Pertemuan 1

Rekayasa Perangkat Lunak dan Konsep Dasar Berorientasi Objek

Sistem dan Kriteria Penilaian

- **Sistem Penilaian:**
  - 20% Absensi
  - 25% Tugas & Quiz (Tugas kelompok di tiap pertemuan)
  - 25% UTS (Ujian Online)
  - 30% UAS (Ujian Online)

- **Kriteria Penilaian Tugas:**
  1. Dapat mengerjakan sesuai dengan perintah tugas (50%).
  2. Dapat menganalisa, menjelaskan deskripsi dan membuat diagram UML (25%).
  3. Dapat mengerjakan tugas dengan baik sesuai dengan tujuan, tepat waktu dan dapat bekerja sama (25%).

Rekayasa Perangkat Lunak

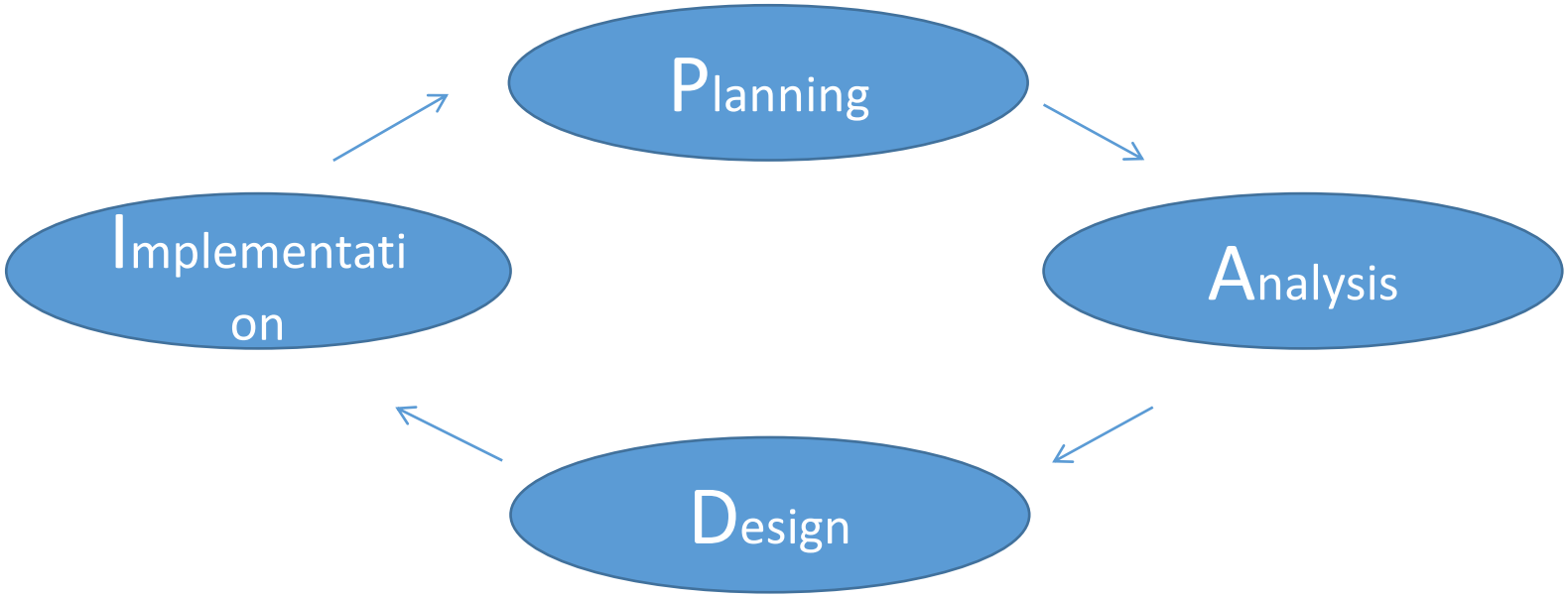
- Rekayasa Perangkat lunak merupakan Sebuah disiplin yang mengintegrasikan proses, metode dan alat-alat bantu untuk mengembangkan perangkat lunak komputer.
- Menghasilkan perangkat lunak komputer

Metode Pengembangan Perangkat Lunak

- Metode Pengembangan Perangkat Lunak atau Sistem merupakan Teknik yang digunakan dalam membuat sistem/perangkat lunak atau biasa disebut Software Development Life Cycle (SDLC)
- Dalam SDLC terdapat beberapa model yang umum digunakan:

1. Waterfall
2. Prototype
3. RAD
4. Spiral

Tahapan dalam Software Development Life Cycle (SDLC)



- Diagram diatas merupkan tahapan SDLC secara umum, secar khusus akan ada perbedaan pada tahapannya. Hal ini tergantung dari jenis metode SDLC yang diguankan dan buku referensi yang digunakan

Software Development Life Cycle (SDLC)

1. *Planning* (Perencanaan): Mengapa membuat sistem?

   Yang dilakukan pada tahap ini adalah membuat permintaan sistem (*System Request*) dan analisis kelayakan (*Feasibility Analysis).*

2. *Analysis* (Analisis): Siapa, apa, kapan, di mana sistem akan digunakan?

   Yang dilakukan pada tahap ini adalah pengumpulan kebutuhan (*Requirement Gathering*) dan membuat pemodelan proses bisnis (*Business Process Modeling*).

Tahapan Software Development Life Cycle (SDLC)

3. *Design* (Desain): Bagaimana sistem akan bekerja?

   Yang dilakukan pada tahap ini adalah merancang program (*Program Design*), merancang antar muka (*User Interface Design*), dan merancang data (*Data Design*). Dalam tahapan ini design sistem menggunakan diagram-diagram pemodelan sistem berbasis objek.

4. *Implementation* (Implementasi): Pembangunan dan penyampaian sistem.

   Yang dilakukan pada tahap ini adalam membangun, menguji, mendokumentasikan dan memasang sistem.

Pemodelan Sistem Berbasis Objek

Apa itu Pemodelan(Modeling)?

Proses merancang/mengambarkan perangkat lunak sebelum dilakukan pembuatan code program. Pemodelan digunakan untuk menyederhanakan permasalahan yang komplek sehingga lebih mudah dipahami dan dipelajari.

Pemodelan Sistem Berbasis Objek

- Sistem

  Kumpulan elemen yang saling terkait untuk mencapai suatu tujuan tertentu.

- Konsep dasar berorientasi objek

  Suatu teknik atau cara dalam melihat permasalahan suatu sistem. Sistem yang dikembangkan dianggap sebagai kumpulan objek dalam dunia nyata.

Pemodelan Sistem Berbasis Objek

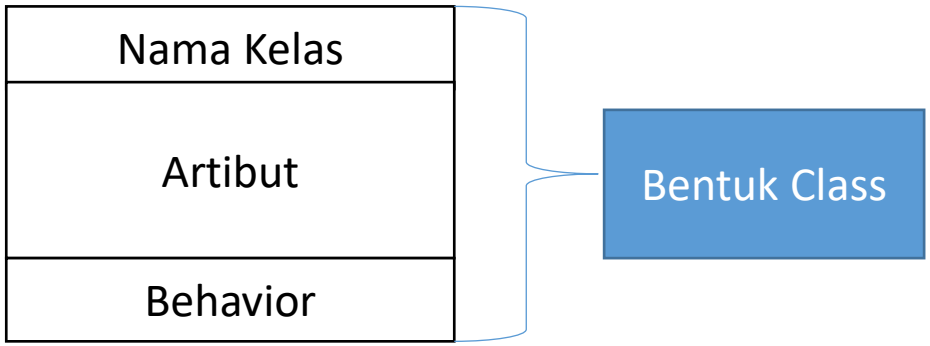
Modeling yang digunakan untuk merancang atau mendesign sistem atau perangkat lunak pada pemograman berorientasikan objek.

Konsep Dasar Berbasis Objek

- Kelas (class)

Kumpulan objek-objek dengan karakteristik yang sama. Sebuah kelas akan memiliki sifat (atribut/variabel), perilaku(metode/behavior), hubungan dan arti. Suatu kelas terdiri dari Nama kelas, atribut dan behavior.

- Class **adalah cetakan,** ***template*** **atau** ***blueprint***



Konsep Dasar Berbasis Objek

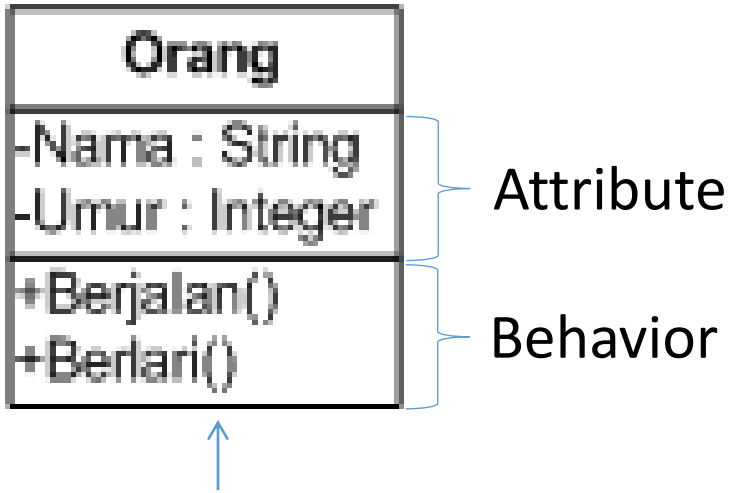
- Objek

  Objek adalah abstarsi dan sesuatu yang mewakili dunia nyata. Objek suatu entitas yang mampu menyimpan informasi ciri-ciri (state) dan perilaku (behavior).

- Objek adalah **bentuk nyata dari class**.
- Attribute dapat disebut juga **Variabel** (member) yaitu nilai datanya bisa ditentukan di object.
- Behavior dapat disebut juga **Perilaku** suatu objek dinyatakan dalam operation (method/fungsi).

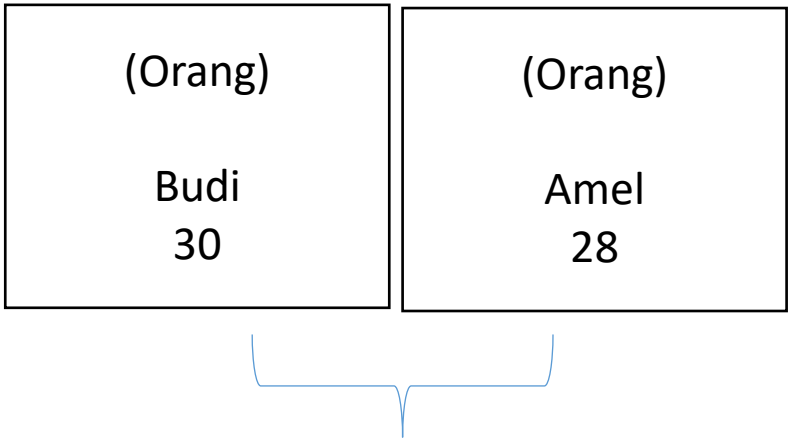
Perbedaan Class dan Object

Class/Cetakan



Class dengan atribut

Objek/Hasil



Objek dengan nilai

Karakteristik Pemrograman Berorientasi Objek

- Encapsulation
  - Mekanisme menyembunyikan suatu proses dalam sistem untuk menghindari interferensi dan menyederhanakan penggunaan sistem itu sendiri. Contoh: Tombol on/off pengaturan suhu pada AC.
  - Enkapsulasi berarti membungkus class dan menjaga apa apa saja yang ada di dalam class tersebut, baik method ataupun atribut, agar tidak dapat diakses oleh class lainnya. Oleh karena itu, terdapat level akses class yang terdiri dari Public, Protected, dan Private.
  - Enkapsulasi data dapat dilakukan dengan cara:
    - mendeklarasikan instance variable sebagai private
    - mendeklarasikan method yang sifatnya public untuk mengakses variable tersebut

Encapsulation dan Access Modifier

Modifier	Dalam Class yang Sama	Dalam Package yang Sama	Dalam SubClass	Dalam Package Lain
private	✓			
protected	✓	✓	✓	
public	✓	✓	✓	✓

Contoh Encapsulation

- **Class Mahasiswa**

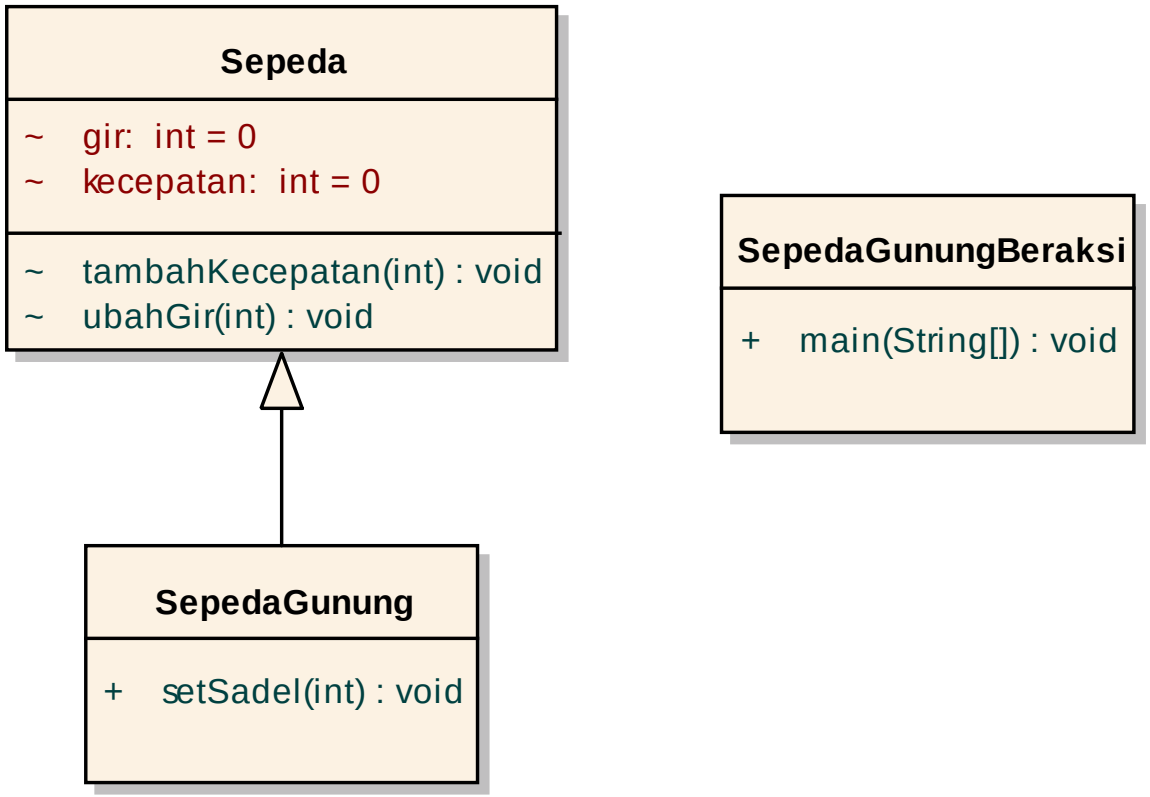
```
package latihan;
 public class Mahasiswa {
      private String nim;
      private String nama;
      private String kelas;
      private int nilaiAbsen;
      private int nilaiTugas;
      private int nilaiUTS;
      private int nilaiUAS;
      public Mahasiswa(){

      }
      public Mahasiswa(int kondisi){
      System.out.println("INPUT DATA MAHASISWA CUTI");
      }
 }
```

Karakteristik Pemrograman Berorientasi Objek

- Inheritance (Pewarisan)
  - Suatu class dapat mewariskan atribut dan method kepada class lain (subclass) serta membentuk class hierarchy.
  - Dalam pemrograman java, penerapan inheritance ditandai dengan keyword ***extends***.

Contoh Inheritance (Pewarisan)

- Class SepedaGunung mewarisi class Sepeda



Contoh Inheritance (Pewarisan)

- Class Sepeda

```
public class Sepeda{
        int kecepatan, gir ;

        // method
        void ubahGir(int pertambahanGir) {
                gir= gir+ pertambahanGir;
                System.out.println("Gir:" + gir);
        }

        void tambahKecepatan(int pertambahanKecepatan) {
                kecepatan = kecepatan+ pertambahanKecepatan;
                System.out.println("Kecepatan:" + kecepatan);
        }
}
```

Class SepedaGunung Mewarisi Class Sepeda

Class SepedaGunung	Class SepedaGunungBeraksi

class SepedaGunung extends Sepeda{
 	
public void setSadel(int nilaiSadel) {
System.out.println("Tinggi Sadel:"+ nilaiSadel);
}
}
	class SepedaGunungBeraksi {
	public static void main(String[] args) {
	// Membuat object
	SepedaGunung spd= new SepedaGunung();
	// Memanggil method di object
	spd.tambahKecepatan(10);
	spd.ubahGir(2);
	spd.setSadel(20);
	}
	}

Karena class SepedaGunung mewarisi class Sepeda, maka class SepedaGunung dapat mengakses method ubahGir dan tambahKecepatan dari class Sepeda.

Karakteristik Pemrograman Berorientasi Objek

- Polymorphism
  - Suatu objek dapat memiliki berbagai bentuk.
  - Implementasi konsep polymorphism:
    - Overloading: Penggunaan satu nama untuk beberapa method yang berbeda parameter.
    - Overriding: Terjadi ketika deklarasi method subclass persis sama dengan method dari superclassnya.

Contoh Polymorphism - Overloading

```
class Lingkaran{
   void gambarLingkaran(){
   }
    void gambarLingkaran(int diameter){
   ...
   }
    void gambarLingkaran(int diameter, int x, int y){
   ...
   }
    void gambarLingkaran(int diameter, int x, int y, int warna, String
   namaLingkaran){
   ...
   }
}
```

Contoh Polymorphism - Overriding

- **Class Sepeda**

```
class Sepeda{
        int kecepatan, gir ;

        // method
        void ubahGir(int pertambahanGir) {
                gir= gir+ pertambahanGir;
                System.out.println("Gir:" + gir);
        }

        void tambahKecepatan(int pertambahanKecepatan) {
                kecepatan = kecepatan+ pertambahanKecepatan;
                System.out.println("Kecepatan:" + kecepatan);
        }
}
```

Class SepedaGunung Mewarisi Class Sepeda

Class SepedaGunung	Class SepedaGunungBeraksi

class SepedaGunung extends Sepeda {
 	
void ubahGir(int pertambahanGir) {
gir= 2*(gir+ pertambahanGir);
System.out.println("Gir:" + gir);
}
}
	class SepedaGunungBeraksi {
	public static void main(String[] args) {
	// Membuat object
	SepedaGunung sepedaku = new sepedaGunung();
	// Memanggil method di object
	sepedaku.tambahKecepatan(10);
	sepedaku.ubahGir(2);
	}
	}

Class SepedaGunung membuat method yang sama persis dengan superclassnya (class Sepeda), yaitu void ubahGir(int pertambahanGir)

Model Analisis Desain dan Diagram

Model	Diagram
Data-oriented	*Data Flow Diagram (DFD)*
Process-oriented	*Flowchart*
Object-oriented (Data + process)	*Unified Modeling Langauge (UML)*

UML

- UML: *Unified Modeling Language*
- UML adalah bahasa pemodelan visual yang digunakan untuk menspesifikasi, memvisualisasi, membangun, dan mendokumentasikan artefak dari sistem perangkat lunak.

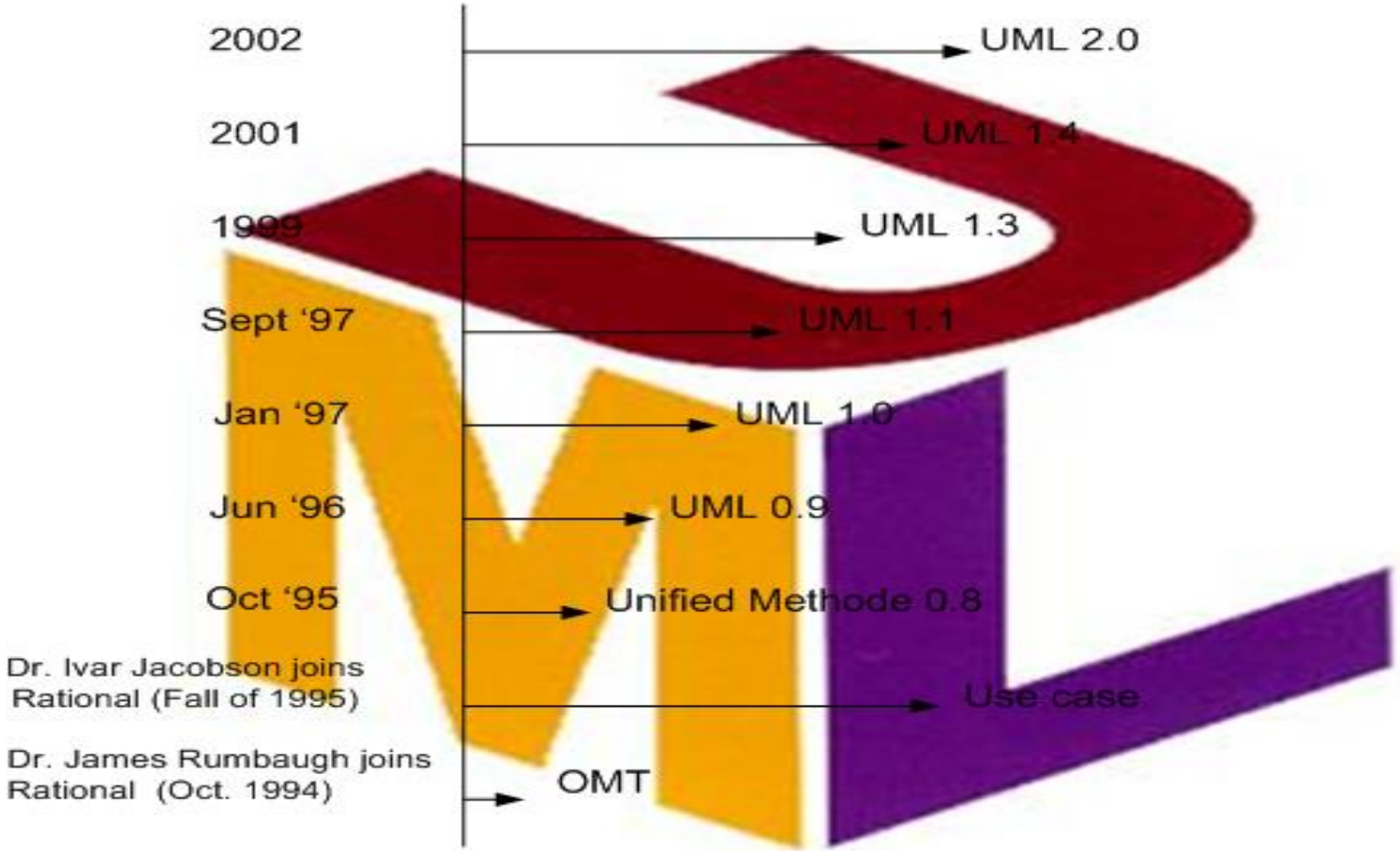
Sejarah UML

- Pada Oktober 1994, Dr. James Rumbaugh bergabung dengan Perusahaan Rational sotware, dimana Grady Booch sudah bekerja disana sebelumnya. Grady Booch mengembangkan Object Oriented Design (OOD) dan Dr. James Rumbaugh mengembangkan Object Modeling Technique (OMT). Duet Mereka pada Oktober 1995 menghasilkan Unified Method versi 0.8.

Sejarah UML

- Musim gugur 1995 Dr. Ivar Jacobson ikut pula bergabung dengan duet Rumbaugh-Booch, dengan memperkenalkan tool use case. Trio tersebut pada bulan Juni 1996 menghasilkan Unified Modeling Language (UML) versi 0.9. Sebelumnya Dr. Ivar Jacobson mengembangkan Object Oriented Software Engineering (OOSE)
- Banyak perusahaan software merasakan bagaimana pentingnya UML dalam tujuan strategis mereka, sehingga beberapa perusahaan membentuk sebuah konsorsium yang terdiri dari perusahaan-perusahaan seperti Microsoft, Oracle, IBM, Hewlett-Packard, Intellicorp, I-Logix, DEC, Digital Equipment Corp. texas instrument

Sejarah UML



UML Tools

- Microsoft Visio
- draw.io
- Enterprise Architect
- Star UML
- Netbeans UML Plugin
- Rational Rose
- Visual Paradigm

Diagram UML

- UML versi 2.4 memiliki 14 diagram yang dibagi ke dalam 2 grup utama:
  - Structure Diagram
  - Behavior Diagram

Structure Diagram

- Structure Diagram merepresentasikan data dan hubungan statis di dalam suatu sistem informasi.
- Structure Diagram terdiri dari:
  - Class Diagram
  - Object Diagram
  - Package Diagram
  - Deployment Diagram
  - Component Diagram
  - Composite Structure Diagram

Structure Diagram

- Class Diagram
  - Mewakili sesuatu, contoh: pegawai, gaji, dst.
  - Menunjukkan relasi antar class
- Object Diagram
  - Mirip dengan class diagram
  - Menunjukkan relasi antar objek
- Package Diagram
  - Mengelompokkan elemen-elemen UML untuk membentuk tingkat konstruksi yang lebih tinggi

Structure Diagram

- Deployment Diagram
  - Menunjukkan arsitektur fisik dan komponen perangkat lunak dari sistem
  - Contoh: simpul jaringan (*network nodes*)
- Component Diagram
  - Relasi fisik diantara komponen perangkat lunak
  - Contoh: Client/Server
- Composite Structure Diagram
  - Mengilustrasikan struktur internal dari sebuah class yang kompleks

Behavior Diagram

- Behavior Diagram yaitu kumpulan diagram yang digunakan untuk mengambarkan perilaku atau rangkaian perubahan dalam sistem.
- Behavior Diagram terdiri dari:
  - Activity Diagram
  - Sequence Diagram
  - Communication Diagram
  - Interaction Diagram
  - Timing Diagram
  - Behavior State Machine
  - Protocol State Machine
  - Use Case Diagram

Behavior Diagram

- Activity Diagram
  - Memodelkan proses dalam suatu sistem informasi
  - Contoh: alur kerja bisnis (*business workflows*), logika bisnis (*business logic*)
- Interaction Diagram
  - Menunjukkan interaksi diantara objek
- Sequence Diagram
  - Pengurutan interaksi berdasarkan waktu
- Communication Diagram
  - Komunikasi antara sekumpulan objek yang berkolaborasi dari suatu aktivitas

Behavior Diagram

- Timing Diagram
  - Menunjukkan bagaimana suatu objek berubah seiring waktu
- Behavior State Machine
  - Memeriksa perilaku suatu class
  - Memodelkan keadaan dan transisi keadaan yang berbeda yang dapat dialami suatu objek
- Protocol State Machine
  - Mengilustrasikan ketergantungan antara berbagai antarmuka dari suatu class
- Use Case Diagram
  - Menunjukkan interaksi antara sistem dan lingkungannya
  - Menangkap kebutuhan bisnis

SDLC dan Artefak

1. Planning
   - System Request
   - Feasibility Analysis

   → Proposal Sistem

2. Analysis
   - Use Case Diagram
   - Activity Diagram
   - Sequence Diagram
3. Design
   - Class Diagram
   - Deployment Diagram
   - User Interface Design
   - Data Model

   → Spesifikasi Sistem

4. Implementation
   - Program Code
   - Testing Plan
   - Documentation

   → Software Baru