

# Temu 04

# Pemodelan Perangkat Lunak

Konsep UML  
Ragam UML

# Konsep UML

# Definisi Unified Modeling Language (UML)

- *“UML adalah bahasa pemodelan standar yang memiliki sintak dan semantik”.*
- *“UML (Unified Modeling Language) adalah bahasa pemodelan untuk sistem atau perangkat lunak yang berparadigma (berorientasi) objek”. Pemodelan (modeling) sesungguhnya digunakan untuk penyederhanaan permasalahan-permasalahan yang kompleks sedemikian rupa sehingga lebih mudah dipelajari dan dipahami.*
- *UML adalah sebuah bahasa yang berdasarkan grafik atau gambar untuk memvisualisasikan, menspesifikasikan, membangun dan pendokumentasian dari sebuah sistem pengembangan perangkat lunak berbasis Objek (Object Oriented programming).*

# Konsep Pemodelan Menggunakan UML

- Sesungguhnya tidak ada batasan yang tegas di antara berbagai konsep dan konstruksi dalam UML,
- tetapi untuk menyederhanakannya, kita membagi sejumlah besar konsep dan dalam UML menjadi beberapa *view*.
- Suatu *view* sendiri pada dasarnya merupakan sejumlah konstruksi pemodelan UML yang merepresentasikan suatu aspek tertentu dari sistem atau perangkat lunak yang sedang kita kembangkan.
- Pada peringkat paling atas, *view-view* sesungguhnya dapat dibagi menjadi tiga area utama, yaitu:
  - klasifikasi struktural (*structural classification*),
  - perilaku dinamis (*dynamic behaviour*), serta
  - pengolahan atau manajemen model (*model management*).

# Bangunan dasar Metodologi UML

Bangunan dasar metodologi Unified Modeling Language (UML) menggunakan tiga bangunan dasar untuk mendeskripsikan sistem/perangkat lunak yang akan dikembangkan, yaitu:

1. Sesuatu (things)
2. Relasi (Relationship)
3. Diagram



# Bangunan dasar Metodologi UML: Sesuatu (things)

1. Struktur things → relatif statis berupa elemen-elemen yang bersifat fisik maupun konseptual
2. Behavioral things
  - bagian yang dinamis pada model UML,
  - biasanya merupakan kata kerja dari model UML,
  - mencerminkan perilaku sepanjang ruang dan waktu
3. Grouping things
  - merupakan bagian pengorganisasian dalam UML
  - penggambaran model yang rumit kadang diperlukan penggambaran paket yang menyederhanakan model.
  - Paket-paket ini kemudian dapat didekomposisi lebih lanjut.
  - Paket berguna bagi pengelompokan sesuatu, misalnya model-model dan subsistem-subsistem
4. Annotational things
  - Merupakan bagian yang memperjelaskan model UML
  - berupa komentar-komentar yang menjelaskan fungsi serta cirri-ciri setiap elemen dalam model UML

# Bangunan dasar Metodologi UML:

## Relasi (Relationship)

### 1. Dependency

- Merupakan relasi yang menunjukkan bahwa perubahan pada salah satu elemen memberi pengaruh pada elemen yang lain.
- Elemen yang ada di bagian tanda panah adalah elemen yang tergantung pada elemen yang ada di bagian tanpa tanda panah



### 2. Association

- Merupakan apa yang menghubungkan antara objek satu dengan objek yang lainnya, bagaimana hubungan suatu objek dengan objek lainnya.
- Suatu bentuk asosiasi adalah agregasi yang menampilkan hubungan suatu objek dengan bagian-bagiannya.
- Umumnya association digambarkan dengan sebuah garis yang dilengkapi dengan sebuah label, nama, dan status hubungannya.

### 3. Generalization

- Merupakan hubungan dimana objek anak (descendent) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (ancestor).
- Arah dari atas ke bawah dari objek induk ke objek anak dinamakan spesialisasi,
- sedangkan arah berlawanan sebaliknya dari arah bawah ke atas dinamakan generalisasi.

### 4. Realization

- Merupakan hubungan semantik antara pengelompokan yang menjamin adanya ikatan di antaranya.
- Hubungan ini dapat diwujudkan di antara interface dan kelas atau elements, serta antara use case dan collaboration.

# Bangunan dasar Metodologi UML: Diagram

- Diagram:
  - berbentuk grafik yang menunjukkan simbol elemen model yang disusun untuk mengilustrasikan bagian atau aspek tertentu dari sistem.
  - Sebuah diagram merupakan bagian dari suatu view tertentu dan ketika digambarkan biasanya dialokasikan untuk view tertentu
- Jenis-jenis diagram:
  - Use Case Diagram
  - Activity Diagram
  - Sequence Diagram
  - Class Diagram
  - Statechart Diagram
  - Collaboration Diagram
  - Component Diagram
  - Deployment Diagram



# Use Case Diagram

- Use case diagram bersifat statis.
- Diagram ini memperlihatkan himpunan use case dan aktor - aktor (suatu jenis khusus dari kelas).
- Diagram ini terutama sangat penting untuk mengorganisasi dan memodelkan perilaku dari suatu sistem yang dibutuhkan serta diharapkan pengguna.
- Use case mendefinisikan "apa" yang dilakukan oleh sistem dan elemen-elemennya,
- bukan "bagaimana" sistem dan elemen-elemennya saling berinteraksi.
- Use case bekerja dengan menggunakan "skenario", yaitu deskripsi urutan-urutan langkah yang menerangkan apa yang dilakukan penggunaan terhadap sistem maupun sebaliknya.
- Use case diagram mengidentifikasi fungsionalitas yang dimiliki oleh:
  - sistem (use case),
  - user yang berinteraksi dengan sistem (aktor), dan
  - asosiasi/ keterhubungan antara user dengan fungsionalitas sistem.

# Use Case Diagram: Aktor



## **Aktor**

menggambarkan  
entitas/subjek yang  
dapat melakukan  
suatu proses.

# Use Case Diagram: Use case



Use case

*Use Case* adalah simbol yang menggambarkan suatu kegiatan (aktivitas) yang terjadi pada sistem.

# Use Case Diagram: Relasi

<<include>>  
→



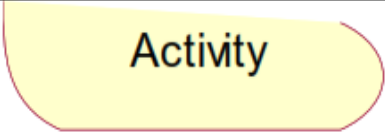
<<extend>>  
→

— →

**Relasi** adalah simbol yang menghubungkan keterkaitan antara *use case* dengan aktor atau dengan *use case* lainnya.

# Activity Diagram


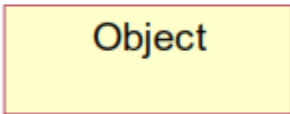

- Secara grafis digunakan untuk menggambarkan rangkaian aliran aktivitas baik proses bisnis maupun use case.
- Activity diagram dapat juga digunakan untuk memodelkan action yang akan dilakukan saat sebuah operasi dieksekusi, dan memodelkan hasil dari action tersebut

Gambar	Keterangan
	<i>Start State</i> adalah simbol yang menyatakan awal dari aktivitas.
	<i>End State</i> adalah simbol yang menyatakan akhir dari aktivitas.
	<i>Activity</i> menggambarkan keadaan dari suatu elemen dalam suatu aliran aktivitas.



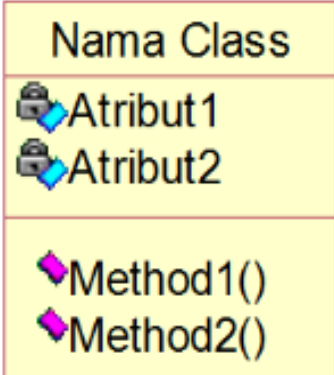

# Sequence Diagram

- Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah scenario,
- diagram ini menunjukkan sejumlah contoh objek dan message (pesan) yang diletakkan di antara objek-objek yang ada di dalam use case".

Gambar	Keterangan
 Aktor	<b>Aktor</b> adalah simbol yang menggambarkan pihak yang berhubungan dengan sistem.
 Object	<b>Object</b> adalah simbol yang menggambarkan suatu objek yang saling berinteraksi.
	<b>Object Message</b> adalah simbol yang menggambarkan alur interaksi antara objek satu dengan objek lainnya.






# Class Diagram

- Class diagram adalah diagram yang digunakan untuk menampilkan beberapa class serta paket-paket yang ada dalam sistem/ perangkat lunak yang sedang dikembangkan.
- Class diagram memberi gambaran/ diagram statis tentang sistem/ perangkat lunak dan relasi-relasi yang ada di dalamnya

Gambar	Keterangan
 A diagram of a class box divided into three horizontal compartments. The top compartment is labeled 'Nama Class'. The middle compartment contains two attributes, 'Atribut1' and 'Atribut2', each preceded by a small icon of a document with a blue tab. The bottom compartment contains two methods, 'Method1()' and 'Method2()', each preceded by a small icon of a document with a blue tab.	<b>Class</b> adalah simbol yang menggambarkan suatu kelas.
 A diagram of an association arrow, consisting of a solid line with an open arrowhead at one end.	<b>Association</b> adalah simbol yang menggambarkan hubungan antar kelas.


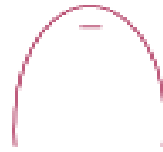
# Statechart Diagram

- Digunakan untuk memodelkan behaviour objek khusus yang dinamis.
- Diagram ini mengilustrasikan siklus hidup objek berbagai keadaan yang dapat diasumsikan oleh objek dan event-event (kejadian) yang menyebabkan objek beralih dari satu state ke state yang lain

Gambar	Keterangan
	<i>State</i> adalah simbol untuk menambahkan suatu state pada diagram
	<i>Start State</i> adalah simbol yang menyatakan awal dari aktivitas.
	<i>End State</i> adalah simbol yang menyatakan akhir dari aktivitas.
	<i>Transition</i> adalah simbol untuk menambahkan transisi pada diagram
	<i>Transition to self</i> adalah simbol untuk menambahkan transisi yang mengarah pada state tunggal

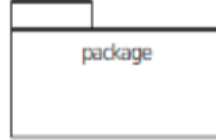
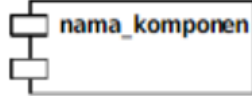
# Collaboration Diagram



- Collaboration Diagram bersifat statis. Diagram kolaborasi adalah diagram interaksi yang menekankan organisasi structural dari objek-objek yang menerima serta mengirim pesan (message).

Gambar	Keterangan
	<b>Aktor</b> adalah simbol yang menggambarkan pihak yang berhubungan dengan sistem.
	<b>Object</b> adalah simbol yang menggambarkan suatu objek yang saling berinteraksi.
	<b>Link Message</b> adalah simbol yang menggambarkan alur interaksi antara objek satu dengan objek lainnya.
	<b>Object Link</b> adalah simbol yang menggambarkan hubungan antara objek satu dengan objek lainnya.
	<b>Link to self</b> adalah simbol yang menggambarkan interaksi objek dengan operasi dalam objek itu sendiri.

# Component Diagram

- Component diagram bersifat statis.
- Diagram komponen ini memperlihatkan organisasi serta kebergantungan sistem/perangkat lunak pada komponen komponen yang telah ada pada sebelumnya.
- Diagram ini berhubungan dengan diagram kelas dimana komponen secara tipikal dipetakan ke dalam satu atau lebih kelas-kelas, antarmuka (interface), serta kolaborasi-kolaborasi.

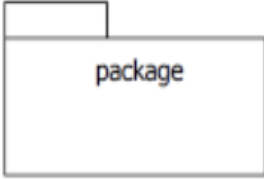
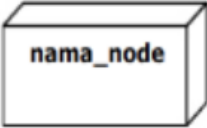


Simbol	Diskripsi
<b>Package</b> 	package merupakan sebuah bungkus dari satu atau lebih komponen
<b>Komponen</b> 	Komponen sistem
<b>Kebergantungan / dependency</b>	Kebergantungan antar komponen, arah panah mengarah pada komponen yang dipakai

Simbol	Diskripsi
<b>Antarmuka / interface</b> 	sama dengan konsep <i>interface</i> pada pemrograman berorientasi objek, yaitu sebagai antarmuka komponen agar tidak mengakses langsung komponen
<b>Link</b> 	



# Deployment Diagram

- Deployment diagram bersifat statis.
- Diagram ini memperlihatkan konfigurasi saat aplikasi dijalankan (saat run-time).
- Diagram ini memuat simpul-simpul (node) beserta komponen-komponen yang ada di dalamnya.
- Deployment diagram berhubungan erat dengan diagram komponen dimana deployment diagram memuat satu atau lebih komponen-komponen.
- Diagram ini sangat berguna saat aplikasi ini berlaku sebagai aplikasi yang dijalankan pada banyak mesin (distributed computing).

Simbol	Deskripsi
<b>Package</b> 	package merupakan sebuah bungkusan dari satu atau lebih node
<b>Node</b> 	biasanya mengacu pada perangkat keras ( <i>hardware</i> ), perangkat lunak yang tidak dibuat sendiri ( <i>software</i> ), jika di dalam node disertakan komponen untuk mengkonsistenkan rancangan maka komponen yang diikutsertakan harus sesuai dengan komponen yang telah didefinisikan sebelumnya pada diagram komponen
<b>Kebergantungan / dependency</b> 	Kebergantungan antar node, arah panah mengarah pada node yang dipakai
<b>Link</b> 	relasi antar node

# Langkah-langkah penggunaan UML (1)

1. Buatlah daftar business process dari level tertinggi untuk mendefinisikan aktivitas dan proses yang mungkin muncul.
2. Petakan use case untuk setiap business process untuk mendefinisikan dengan tepat fungsional yang harus disediakan oleh sistem, kemudian perhalus use case diagram dan lengkapi dengan requirement, constraints dan catatan-catatan lain.
3. Buatlah deployment diagram secara kasar untuk mendefinisikan arsitektur fisik sistem.
4. Definisikan requirement lain non fungsional, security dan sebagainya yang juga harus disediakan oleh sistem.
5. Berdasarkan use case diagram, mulailah membuat activity diagram.
6. Definisikan obyek-obyek level atas package atau domain dan buatlah sequence dan/atau collaboration untuk tiap alur pekerjaan, jika sebuah use case memiliki kemungkinan alur normal dan error, buat lagi satu diagram untuk masing-masing alur.

# Langkah-langkah penggunaan UML (2)

7. Buatlah rancangan user interface model yang menyediakan antar muka bagi pengguna untuk menjalankan skenario use case.
8. Berdasarkan model-model yang sudah ada, buatlah class diagram. Setiap package atau domain dipecah menjadi hirarki class lengkap dengan atribut dan metodenya. Akan lebih baik jika untuk setiap class dibuat unit test untuk menguji fungsionalitas class dan interaksi dengan class lain.
9. Setelah class diagram dibuat, kita dapat melihat kemungkinan pengelompokkan class menjadi komponen-komponen karena itu buatlah component diagram pada tahap ini. Juga, definisikan test integrasi untuk setiap komponen meyakinkan ia bereaksi dengan baik.
10. Perhalus deployment diagram yang sudah dibuat. Detilkan kemampuan dan requirement piranti lunak, sistem operasi, jaringan dan sebagainya. Petakan komponen ke dalam node.
11. Mulailah membangun sistem. Ada dua pendekatan yang tepat digunakan : Pendekatan use case dengan mengassign setiap use case kepada tim pengembang tertentu untuk mengembangkan unit kode yang lengkap dengan test dan pendekatan komponen yaitu mengassign setiap komponen kepada tim pengembang tertentu.