan UML, sebagai model desain dan analisis berorientasi obyek
- UML hanyalah sebagai bagian dari model pengembangan
- UML (Booch-Jacobson-Rumbaugh 1998) adalah hasil kolaborasi antara beberapa perusahaan dan ahli pemodelan OO, yang dipimpin oleh Rational Software Corporation dan mengambil dari Booch (Booch 1994), OMT (Rumbaugh et al 1991), dan model OO lain (Coad-Yourdon 1991a)(Coad-Yourdon 1991b)(Jacobson 1992)
- Desainer UML adalah Booch, Jacobson dan Rumbaugh
- UML adalah bahasa untuk spesifikasi, visualisasi, konstruksi dan dokumentasi artifact dari sistem sofware sebagaimana pemodelan bisnis dan sistem non-software lain (Booch-Rumbaugh- Jacobson 1997).

COMPANY

PENDAHULUAN #2

4

- 4 teknik desain grafis populer yang didukung oleh UML yaitu diagram collaboration, diagram class, diagram use-case dan diagram state
- **Use case (atau diagram use case) :**
  - Menggambarkan interaksi antara pengguna dan sistem atau antara satu sistem dan lain.
  - Use case menyediakan *high level view* dari penggunaan sistem dimana sering menunjukkan interaksi beberapa fungsi dari sistem.
  - Misalnya: mengedit dokumen melibatkan fungsi *open document, edit document* dan *save document*

COMPANY

PENDAHULUAN #3

5

- **Interaction (misalnya diagram Sequence dan diagram Collaboration):**
  - Juga disebut mini-uses
  - Menunjukkan contoh konkrit (misalnya test case) dari bagaimana komponen berkomunikasi
  - Juga dilihat sebagai test case aktual yang melibatkan penggunaan satu fungsi (melihat use case)
  - Sebuah fungsi merujuk ke GUI (misalnya open file) atau ke sub komponen dari sistem

COMPANY

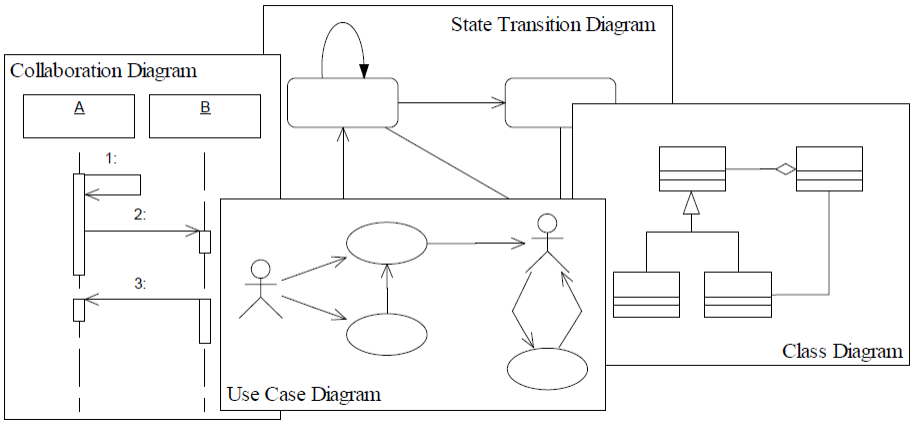
PENDAHULUAN #4

6

- **Obyek dan Class (misalnya diagram Class, CRC Cards):**
  - Class adalah bentuk utama dari UML (dan semua model OO lain)
  - Menggambbarkan hubungan antara class dan obyek yang merupakan komponen terkecil yang berdiri sendiri pada OO
  - Hubungan tersebut dapat menggambarkan instance (anggota), bagian dari hubungan, ketergantungan (dependency) dan lain-lain

COMPANY

PENDAHULUAN #5



Diagrammatic View yang didukung oleh UML #1

- **Package (misalnya diagram Package):**
  - Paket digunakan untuk mengelompokkan class ke dalam layer dan partisi
  - Menunjukkan dekomposisi dari sebuah sistem
- **Transisi state (misalnya diagram State dan diagram Aktifitas):**
  - Merupakan teknik yang well-known yang digunakan dalam UML untuk menggambarkan state dari sebuah class yang dilalui
  - Diagram state dibatasi hanya untuk satu class
  - Diagram aktifitas adalah generalisasi dari diagram state yang digunakan untuk menggambarkan event atau elemen transisi lain
- **Deployment (misalnya diagram Deployment):**
  - Menunjukkan realsi dari komponen dari sistem selama deployment
  - Menyatakan bentuk fisik dari sistem
  - Tetapi biasanya digunakan untuk menggambarkan ketergantungan kompenen dari implementasi aktual

Diagrammatic View yang didukung oleh UML #2

9

- UML tidak meng-cover keseluruhan bentuk stakeholder. Beberapa bentuk tambahan yang juga penting
  - **Aliran Informasi (misalnya diagram data flow):**
    - Menunjukkan aliran fungsional dari informasi
    - Khususnya berguna untuk user dan konsumen karena manusia lebih banyak berfikir tentang aliran informasi (dokumen dll)
    - Tetapi, tidak sesuai dengan desain obyek orientasi
  - **Antarmuka/dialog (misalnya diagram interface flow):**
    - Menggambarkan penggunaan antar muka user dari sistem yang tidak menyatakan struktur OO
    - Menyatakan use case dalam menggambarkan fungsional yang perlu tersedia dalam antar muka user (GUI)
    - Diagram interface/dialog menyatakan apa yang user lihat dari sistem

COMPANY

Diagrammatic View yang didukung oleh UML #3

10

- Object contraint language (OCL) (Booch-Rumbaugh-Jacobson 1997) mendukung model UML dan menyediakan beberapa integrasi terbatas dalam dan antar bentuk
- OCL adalah bahasa formal untuk mengekspresikan batasan dalam elemen model UML
- Karena user dapat mengembangkan UML (misalnya melalui stereotype), OCL juga membantu mengintegrasikan teknik baru dalam UML

COMPANY

Dari permulaan ke transisi #1

11

- Secara natural, sebuah model pengembangan software seharusnya berisi informasi yang cukup sehingga berguna melalui seluruh siklus pengembangan
- Sehingga model pengembangan seharusnya mempunyai bentuk memberikan informasi tentang persyaratan, organisasi, orang, teknologi, sejarah perubahan, struktur produk, transisi dan lain-lain
- Melihat model semata-mata sebagai kumpulan bentuk bukanlah sesuatu yang baik, tetapi seharusnya lebih dari itu.
- Sebagai contoh, seharusnya memberikan informasi bagaimana menggunakan model dan bentuknya. Seperti, menjawab pertanyaan 'apa yang harus dilakukan pertama' dan 'siapa yang perlu mengerjakannya'
- Model tradisional yang memberikan tuntunan disebut model proses atau model life-cycle adalah model Waterfall (Royce 1970), menatakan proses pengembangan sebagai proses berurutan (feedback loop diperbolehkan) dengan tahap perencanaan, dilanjutkan desain tingkat tinggi dan implementasi
- Meskipun secara luas diketahui bahwa proses berurutan umumnya tidak layak untuk diikuti, tahap yang didefinisikan masih tepat pada beberapa model proses yang baru

COMPANY

Dari permulaan ke transisi #2

12

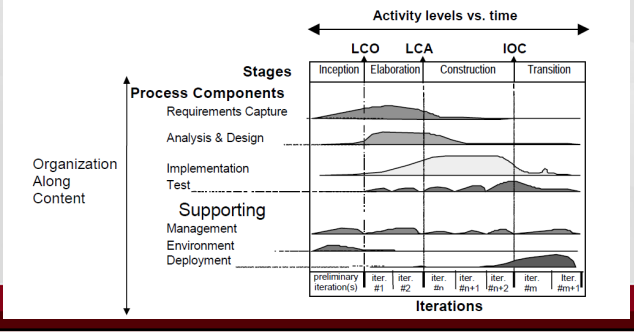
- Model waterfall telah mati dan tidak ada model lain yang sama populer nya
- Komunitas berorientasi obyek percaya bahwa pengembangan OO berbeda dan model proses umum tidak dapat mengaplikasikannya
- Model waterfal masih dibutuhkan untuk melihat kompleksitas dari perangkat lunak, kesesuaian, kemampuan perubahan dan invisibilitas
- Paradigma object-oriented mungkin mudah untuk beberapa permasalahan, tetapi fakta lain menyatakan masih terdapat permasalahan yang sama dengan 30 tahun yang lalu – meskipun dengan OO

COMPANY

Rational Unified Process dengan model Spiral WinWin

13

- Model proses untuk pengembangan OO contohnya adalah ***WinWin Spiral Model (Boehm 1996)*** dan ***Rational Unified Process (Kruchten 1999).***
- **Rational Unified process** adalah integrasi dari **life-cycle objectives** (LCO), **life-cycle architecture** (LCA) dan **initial operational capability** (IOC) dari WinWin Spiral Model. Tahap UML dari pengembangan OO disebut **Permulaan (Inception), perluasan (Elaboration), konstruksi (Construction) dan transisi (Transition).** Setiap tahap bisa menggunakan satu iterasi atau lebih (misalnya siklus spiral) untuk menyelesaikannya dan setiap milestone harus sesuai dengan komponen proses di sebelan kiri. Misalnya, fokus pada analisa persyaratan, kemudian dilanjutkan konsentrasi pada implementasi dan tes, meskipun semua aktifitas berjalan konkuren. Meskipun selama konstruksi diidentifikasi persyaratan baru, maka model iteratif diperlukan.



Meng-arsitektur Software #1

14

- Untuk menggambarkan arsitektur software menjadi permasalahan sendiri.
- Ada yang menganalogikan arsitektur software dengan arsitektur bangunan.
- Baik bangunan dan software mempunyai representasi implementasi; Bangunan dalam bentuk bata, batu dll; implementasi produk software dalam bahasa pemrograman
- Baik arsitek bangunan dan software menggunakan deskripsi logika (**blueprint**) untuk menggambarkan implementasi.
  - **Arsitek bangunan** melakukannya dengan menggambarkan fitur esensial dari bangunan tanpa terlalu membahas detail konstruksi misalnya material yang digunakan
  - **Arsitek software** juga sama, tidak membahas detail implementasi misalnya bahasa pemrograman yang digunakan
- Bangunan berdiri untuk bertahun-tahun tetapi mengalami beberapa perubahan, demikian juga software juga mengalami perubahan sepanjang waktu
- Baik bangunan dan software mempunyai beberapa desain (yang harus selesai lebih dahulu) yang dapat berakibat untuk keputusan berikutnya, misalnya untuk mendukung atap tidak semudah memindahkannya, demikian juga beberapa desain software juga mengalami batasan yang sejenis.

Meng-arsitektur Software #2

15

- Arsitek bangunan dapat menggunakan kembali informasi, misalnya beberapa rumah dibangun, arsitek dapat menggunakan blueprint yang sama. Demikian juga software, bila konsep pengembangan populer, kita juga dapat menggunakannya untuk meng-arsitektur software lain
- Arsitek bangunan menggambarkan bangunan dengan detail sehingga pekerja dapat membangun tanpa banyak interaksi dengan arsitek, demikian juga dengan software
- Arsitektur software dan bangunan mempunyai beberapa prinsip yang sama. Prakteknya, deskripsi arsitektur software tidak sedekat dengan deskripsi arsitek bangunan, tetapi tujuannya yang mendekati
- Standard IEEE 1471 adalah standarisasi arsitektur software
- Setiap sistem mempunyai arsitektur, definisinya:

  sebuah arsitektur adalah konsep level tertinggi dari sistem dalam lingkungkan dimana: abstrak '**level tertinggi**' menjauhi detail desain, implementasi dan operasi dari sistem untuk fokus pada sistem '**bentuk seragam atau koheren**'; konsepsi '**menekankan sifat abstraksi manusia, tidak membingungkan dengan representasi konkret dalam dokumen, produk atau artefak lainnya; dan dalam lingkungan**' sistem berada dalam lingkungan tersebut, arsitektur sistem mencerminkan lingkungan tersebut.

COMPANY

Meng-arsitektur Software #3

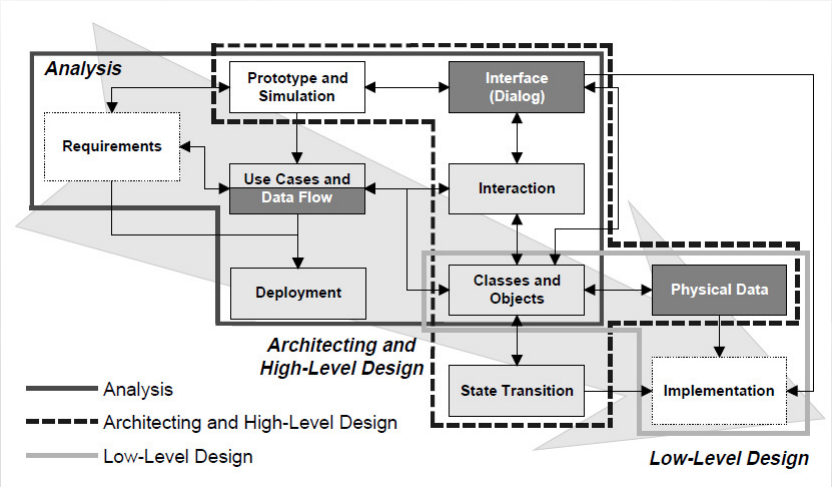
16

- Masalah utama ini definisi ini adalah **ketidakjelasan**: itu bisa berlaku juga untuk konsep arsitektur, persyaratan atau konsep operasional. Masalah lain yang kita lihat dengan definisi ini adalah bahwa hal itu tampaknya menekankan terlalu sedikit ke analisis dan interpretasi deskripsi arsitektur. Arsitek bukan hanya karena ingin membangun tetapi juga karena ingin memahami. Dengan demikian, meng-arsitek memiliki banyak hubungannya dengan menganalisis dan memverifikasi konseptual integritas, konsistensi, dan kelengkapan desain.
- Meng-arsitektur dalam UML adalah bagian dari tahap pengembangan mayor (dari pandangan developer) – analisis, arsitektur dan desain sistem software.

COMPANY

Meng-arsitektur Software #4

17



COMPANY