

Mata Kuliah

ANALISIS & DESAIN SISTEM (ADS)

System Analysis & Design

Dosen:

Budi Nugroho, S.Kom, M.Kom

Materi 5

Unified Modeling Language (UML)



**Program Studi S1 Informatika
Fakultas Ilmu Komputer UPN “Veteran” Jawa Timur**

TA 2019/2020 Semester Genap

Analysis & Design Methodology

Structured / Functional

- ✓ Melihat Sistem dengan pendekatan Prosedur.
- ✓ Sistem = kumpulan fungsi-fungsi yang saling berinteraksi membentuk kesatuan proses.
- ✓ Definisi Data dan Fungsi terpisah.
- ✓ Antar komponen saling dependen (tergantung satu sama lainnya).
- ✓ Waktu pembuatan sistem lebih cepat.
- ✓ Untuk sistem yang kompleks → pengembangan lebih sulit.

Object Oriented

- ✓ Melihat Sistem dengan pendekatan komponen.
- ✓ Sistem = kumpulan obyek-obyek yang saling berinteraksi.
Obyek = variabel + fungsi
- ✓ Definisi Data dan Fungsi di dalam 1 entitas (= Obyek).
- ✓ Antar komponen independen (tidak tergantung).
- ✓ Waktu pembuatan sistem lebih lama.
- ✓ Untuk sistem yang kompleks → pengembangan lebih mudah.

Sistem Akademik

Structured
/ Functional

Dekomposisi berdasarkan
fungsi / proses

Object Oriented

Dekomposisi berdasarkan
Objek (data + fungsi)

Presensi

Perkuliahan

Penjadwalan

Penilaian

Dosen

Jadwal

Kuliah

Mahasiswa

ooc

Object Orientation Concepts

Kelebihan

- ☞ Lebih mencerminkan dunia nyata (lebih tepat dalam menggambarkan entitas, dekomposisi berdasarkan pembagian yang natural, lebih mudah untuk dipahami & dirawat).
- ☞ Memudahkan pemanfaatan ulang code dan arsitektur.
- ☞ Kestabilan (perubahan kecil dalam requirement tidak mengakibatkan perubahan yang signifikan pada sistem yang sedang dikembangkan).
- ☞ Lebih mudah disesuaikan dengan perubahan proses bisnis dan kebutuhan pengguna.

Paradigma Berorientasi Obyek

Object dan Class

- Object = anggota atau instan dari class.
- Kelas = kategori dari beberapa object yang memiliki atribut dan operasi yang sama.
- Object memiliki semua karakteristik dari suatu class.

Paradigma Berorientasi Obyek

→ Abstraction

- Proses menentukan atribut dan operasi yang benar-benar diperlukan oleh suatu obyek.
- Tidak menyertakan atribut dan operasi yang tidak diperlukan untuk konteks masalah yang dihadapi.
- Misal : Obyek Sepeda Motor.

Apakah nomor mesin merupakan atribut dari obyek sepeda motor ?

Jika masalahnya berkaitan dengan sistem parkir, maka data yang perlu dicatat adalah nopol kendaraan. Nomor Mesin tidak diperlukan.

Jika masalahnya berkaitan dengan pembuatan STNK, maka nomor mesin diperlukan.

Paradigma Berorientasi Obyek

☞ Inheritance

- Class dapat mewarisi sifat (atribut dan operasi) dari class lain.
- Misal : Class mesin cuci, radio, dan televisi mewarisi sifat dari Class Peralatan Listrik (atribut Type, operasi Turn-on & Turn-off).
- Pewarisan (inheritance) dapat bertingkat.
Misal :
Class radio merupakan sub-class dari Class Peralatan Listrik. Class Peralatan Listrik merupakan sub-class dari Peralatan.

Paradigma Berorientasi Obyek

☞ Polymorphism

- Beberapa Class memungkinkan memiliki nama operasi yang sama, tetapi memiliki makna yang berbeda.
- Misal : operasi ‘membuka’.

Operasi ‘membuka’ pada class jendela dan operasi ‘membuka’ pada class buku memiliki makna yang berbeda.

Paradigma Berorientasi Obyek

☞ Encapsulation

- Suatu sistem memiliki kompleksitas yang tidak perlu diketahui oleh pengguna.
- Yang dibutuhkan pengguna hanyalah bagaimana bisa berinteraksi dengan sistem tersebut secara mudah.
- Encapsulation → menyembunyikan kompleksitas dari luar dan membuka operasi-operasi yang diperlukan.
- Misal : Sistem Televisi.
Pengguna tidak peduli dengan cara kerja rangkaian elektronika di dalamnya. Mereka hanya memperhatikan tombol apa saja untuk mengoperasikannya.

Paradigma Berorientasi Obyek

☞ Message Sending

- Antar obyek saling berkomunikasi untuk melakukan suatu operasi tertentu.
- Komunikasi antar obyek dilakukan dengan saling mengirimkan pesan (*message sending*).
- Misal : Televisi dan Remote Control.

Untuk melihat acara Televisi, tombol ‘on’ pada Remote Control ditekan → Obyek Remote akan mengirimkan pesan ke Televisi → Televisi akan menjalankan operasi ‘turn-on’.

Paradigma Berorientasi Obyek

☞ Association dan Aggregation

- Association : hubungan komunikasi / kerjasama antar obyek.
- Beberapa obyek yang saling mengirimkan pesan merupakan bentuk Association antara obyek tersebut.
- Aggregation : Association antara ‘keseluruhan’ dengan ‘sebagian’.
Misal : Komputer terdiri atas keyboard, mouse, monitor, CPU, dsb.

Hubungan antara Komputer dan komponen-komponen pembentuknya adalah Aggregation.

Unified Modeling Language (UML)

Definisi :

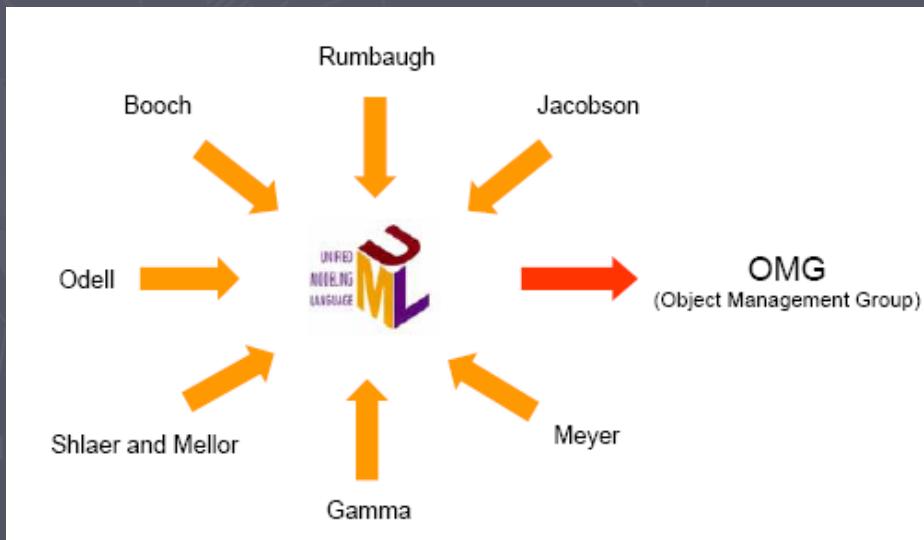
- ✓ Bahasa grafikal untuk visualisasi, spesifikasi, konstruksi, dan dokumentasi sistem.
- ✓ Pemodelan sistem berorientasi obyek.
Sebagai media untuk berkomunikasi antar stakeholder SI.

Perangkat Lunak Bantu pemodelan UML :

- ✓ Rational Rose, Power Designer, SmartDraw, StarUML, Enterprise Architecture, dsb.

UML History

- 1980 – 1990 :
Tiga pakar Analis terkenal (Grady Booch, James Rumbaugh, & Ivar Jacobson) membangun Bahasa Pemodelan dengan metodenya masing-masing.
 - ✓ Grady Booch → Notasi Booch / OOD (Object Oriented Design)
 - ✓ James Rumbaugh → OMT (Object Management Technology)
 - ✓ Ivar Jacobson → OOSE (Object Oriented Software Engineering)
- 1994 :
Kolaborasi metode ketiga Analis tersebut & sejumlah pakar lain (sebagian besar bekerja di Rational Software Corp.) menghasilkan UML.
- Januari 1997 :
Publikasi UML Versi 1.0 & dikirim ke OMG (Object Management Group).
OMG → Konsorsium Industri IT Internasional (berdiri 1989) yang mempromosikan Object-Oriented Technology.



- November 1997 :
UML 1.0 disetujui oleh OMG.
- 1998 :
OMG mengambil alih & merevisi UML. UML menjadi Standar Desain.
- 2002 : UML 1.5
- 2004 : UML 2.0
- 2010 : UML 2.3
- 2012 : UML 2.5
- 2017 : UML 2.5.1

Diagram UML

Business Process View	Business Functionality	Business Use Case Diagram
	Business Workflow	Business Activity Diagram
Functional System View	System Functionality	Use Case Diagram
	System Workflow	Activity Diagram
Organizational System View	Dinamic View	<ul style="list-style-type: none">Interaction Diagram• Sequence Diagram• Collaboration Diagram
	Logical View	Statechart Diagram
	Architectural View	Class Diagram
		Component Diagram
		Deployment Diagram



Terima
kasih

Thank You!
☺