

REKAYASA PERANGKAT LUNAK



ANALISIS

Defri Kurniawan M.Kom

email

Fasilkom | 4/25/2014

Penyelesaian Masalah by George Poyla



- George Poyla memberikan esensi praktik rekayasa perangkat lunak dalam menyelesaikan masalah meliputi [Pol45]:
 1. Pahami permasalahannya (komunikasi & analisa)
 2. Rancang solusinya (pemodelan & rancangan)
 3. Laksanakan rancangannya (kegiatan menulis kode)
 4. Periksa ketepatan hasilnya (pengujian & penjaminan kualitas)

Komunikasi



- Spesifikasi-spesifikasi kebutuhan pengguna harus diperoleh melalui aktifitas-aktifitas komunikasi sebelum dilakukannya analisis
- Sasaran dari spesifikasi kebutuhan adalah untuk memahami berbagai hal yang para *stakeholder* inginkan dari perangkat lunak yang akan dikembangkan

Software Requirement



- *Requirements engineering* adalah fase terdepan dari proses rekayasa perangkat lunak (*software engineering*), dimana *software requirements* (kebutuhan) dari user (pengguna) dan customer (pelanggan) dikumpulkan, dipahami dan ditetapkan.
- Kebanyakan kegagalan pengembangan *software* disebabkan karena adanya:
 - Ketidakkonsistenan (*inconsistent*),
 - Ketidaklengkapan (*incomplete*), maupun
 - Ketidakbenaran (*incorrect*)dari *requirements specification* (spesifikasi kebutuhan)

Software Requirement



- Studi di The Standish Group mencatat bahwa prosentase akumulatif kegagalan sebuah project pengembangan software sebagian besar disebabkan oleh masalah requirements dan spesifikasinya [Standish-94].

Software Requirement - Definisi



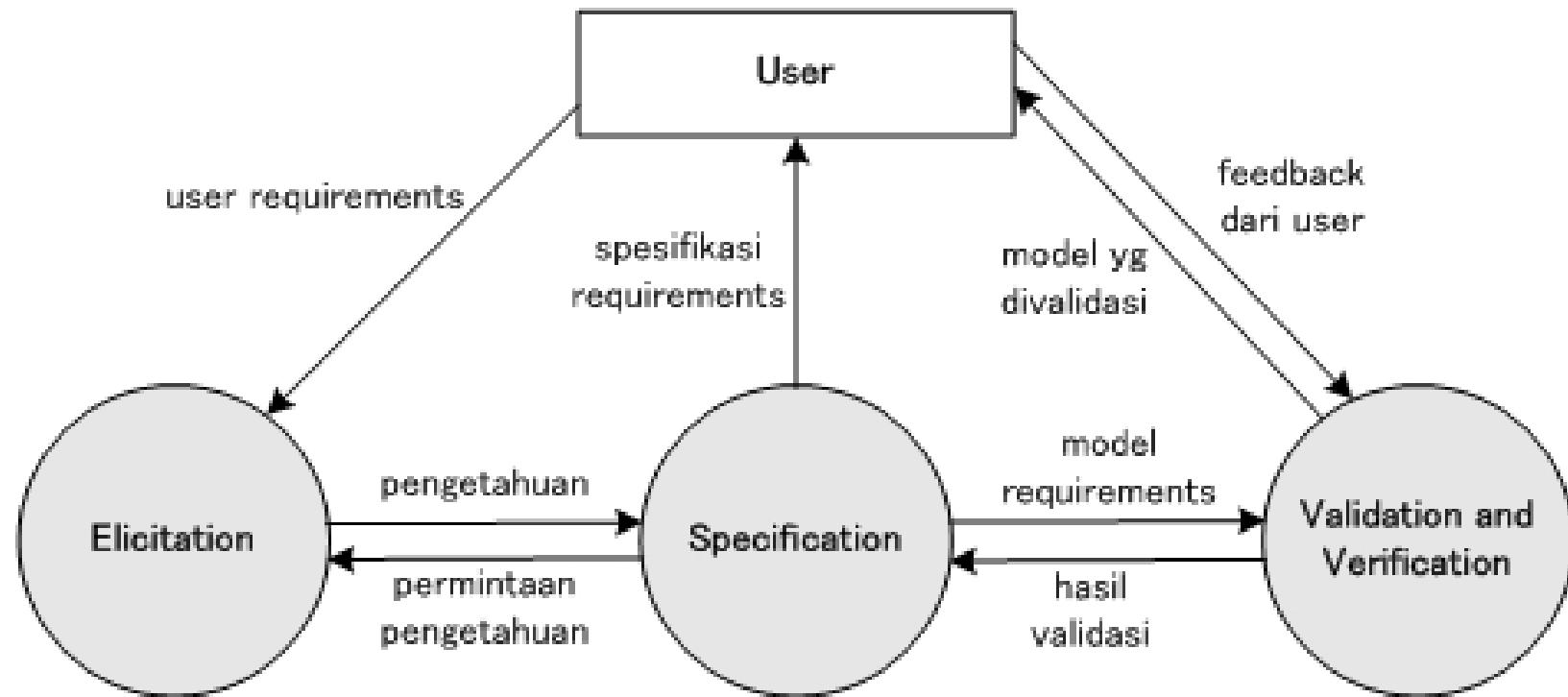
- *Requirements engineering* adalah cabang dari *software engineering* yang mengurusi masalah yang berhubungan dengan: tujuan (dunia nyata), fungsi, dan batasan-batasan pada sistem software. Termasuk hubungan faktor-faktor tersebut dalam menetapkan spesifikasi yang tepat dari suatu *software*, proses evolusinya baik berhubungan dengan masalah waktu maupun dengan software lain [Zave-97]

Software Requirement



- *Requirements engineering* dibagi dalam 3 proses besar yaitu:
 - elicitation,
 - specification,
 - validation and verification.
- Formula ini kemudian juga dikenal dengan nama ***The Three Dimensions of Requirements Engineering***
- Proses *requirements engineering* ini dilakukan secara iterasi dengan mengakomodasi adanya *feedback* dari customer (user).

Software Requirement



Software Requirement Process

Requirements Elicitation



- Adalah proses mengumpulkan dan memahami *requirements* dari user. Kadang masalah yang muncul berakar dari gap masalah *knowledge domain* (perbedaan disiplin ilmu yang dimiliki). Customer adalah expert pada domain yang softwarenya ingin dikembangkan (*domain specialist*), dilain pihak sang pengembang (*requirements analyst*) adakalanya sama sekali buta terhadap *knowledge domain* tersebut
- Gap *knowledge domain* tersebut yang diharapkan bisa diatasi dengan adanya interaksi terus menerus dan berulang (iterasi) antara pengembang dan customer

Requirements Specification



- Setelah masalah berhasil dipahami, pengembang mendeskripsikannya dalam bentuk dokumen spesifikasi. Spesifikasi ini berisi tentang fitur dan fungsi yang diinginkan oleh customer, dan sama sekali tidak membahas bagaimana metode pengembangannya.
- IEEE mengeluarkan standard untuk dokumen spesifikasi *requirements* yang terkenal dengan nama IEEE Recommended Practice for Software Requirements Specifications [IEEE-830].
- Dokumen spesifikasi requirements bisa berisi functional requirements, performance requirements, external interface requirements, design constraints, maupun quality requirements.

Requirements Validation and Verification



- Setelah spesifikasi requirements berhasil dibuat, perlu dilakukan dua usaha:
 - Validation (validasi), yaitu proses untuk memastikan bahwa requirements yang benar sudah ditulis.
 - Verification (verifikasi), yaitu proses untuk memastikan bahwa requirements sudah ditulis dengan benar.
- Proses validasi dan verifikasi ini melibatkan customer (user) sebagai pihak yang menilai dan memberi feedback berhubungan dengan requirements.

Requirement (Persyaratan)



- *Requirement* adalah pernyataan yang mendefinisikan tujuan atau batasan sistem yang harus terpenuhi
 - Perlu dipahami oleh tim pengembang dan divalidasi oleh para *stakeholder* dan pengguna (*user*)
 - Sebagai kriteria penentuan lolos / gagal yang dapat diverifikasi oleh tim penguji
 - Prioritas yang ditetapkan dalam kaitannya dengan persyaratan lain

Requirement (Persyaratan)



Requirement dibagi menjadi 2 (dua):

1. *Functional Requirement* (persyaratan fungsional)

"Functional requirements define what the system or application will do"

2. *Non-functional Requirement* (persyaratan non fungsional)

"A software requirement that describes not what the software will do, but how the software will do it, for example software performance requirements, software external interface requirements, design constraints, and software quality attributes" IEEE Definition

Non Functional Requirement (NFR)



- Persyaratan perangkat lunak yang menggambarkan bagaimana perangkat lunak akan melakukannya, misalnya, persyaratan kinerja perangkat lunak, persyaratan antarmuka eksternal perangkat lunak, dan atribut kualitas perangkat lunak.

- Persyaratan nonfungsional sulit untuk diuji oleh karena itu, mereka biasanya dievaluasi secara subyektif

Contoh Functional & Non Functional



- Contoh *Functional & Non Functional requirements* dalam pengembangan *Mobile Application*:
- *Functional Requirement*:
 - Cross platform compatible and works on most mobile browser
 - Integrates a selected number of popular social networking sites in one place
 - Communicates with social networking APIs
 - Uses login and OAuth mechanisms to authorize
 - Records and monitors social networking activity
 - Stores the data locally Displays total statistics for the user

Contoh Functional & Non Functional



■ Non functional requirements

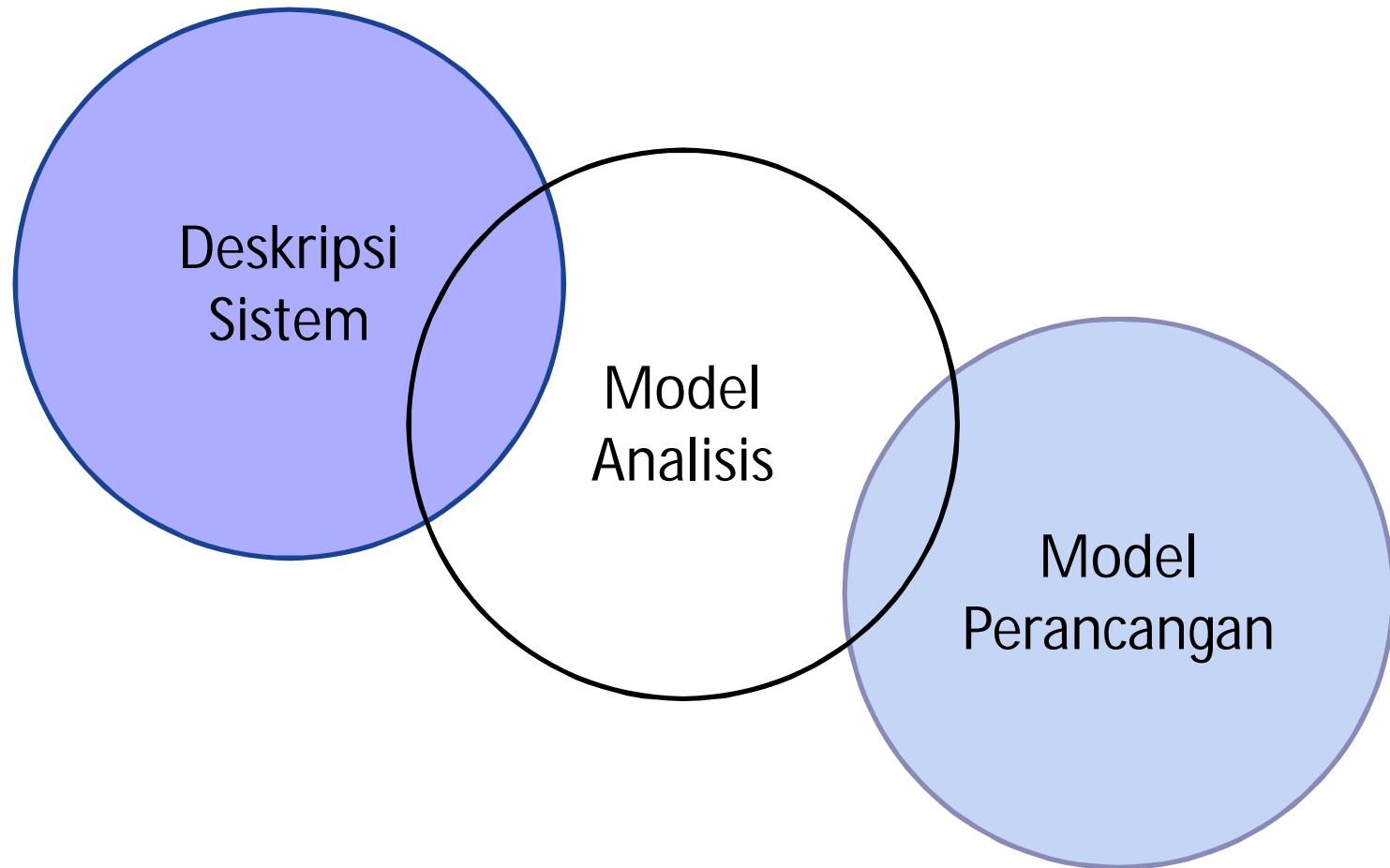
- Record statistics accurately
- Fast navigation
- Flexibility to choose which sites they want to integrate out of 3 and do not always have to use all 3. For example; the user should still be able to use Facebook and Twitter in the App and leave out YouTube (if they are not interested in YouTube).
- App should be able to function with chosen sites.
- Should be flexible in terms of being able to integrate other popular social networking sites too
- Should be available to users to use anytime

Model Analisis



- Analisis adalah tindakan yang terjadi saat kebutuhan-kebutuhan sudah didapatkan
- Sasaran model analisis adalah untuk memberikan deskripsi dari ranah informasional, fungsional, dan perilaku yang dibutuhkan untuk sistem-sistem berbasis komputer.
- Pemodelan analisis berfokus pada “Apa”, bukan “Bagaimana”

Letak Model Analisis



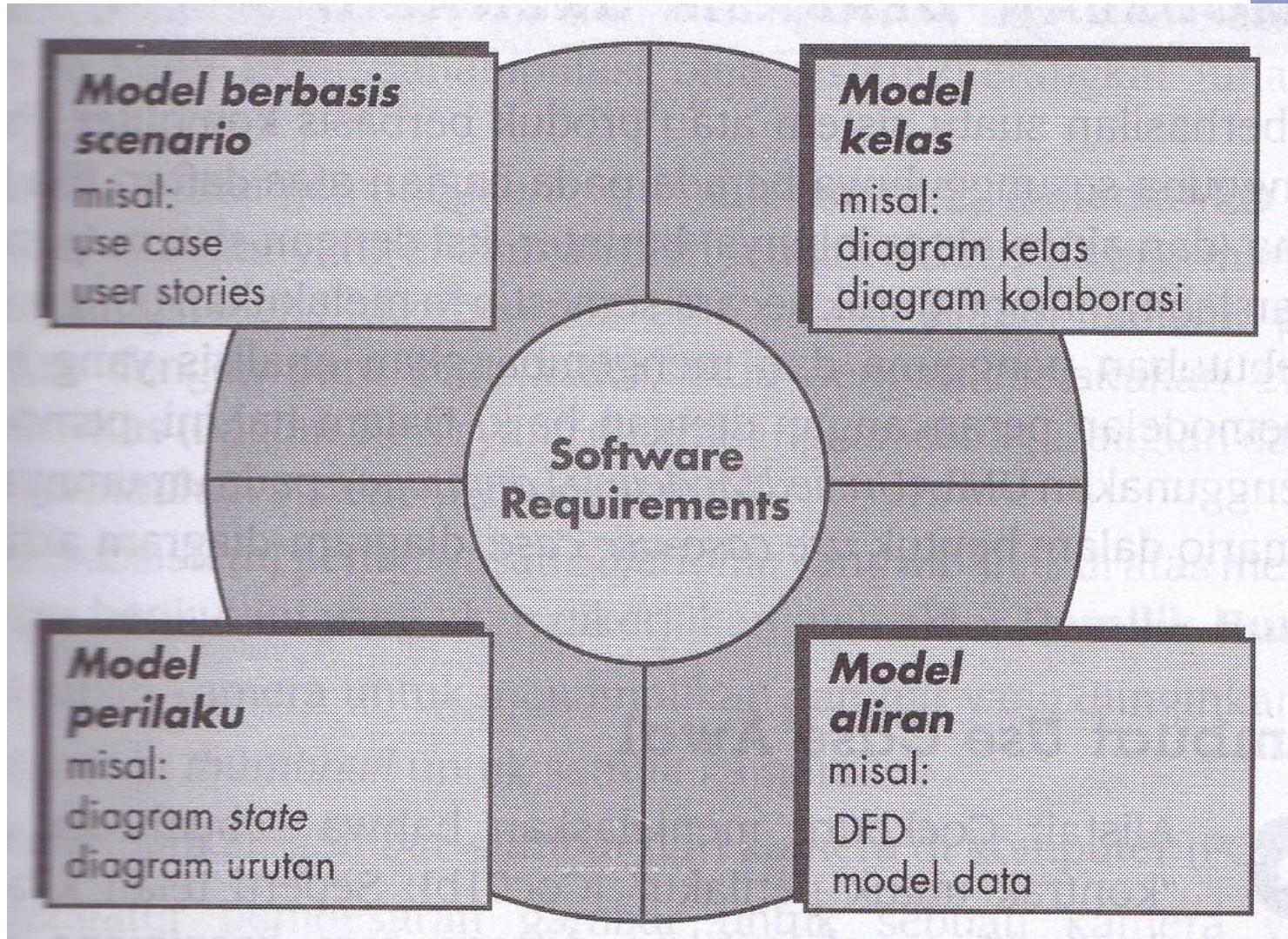
Model Analisis sebagai jembatan Deskripsi Model dan Model Perancangan

Elemen-elemen Model Analisis



- Secara umum, model-model analisis memiliki elemen-elemen spesifik seperti di bawah ini:
 - Elemen berbasis skenario
 - Elemen berbasis kelas
 - Elemen berbasis aliran
 - Elemen-elemen perilaku
- Bentuk representasi yang berbeda memberi pertimbangan kebutuhan-kebutuhan sistem/ perangkat lunak dari berbagai sudut pandang yang berbeda

Elemen-elemen Model Analisis



- Elemen-elemen berbasis skenario
 - Memperlihatkan bagaimana interaksi yang kelak akan terjadi antara pengguna dengan sistem/perangkat lunak
 - Memperlihatkan sejumlah aktifitas berurutan yang terjadi saat perangkat lunak digunakan
- Elemen model berbasis kelas
 - Memodelkan objek-objek yang akan dimanupulasi oleh sistem
 - Memodelkan operasi-operasi yang akan diterapkan
 - Memodelkan relasi yang terjadi antara objek satu dengan lainnya



- Elemen-elemen perilaku (*behavior*)
 - Memperlihatkan bagaimana event-event eksternal melakukan perubahan pada keadaan (state) sistem atau kelas-kelas yang ada di dalamnya
- Elemen-elemen berorientasi aliran
 - Memperlihatkan sistem/perangkat lunak yang bertindak sebagai pelaku transformasi informasi
 - Memperlihatkan bagaimana objek-objek data ditransformasikan saat mereka mengalir melintasi berbagai fungsi yang dimiliki sistem

Sasaran Model Analisis



- Model-model analisis harus mencapai 3 sasaran:
 - Untuk mendeskripsikan apa yang pelanggan inginkan
 - Menetapkan dasar bagi perancangan sistem/perangkat lunak
 - Untuk mendefinisikan sejumlah kebutuhan yang dapat divalidasi saat sistem/perangkat lunak dikembangkan



Analisis Terstruktur

- Objek-objek data dimodelkan dengan cara mendefinisikan atribut-atribut serta relasi-relasinya
- Memperlihatkan bagaimana caranya mereka melakukan transformasi data saat objek-objek data mengalir di dalam sistem yang akan dikembangkan

Analisis Berorientasi Objek

- Berfokus pada pendefinisan kelas-kelas dan cara bagaimana mereka saling bekerjasama satu dengan yang lainnya

REKAYASA PERANGKAT LUNAK



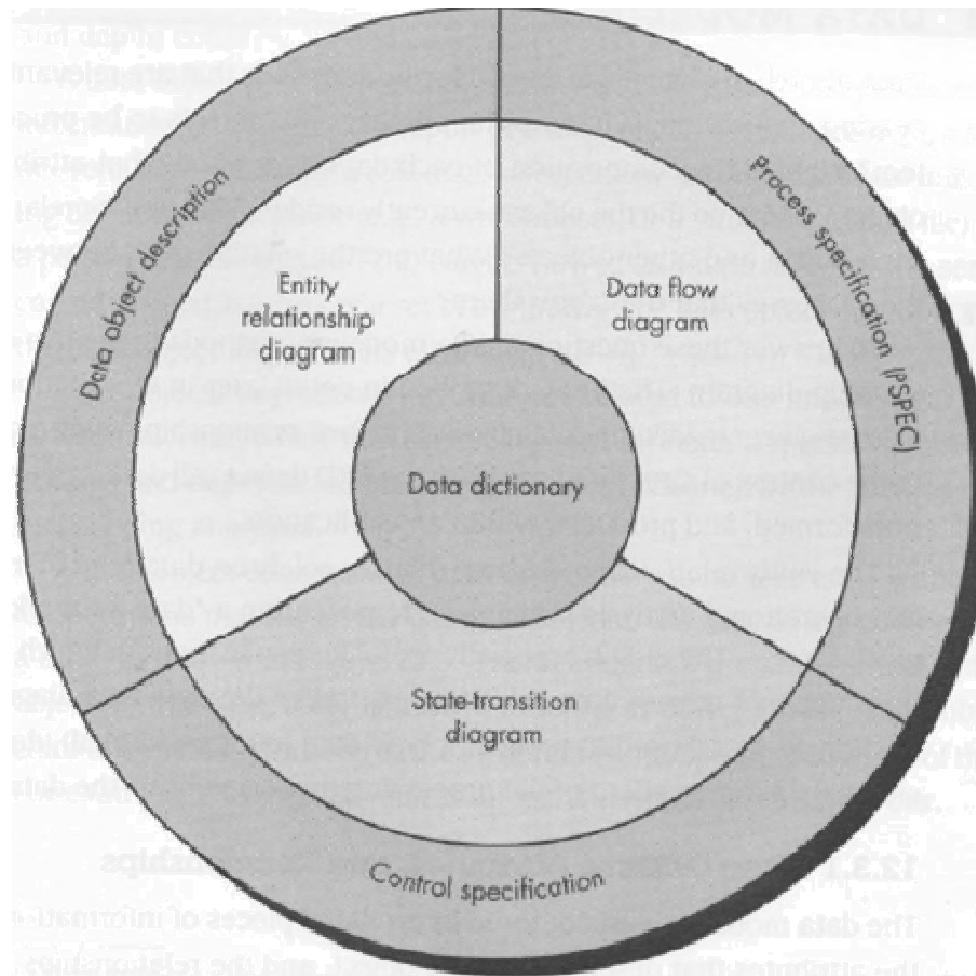
ANALISIS TERSTRUKTUR



Analisis Terstruktur

- Objek-objek data dimodelkan dengan cara mendefinisikan atribut-atribut serta relasi-relasinya
- Memperlihatkan bagaimana caranya mereka melakukan transformasi data saat objek-objek data mengalir di dalam sistem yang akan dikembangkan
- Mempertimbangkan data dan proses-proses yang melakukan transformasi terhadap data tersebut sebagai entitas-entitas yang saling terpisah satu dengan lainnya

Analisis Terstruktur



Bagan Model Analisis Terstruktur

Analisis Terstruktur



- **Data dictionary** : Deskripsi dari semua obyek data
- **ERD** : Menggambarkan hubungan antar obyek data.
- **DFD** :
 - Bagaimana data ditransformasikan pada sistem
 - Fungsi yang mentransformasikan aliran data
- **STD (*State Transition Diagram*)**: Bagaimana sistem bertingkah laku akibat kejadian eksternal
- **DOD (*Data Object Description*)** : deskripsi atribut untuk tiap obyek data
- **PSpec (*Process Spec.*)**: deskripsi tiap proses pada DFD
- **Control Spec.** : Deskripsi tiap transisi pada DFD



Kapan menggunakan Pemodelan Data?

- Jika kebutuhan-kebutuhan perangkat lunak mencakup kebutuhan untuk membuat, memperluas atau bersinggungan dengan basis data atau jika struktur data yang kompleks harus dibentuk dan dimanipulasi.
- Analis sistem akan menggunakan pendekatan analisis terstruktur dengan elemen-elemen berorientasi aliran

Data Modeling - ERD



- Memungkinkan untuk identifikasi obyek data dan hubungannya dengan menggunakan notasi grafis
- Menetapkan semua data yang dimasukkan, disimpan, ditransformasikan dan diproduksi pada suatu aplikasi
- Hanya berfokus pada data



Komponen-komponen ERD

- Entitas (*entity*)
- Relasi (*relationship*)
- Atribut (*attribute*)
- Kardinalitas (*kardinality*)
- Modalitas (*modality*)

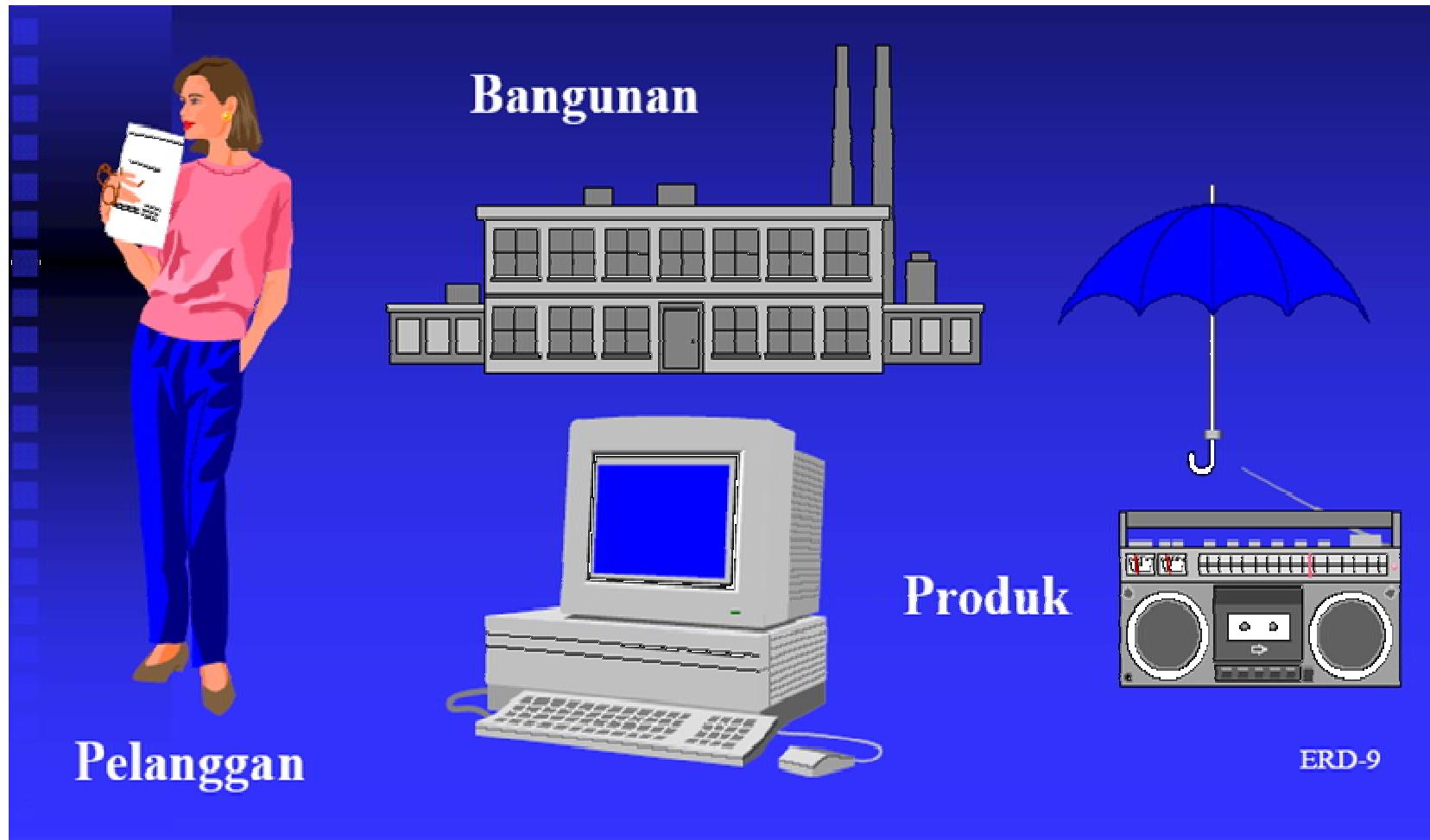
■ Definisi

- Sebuah obyek yang dapat dibedakan dari obyek lain

■ Contoh

- Individu : pegawai, pelanggan, mahasiswa, distributor
- Tempat : kampus, kantor, lapangan
- Obyek : buku, motor, paket software
- Peristiwa : pendaftaran, pemesanan, penagihan
- Konsep : rekening, kualifikasi

ERD – Enititas (Contoh)



Relasi

- Definisi

- ◆ Asosiasi 2 atau lebih entitas

- Berupa kata kerja



```
graph LR; Mahasiswa --> Mengambil; Mengambil --> MataKuliah
```

Relasi

Mengambil

Mahasiswa

Mata Kuliah

Atribut

■ Definisi

- ◆ Properti yang dimiliki setiap entitas yang akan disimpan datanya.

■ Contoh

- ◆ Atribut Pelanggan

- ◆ No KTP/SIM
- ◆ Nama
- ◆ Alamat



■ Definisi

- ◆ Angka yang menunjukkan banyaknya kemunculan suatu obyek terkait dengan kemunculan obyek lain pada suatu relasi
- ◆ Kombinasi yang mungkin : (1:1, 1:N, M:N)

ERD - Kardinalitas (Contoh)



1 Departemen mungkin mempekerjakan 1 atau lebih pegawai
1 Pegawai hanya bekerja pada sebuah departemen

■ Definisi

- ◆ Partisipasi sebuah entitas pada suatu relasi
- ◆ 0 jika partisipasi bersifat “optional”/parsial
- ◆ 1 jika partisipasi bersifat “wajib”/total

■ Contoh

- ◆ Partisipasi total
 - ◆ Setiap anak memiliki ibu
- ◆ Partisipasi parsial
 - ◆ Tidak setiap perempuan memiliki anak



Tahapan pembuatan E-R Diagram :

- Mengidentifikasi dan menetapkan seluruh himpunan entitas yang akan terlibat
- Menentukan atribut-atribut kunci dari masing-masing himpunan entitas
- Mengidentifikasi dan menetapkan seluruh himpunan relasi di antara himpunan entitas – himpunan entitas yang ada beserta foreign key (kunci tamu)
- Menentukan derajad / kardinalitas relasi untuk setiap himpunan entitas
- Melengkapi himpunan entitas dan himpunan relasi dengan atribut-atribut deskriptif

ERD – Langkah #1



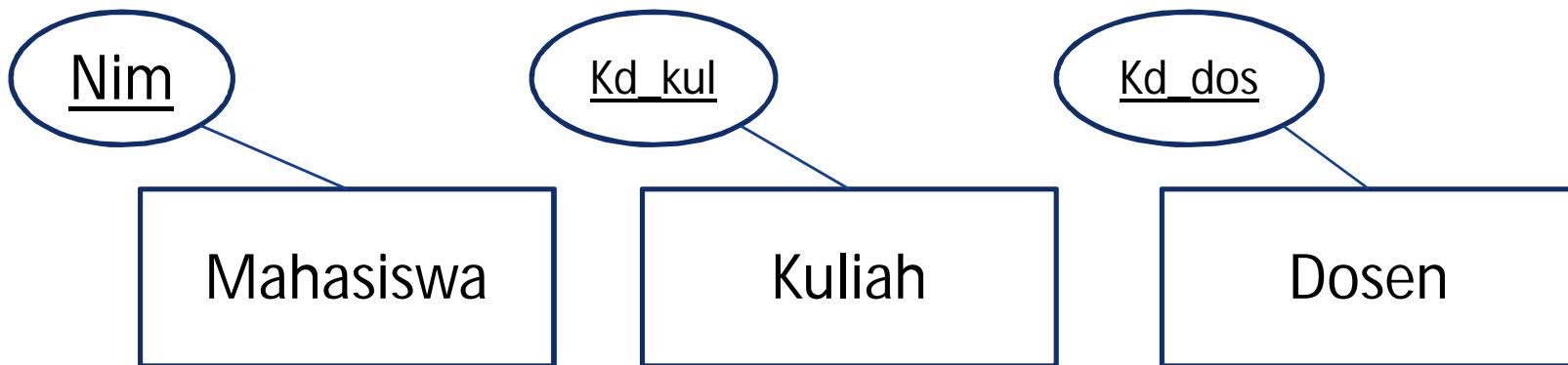
1. Mengidentifikasi dan menetapkan seluruh himpunan entitas yang akan terlibat

Mahasiswa

Kuliah

Dosen

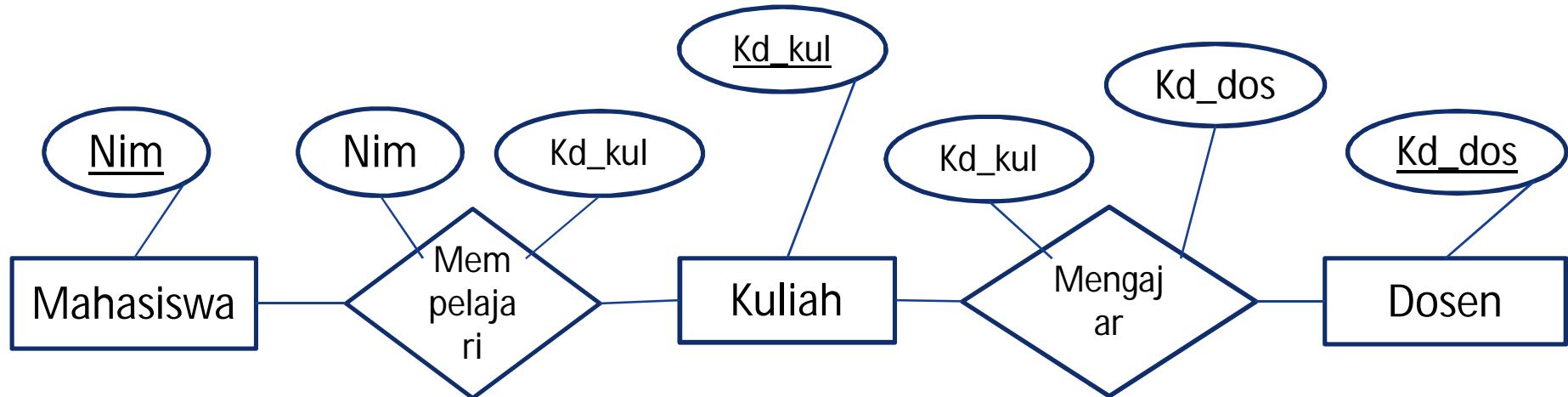
2. Menentukan atribut-atribut kunci dari masing-masing himpunan entitas



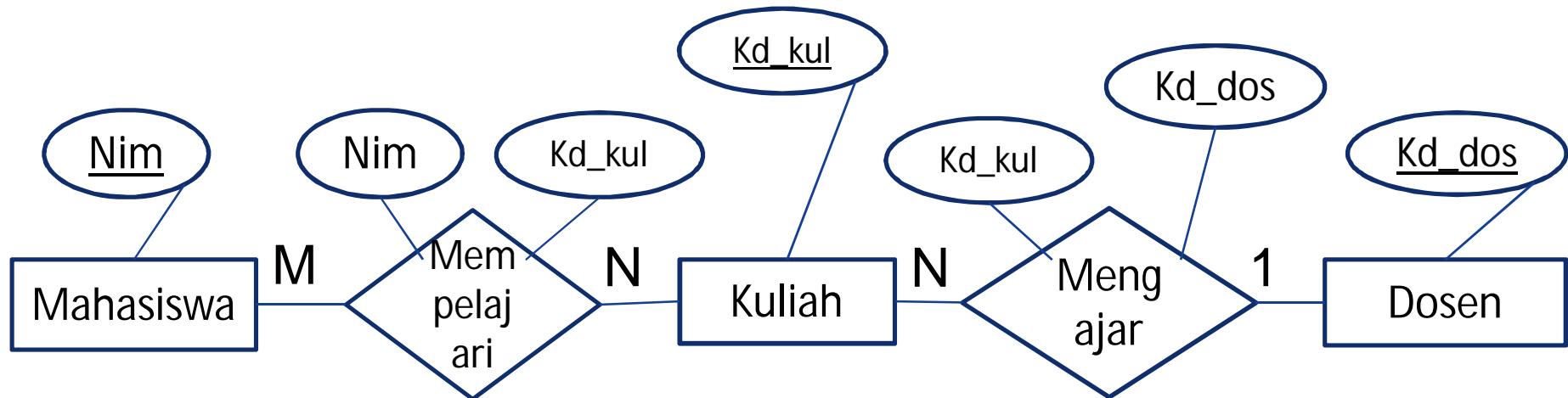
ERD – Langkah #3



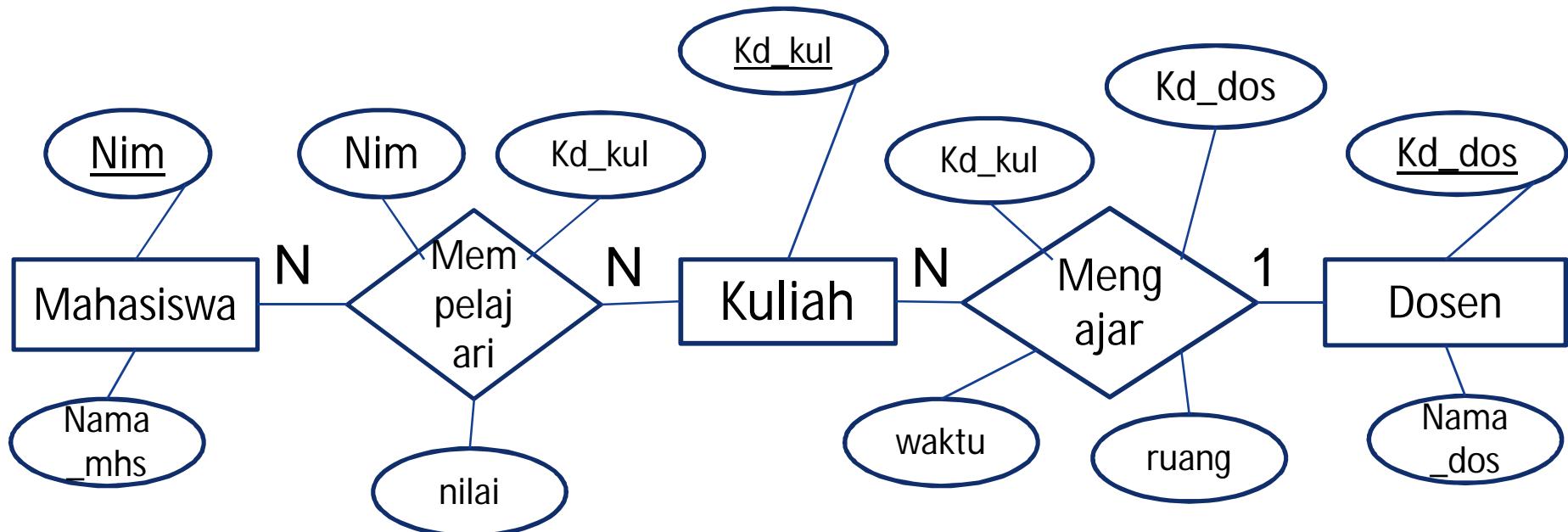
3. Mengidentifikasi dan menetapkan seluruh himpunan relasi di antara himpunan entitas – himpunan entitas yang ada beserta foreign key (kunci tamu)



4. Menentukan derajad / kardinalitas relasi untuk setiap himpunan entitas



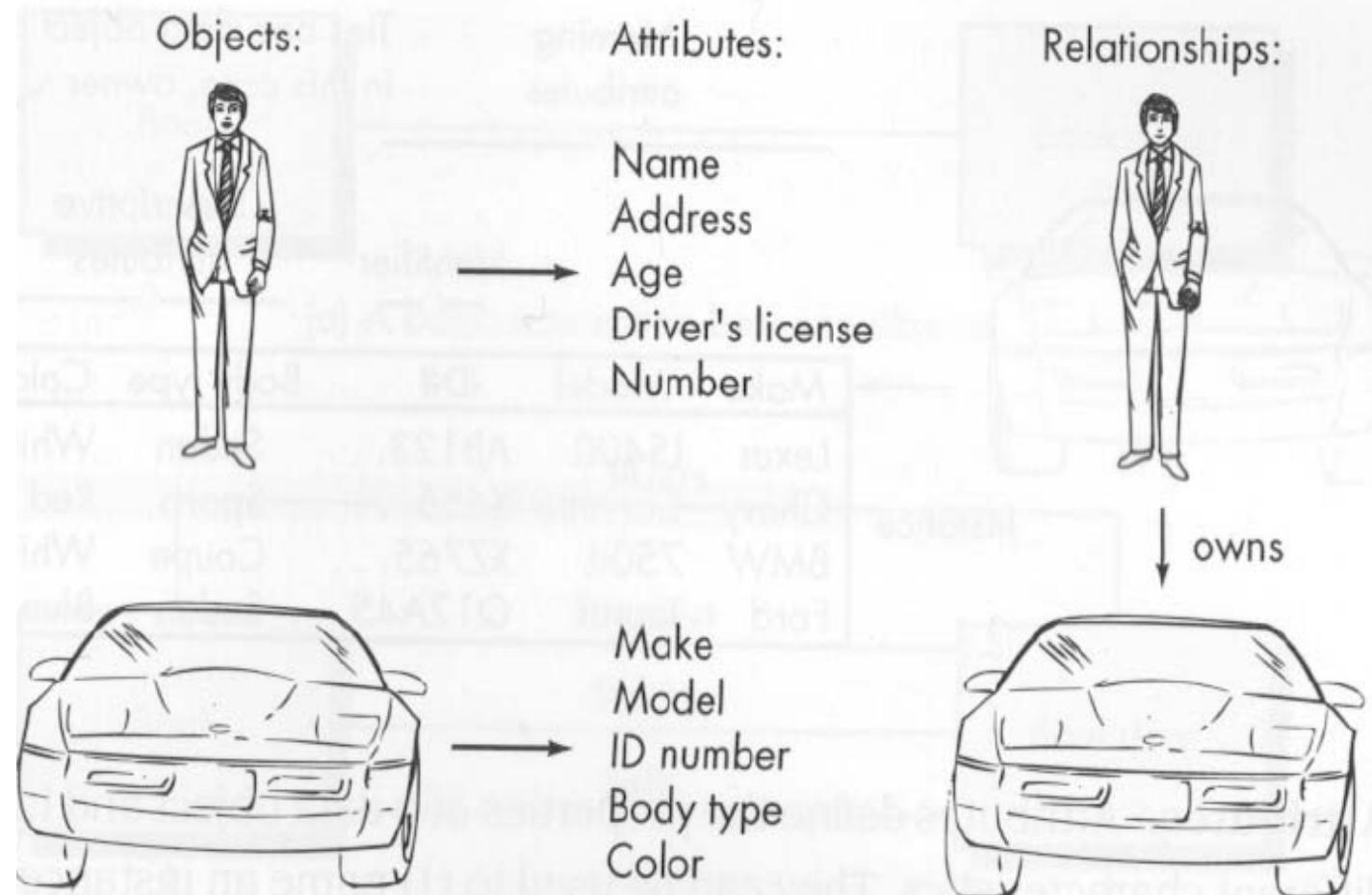
5. Melengkapi himpunan entitas dan himpunan relasi dengan atribut-atribut deskriptif



Pemodelan Data yang Baik

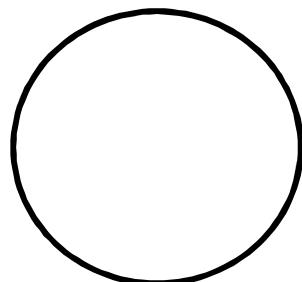
- Sederhana
- Tidak ada duplikasi data (*redundant*)
- Fleksibel dan mudah beradaptasi dengan perkembangan

Data Modeling (Konsep)



DFD (*Data Flow Diagram*)

- Memperlihatkan gambaran tentang masukan-proses-keluaran dari suatu sistem/perangkat lunak yaitu objek-objek data mengalir ke dalam perangkat lunak.
- DFD yang pertama sering sering disebut DFD level 0 atau Context Diagram
- DFD mengambangkan model-model dari suatu ranah informasional dan fungsional



Or

- **Entitas eksternal:** Penghasil/Penerima informasi/Perintah
- **Proses:** transfer informasi (fungsi) yang ada dalam bound sistem
- **Aliran data:** jembatan penghubung antara Entitas eksternal dan Proses ataupun proses dengan proses, proses dengan penyimpanan
- **Penyimpanan data**

Contoh Kasus



- Suatu perusahaan memiliki ide/terobosan tentang produk baru “produk-produk pengelola rumah” yang disebut dengan SafeHome. Teknologinya menggunakan antarmuka nirkabel protokol 802.11g yang memungkinkan pemilik rumah/pemilik bisnis kecil mengendalikan sistem dengan komputer pribadi untuk memantau keamanan/pengawasan rumah.

Contoh Kasus (lanj)



- Fungsi keamanan SafeHome memungkinkan pemilik rumah untuk melakukan konfigurasi terhadap sistem keamanan saat diinstal
- Memungkinkan pemilik rumah memantau semua sensor yang terhubung ke sistem keamanan melalui panel kendali
- Memungkinkan pemilik rumah berinteraksi atau menerima informasi melalui web browser, komputer pribadi atau penel kendali
- Masing-masing sensor akan memiliki nomer & jenisnya masing-masing serta memiliki kata sandi utama untuk mengaktifkan/menonaktifkan sistem

Contoh Kasus (lanj)



- Nomer telepon merupakan masukan (input) untuk pemanggilan telepon saat suatu event pada sensor terjadi
- Saat event pada sensor terjadi, perangkat lunak yang ada di sistem SafeHome akan mengaktifkan alarm suara
- Informasi yang ditampilkan melalui web browser, komputer pribadi atau penel kendali disebut antarmuka, dapat menampilkan pesan-pesan masukan tertentu dan informasi pada status penel kendali

Menyusun DFD – Analisis



- Bagaimana DFD-nya?
- Siapa penghasil informasi pada sistem?
- Siapa penerima informasi pada sistem?
- Apa/siapa saja yang terlibat pada sistem?
- Fungsional apa saja yang dimiliki sistem atau perangkat lunak yang dikembangkan?
- Perintah apa saja yang diberikan ke sistem?
- Kemana perintah yang diberikan itu muncul?
Kepada siapa penerimanya?

Menyusun DFD – Analisis



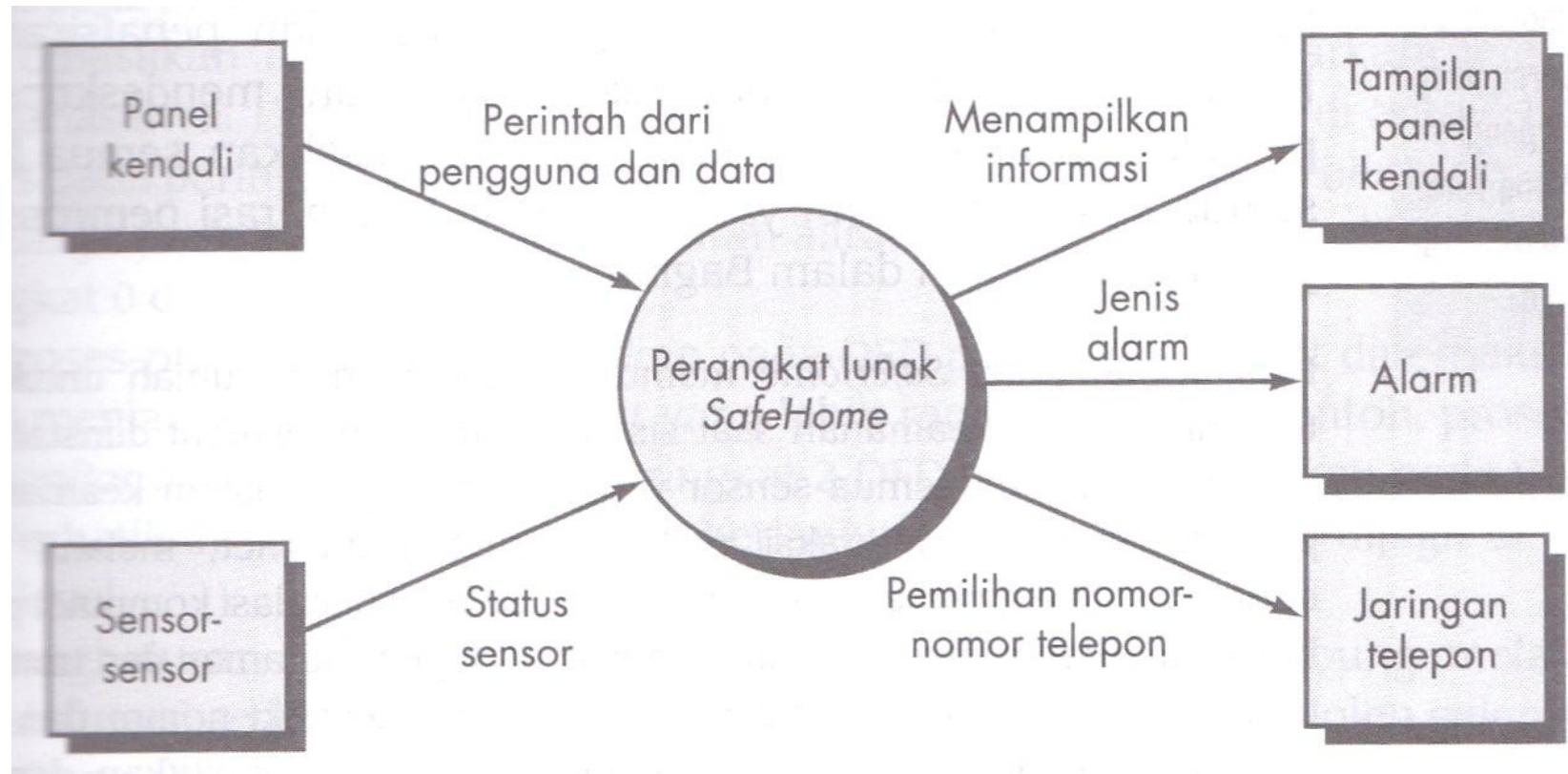
1. Pisahkan **kata benda** (Entitas) & **kata kerja** (aktifitas)
2. Analisis:
 - Aktifitas-aktifitas:
 - Melakukan konfigurasi sistem melalui penel kendali
 - Memantau sensor-sensor melalui panel kendali
 - Berinteraksi melalui panel kendali
 - Mangaktifkan/mnonaktifkan sistem melalui panel kendali
 - Sensor-sensor mengaktifkan alarm
 - Melakukan penggilan telpon saat even terjadi pada sensor
 - Menampilkan pesan-pesan & informasi (status) pada tampilan antarmuka

Menyusun DFD – Analisis



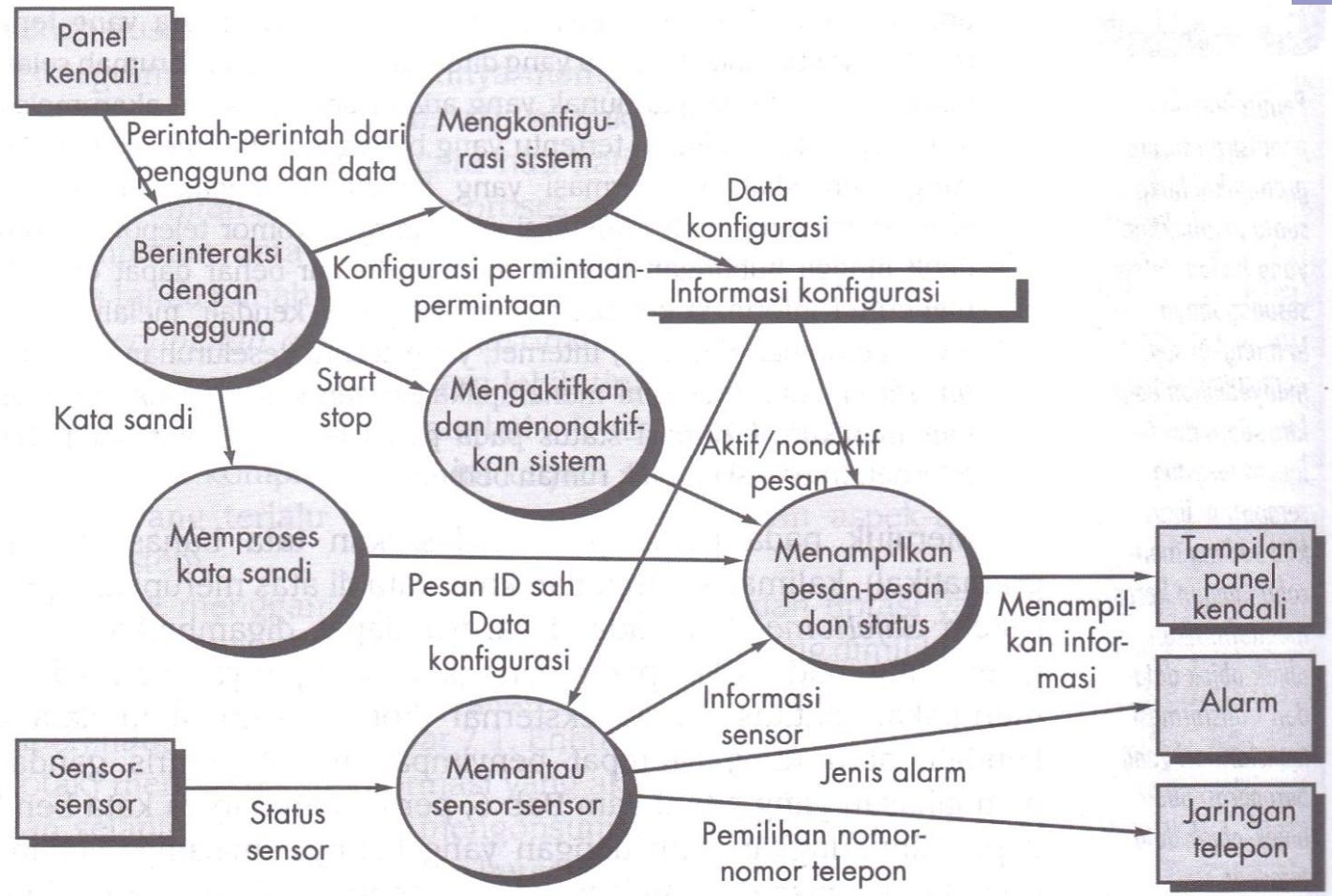
- Perintah/Informasi muncul dari:
 - Panel Kendali, Sensor-sensor
- Penerima perintah/informasi:
 - Alarm, Tampilan Panel Kendali, Nomer Telpon

DFD Level 0 / Context Diagram (CD)



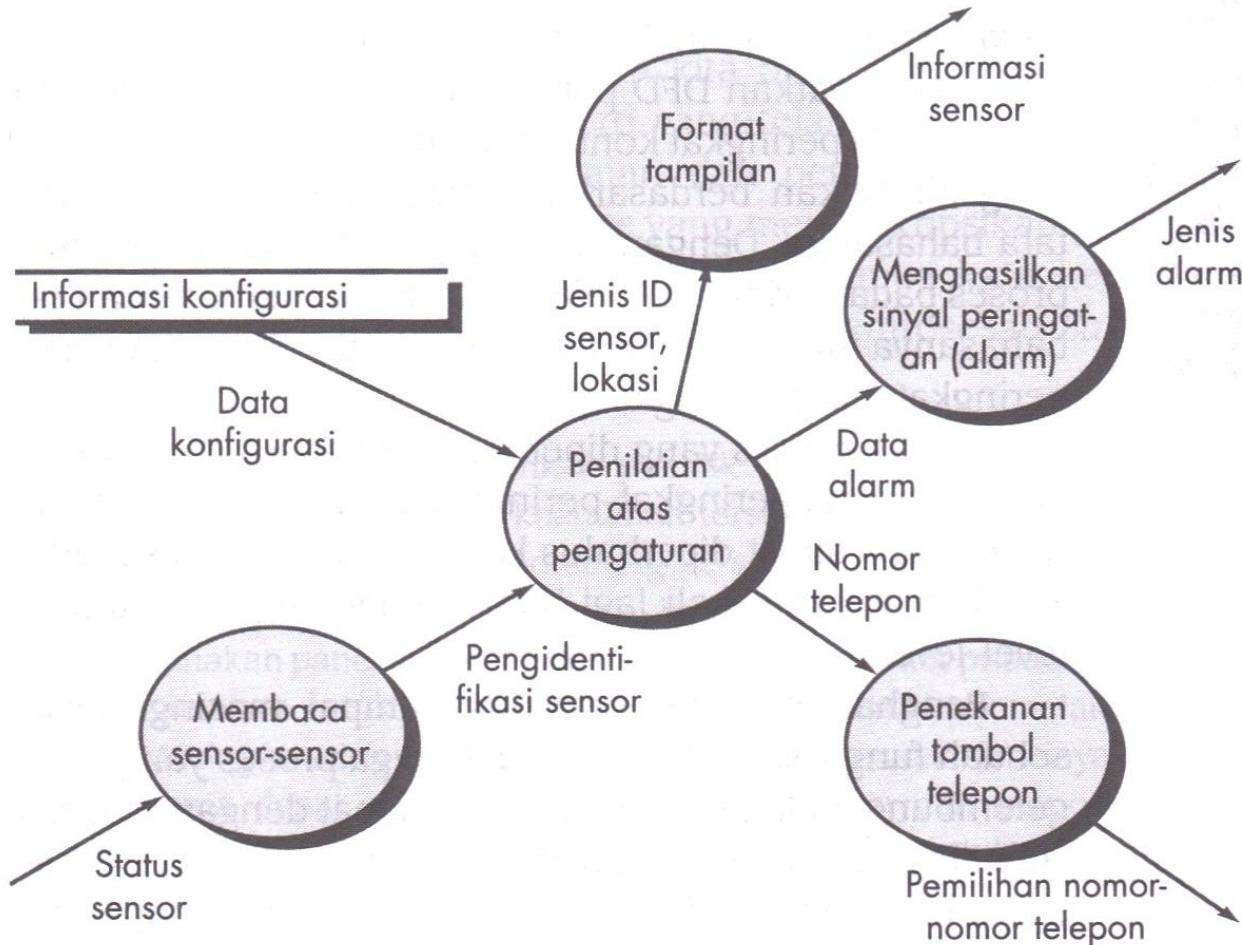
DFD Level 0 / CD Fungsi Keamanan SafeHome

DFD Level 1



DFD Level 1 Fungsi Keamanan SafeHome

DFD Level 2



DFD Level 2 Proses Memantau Sensor-sensor

Latihan DFD



- Perusahaan ingin membuat sistem penggajian, dengan prosedur pegawai melakukan pendaftaran terlebih dahulu pada biro keuangan dengan memberikan data pribadinya. Standar gaji ditentukan berdasar pada tingkat golongan (eselon). Pegawai menerima gaji bersih & slip dengan menghitung keaktifan kerja (presensi), pinjaman (jika ada) dan pajak.
- Rancanglah DFD secara bertingkat (sesuai kebutuhan) pada kasus di atas

REKAYASA PERANGKAT LUNAK



ANALISIS BERORIENTASI OBJEK



Analisis Berorientasi Objek

- Berfokus pada pendefinisian kelas-kelas dan cara bagaimana mereka saling bekerjasama satu dengan yang lainnya untuk memenuhi kebutuhan para pelanggan.
- Pada Paradigma Analysis Design dan Diagram, *Unified Modeling Language (UML)* merupakan perkakas (*tools*) yang digunakan untuk melakukan pemodelan berorientasi objek



1. **Data-oriented** → DFD
2. **Process-oriented** → Flowchart
3. **Object-oriented** (data + process) → UML

What is the UML?

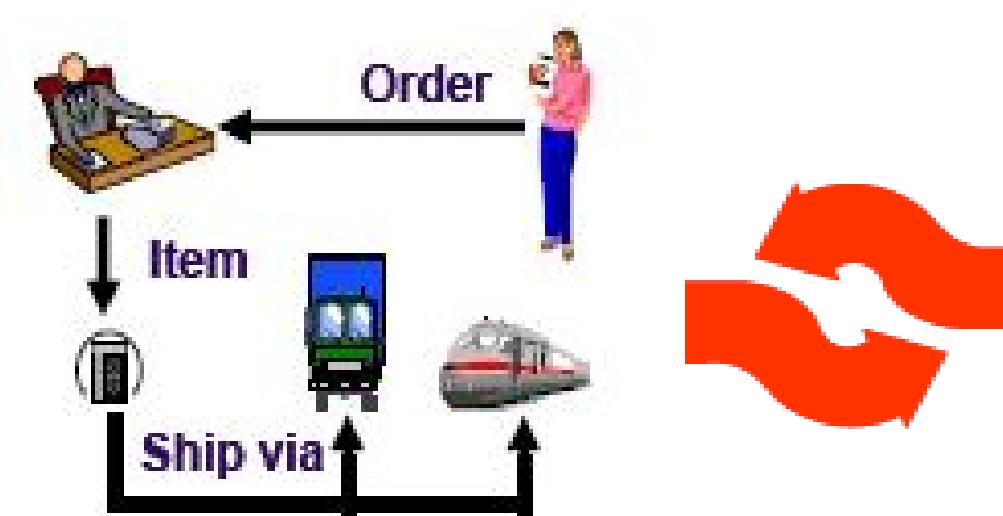


- UML: **Unified Modeling Language**
- UML dapat digunakan untuk **memodelkan semua proses dalam siklus hidup pengembangan** dan seluruh teknologi implementasi yang berbeda
- UML adalah **bahasa standar** untuk memvisualisasikan, menspesifikasi, konstruksi, dan mendokumentasikan artifak dari sistem perangkat lunak
- UML adalah suatu alat **komunikasi** untuk team dan para *stakeholders*

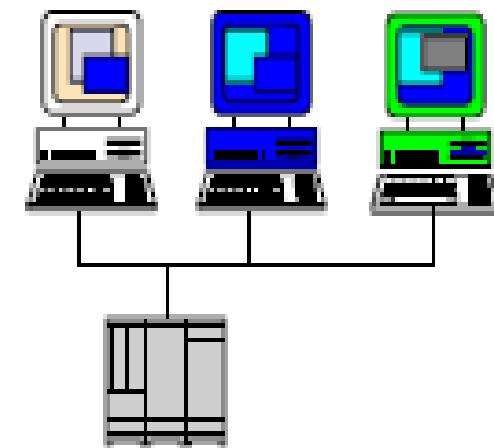
Why Modeling?



Modeling menangkap bagian penting dari sistem
(James Rumbaugh)



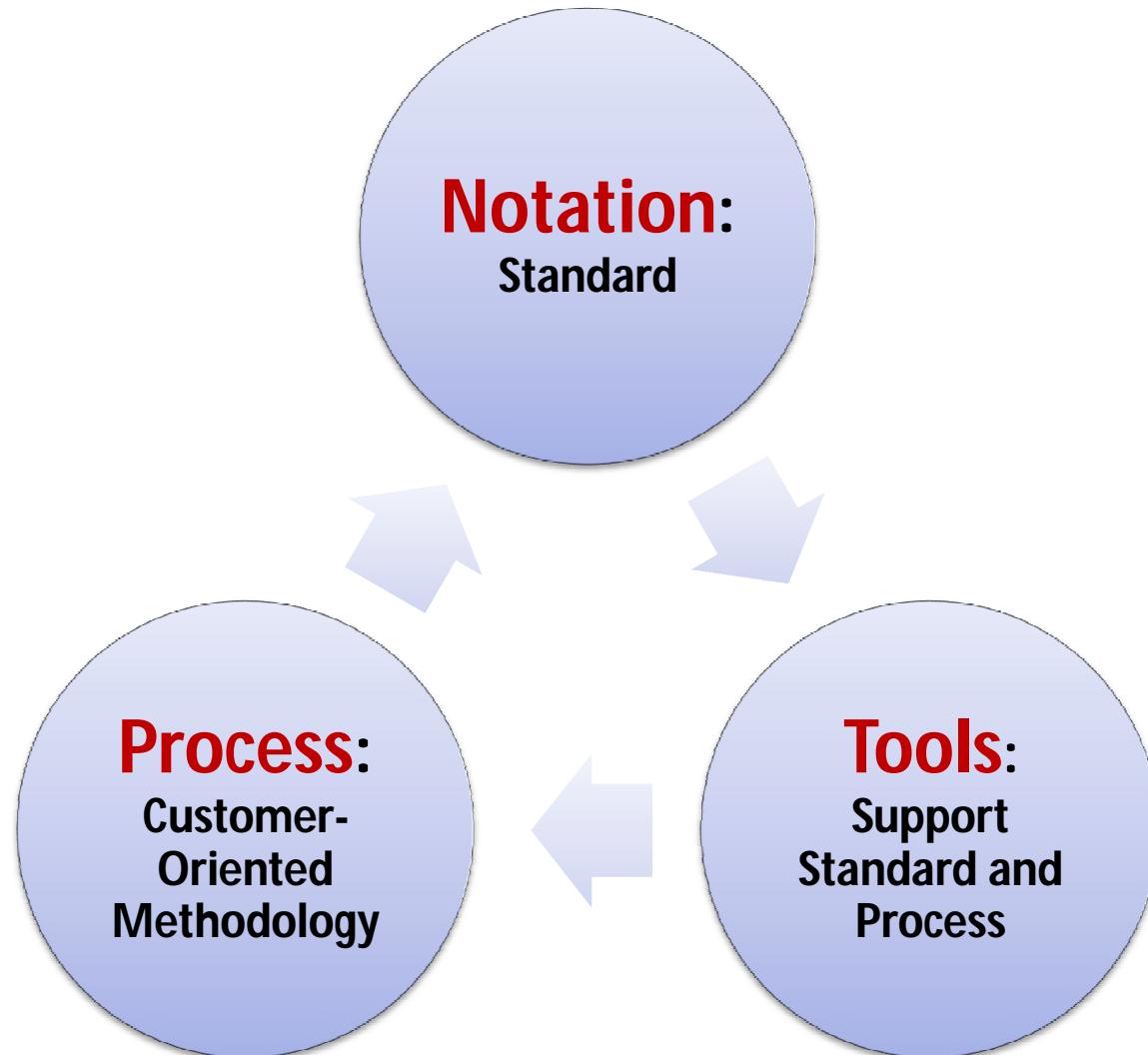
Business Process



Computer System

Visual Modeling adalah pemodelan yang menggunakan **notasi grafik standar**

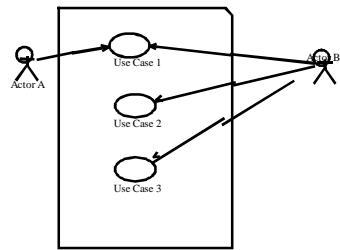
The Triangle of Success in Software Dev.



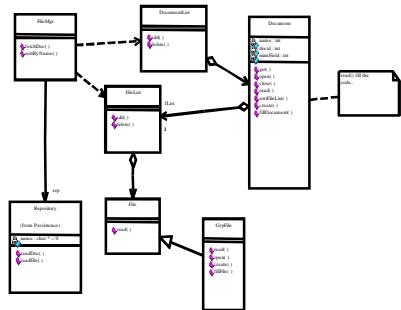
UML Diagrams



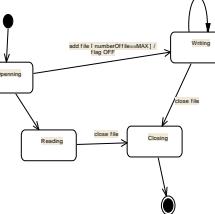
**Use-Case
Diagram**



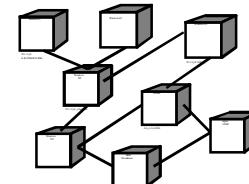
Class Diagram



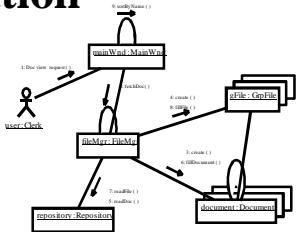
**Statechart
Diagram**



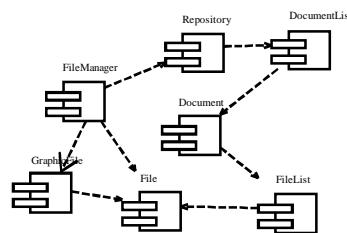
**Deployment
Diagram**



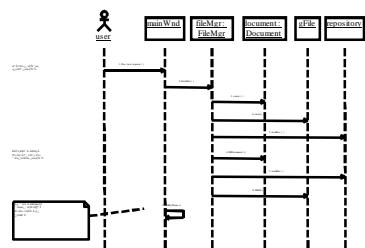
**Collaboration
Diagram**



**Component
Diagram**



**Sequence
Diagram**



**Forward and
Reverse
Engineering**

**Target
System**



UML version 2.0 memiliki 14 diagram yang terbagi pada 2 kelompok besar:

1. **Structure Diagrams**
2. **Behavior Diagrams**

Diagram Name	Used to	Primary Phase
Structure Diagrams		
Class	Illustrate the relationships between classes modeled in the system.	Analysis, Design
Object	Illustrate the relationships between objects modeled in the system. Used when actual instances of the classes will better communicate the model.	Analysis, Design
Package	Group other UML elements together to form higher level constructs.	Analysis, Design, Implementation
Deployment	Show the physical architecture of the system. Can also be used to show software components being deployed onto the physical architecture.	Physical Design, Implementation
Component	Illustrate the physical relationships among the software components.	Physical Design, Implementation
Composite Structure	Illustrate the internal structure of a class, i.e., the relationships among the parts of a class.	Analysis, Design
Behavioral Diagrams		
Activity	Illustrate business workflows independent of classes, the flow of activities in a use case, or detailed design of a method.	Analysis, Design
Sequence	Model the behavior of objects within a use case. Focuses on the time-based ordering of an activity.	Analysis, Design
Communication	Model the behavior of objects within a use case. Focuses on the communication among a set of collaborating objects of an activity.	Analysis, Design
Interaction Overview	Illustrate an overview of the flow of control of a process.	Analysis, Design
Timing	Illustrate the interaction that takes place among a set of objects and the state changes in which they go through along a time axis.	Analysis, Design
Behavioral State Machine	Examine the behavior of one class.	Analysis, Design
Protocol State Machine	Illustrates the dependencies among the different interfaces of a class.	Analysis, Design
Use-Case	Capture business requirements for the system and to illustrate the interaction between the system and its environment.	Analysis

FIGURE 2-6 UML 2.0 Diagram Summary

UML Structure Diagrams



Diagram-diagram yang dikelompokkan ke dalam *Structure Diagram* meliputi:

1. *Class Diagram*
2. *Object Diagram*
3. *Package Diagram*
4. *Deployment Diagram*
5. *Component Diagram*
6. *Composite Structure Diagram*

Structure Diagrams



1. Class Diagrams

- Kosakata umum yang digunakan oleh analis dan pengguna
- Mewakili sesuatu/benda (*employee, paycheck,...*)
- Menunjukkan hubungan antar kelas

2. Object Diagrams

- Mirip dengan Class Diagram
- Gambaran tentang objek-objek dalam sistem
- Hubungan antar objek

3. Package Diagrams

- Kelompok elemen-elemen UML digunakan untuk membentuk tingkat konstruksi yang lebih tinggi

4. Deployment Diagrams

- Menunjukkan **arsitektur fisik** dan komponen perangkat lunak sistem
- For example, network nodes

5. Component Diagrams

- **Hubungan fisik** di antara komponen perangkat lunak
- Example – Client/Server (Mesin mana yang berjalan pada *software* yang mana)

6. Composite Structure

- Menggambarkan struktur internal dari kelas yang kompleks

UML Behavior Diagrams



Diagram-diagram yang dikelompokkan ke dalam
Behavior Diagram meliputi

1. *Activity Diagram*
2. *Sequence Diagram*
3. *Communication Diagram*
4. *Interaction Diagram*
5. *Timing Diagram*
6. *Behavior State Machine*
7. *Protocol State Machine*
8. *Use Case Diagrams*

Behavior Diagrams



1. Activity Diagrams

- Model proses pada suatu sistem informasi
- Example: Business workflows, business logic

2. Interaction Diagrams

- Menunjukkan interaksi antara objek

3. Sequence Diagrams

- Urutan berdasarkan waktu interaksi

4. Communication Diagrams

- Komunikasi antara sekumpulan objek yang berkolaborasi dari suatu aktivitas

Behavior Diagrams



5. Interaction Diagrams

- Kilasan aliran control dari suatu proses

6. Timing Diagrams

- Menunjukkan bagaimana suatu objek berubah dari waktu ke waktu

7. State Machines

- Memeriksa perilaku dari suatu kelas
- Menunjukkan model keadaan-keadaan yang berbeda dan transisi keadaan dari suatu objek

8. Use-Case Diagrams

- Menunjukkan interaksi antara sistem dan lingkungan
- Menangkap kebutuhan bisnis



1. System Analysis

1. Business Process Identification
 - **Use Case Diagram**
2. Business Process Modeling
 - **Activity Diagram**
3. Business Process Realization
 - **Sequence Diagram**

2. System Design

1. Program Design
 1. **Class Diagram**
 2. **Package Diagram** (Gabungan class yang sesuai)
 3. **Deployment Diagram** (arsitektur software dari sistem yang dibangun)
2. **User Interface Design** (Buat UI design)
3. **Entity-Relationship Model** (Buat ER diagram)

THANK YOU!

