**UNIFIED MODELING LANGUAGE**

**(UML)**

**1. Pengertian UML**

UML (Unified Modelling Language) adalah bahasa visual yang digunakan untuk pemodelan sistem berorientasi objek. UML menyediakan sejumlah diagram untuk menggambarkan berbagai aspek sistem, seperti struktur, perilaku, dan interaksi. UML merupakan standar de facto dalam pengembangan perangkat lunak berorientasi objek. Saat ini UML menjadi bahasa standar dalam penulisan blueprint software (arsitektur)

**2. Pentingnya UML**

Jadi, UML dibuat untuk memudahkan para system developer untuk berdiskusi dengan bahasa pemodelan yang mudah dipahami.

UML diharapkan mampu mempermudah pengembangan piranti lunak (RPL) serta memenuhi semua kebutuhan pengguna dengan efektif, lengkap, dan tepat. Hal itu termasuk faktor-faktor scalability, robustness, security, dan sebagainya.

Berikut adalah 3 poin serta penjelasannya dari manfaat/kegunaan dari UML:

1. Meningkatkan komunikasi dan pemahaman antar tim

UML menggunakan diagram dan simbol yang umum dan standar, sehingga dapat membantu tim-tim dalam proyek yang sama untuk berkomunikasi dan memahami satu sama lain dengan lebih baik. Hal ini dapat meningkatkan efisiensi dan produktivitas tim, serta mengurangi risiko terjadinya kesalahan atau kesalahpahaman.

2. Dapat berguna sebagai blueprint, sistem perangkat lunak

UML dapat digunakan sebagai blueprint sistem perangkat lunak. Hal ini karena UML dapat menggambarkan sistem secara lengkap dan detail, termasuk struktur, perilaku, dan interaksinya dengan lingkungan.

Dengan menggunakan UML sebagai blueprint, tim dapat:

* Memahami sistem secara keseluruhan
* Mengembangkan desain sistem yang lebih efisien dan efektif
* Mendokumentasikan sistem sehingga dapat dipahami oleh semua pihak

Informasi yang detail mengenai koding suatu program dapat diperoleh dari diagram-diagram UML, seperti diagram kelas, diagram objek, dan diagram arsitektur sistem.

3. Mempermudah pemeliharaan dan evolusi sistem

UML dapat digunakan untuk mendokumentasikan sistem secara menyeluruh. Hal ini dapat mempermudah pemeliharaan dan evolusi sistem di masa depan.

Area Penggunaan

- Sistem Informasi Perusahaan

- Bidang Pelayanan Elektronik

- Sistem Perbankan dan Perekonomian

- Bidang Pengetahuan

- Bidang Telekomunikasi

- Bidang Pelayanan Berbasis Web Terdistribusi

- Bidang Transportasi

- Bidang Perdagangan

- Bidang Penerbangan

- dll...

UML adalah alat yang kuat yang dapat membantu tim-tim dalam mengembangkan sistem perangkat lunak yang berkualitas tinggi.

**3. UML & DFD**

UML (Unified Modeling Language) dan DFD (Data Flow Diagram) adalah dua bahasa pemodelan yang digunakan dalam pengembangan sistem. Keduanya memiliki tujuan yang sama, yaitu untuk menggambarkan sistem secara visual sehingga dapat dipahami dengan lebih mudah. DFD lebih berfokus pada menggambarkan aliran data dalam sistem, sedangkan UML lebih berfokus pada menggambarkan struktur dan perilaku sistem.

1. Orientasi

UML berorientasi objek, sedangkan DFD berorientasi data. Hal ini berarti bahwa UML berfokus pada objek-objek dalam sistem, sedangkan DFD berfokus pada data yang mengalir dalam sistem

1. Fokus

UML berfokus pada menggambarkan struktur dan perilaku sistem. UML menunjukkan komponen-komponen yang membentuk sistem, hubungan antar komponen, dan bagaimana komponen-komponen tersebut berinteraksi. UML dapat digunakan untuk menggambarkan sistem secara keseluruhan atau secara detail.

DFD berfokus pada menggambarkan aliran data dalam sistem. DFD menunjukkan bagaimana data masuk ke sistem, bagaimana data diproses dalam sistem, dan bagaimana data keluar dari sistem. DFD dapat digunakan untuk menggambarkan sistem secara keseluruhan atau secara detail.

1. Komplesitas

DFD lebih sederhana daripada UML. DFD menggunakan simbol dan notasi yang lebih sedikit daripada UML.

UML lebih kompleks daripada DFD. UML memiliki lebih banyak jenis diagram dan simbol yang lebih kompleks.

1. Jenis Diagram

UML memiliki 14 jenis diagram, sedangkan DFD hanya memiliki 4 jenis diagram. Jenis-jenis diagram UML dapat digunakan untuk menggambarkan berbagai aspek sistem, mulai dari persyaratan sistem hingga implementasi sistem. Jadi lebih banyak jenis dari UML daripada DFD.

1. Kegunaan

DFD dapat digunakan untuk menggambarkan sistem secara umum, sedangkan UML dapat digunakan untuk menggambarkan sistem secara lebih detail. Misalnya, DFD dapat digunakan untuk menggambarkan aliran data dalam sistem informasi akademik, sedangkan UML dapat digunakan untuk menggambarkan struktur dan perilaku sistem informasi akademik

Lalu, mana yang sebaiknya harus kita gunakan? Secara praktek, DFD digunakan untuk sistem yang memiliki skala kecil dan melibatkan beberapa orang saja di dalam sistem tersebut. Kebalikan dengan UML, perancangan ini biasanya digunakan untuk banyak pihak karena skalanya yang menengah hingga besar, misalnya aplikasi startup. Selain itu biasanya DFD digunakan untuk program prosedural dan UML untuk object oriented maupun prosedural.

**4. Versi UML**

Pada awal 1990-an, banyak cara berbeda digunakan oleh orang untuk merancang program komputer. Setiap orang atau tim memiliki cara sendiri dalam membuat gambaran atau model dari program yang ingin mereka buat. Tetapi, ini membuat sulit jika orang-orang dari tim yang berbeda harus bekerja bersama, karena mereka menggunakan metode yang berbeda-beda.

Di tengah situasi ini, beberapa orang ahli, seperti Grady Booch, Jim Rumbaugh, dan Ivar Jacobson, memimpin usaha untuk menggabungkan semua metode ini menjadi satu standar. Mereka ingin membuat cara yang sama untuk semua orang merancang program. Hasilnya adalah UML, suatu bahasa yang digunakan untuk membuat gambaran tentang bagaimana suatu program komputer harus bekerja. UML ini membuatnya lebih mudah bagi orang-orang dari berbagai tim untuk bekerja sama karena mereka menggunakan cara yang sama dalam merancang program mereka. Unified Modeling Language (UML) telah mengalami sejumlah versi sejak pertama kali diperkenalkan pada tahun 1997. Berikut adalah beberapa versi UML yang paling signifikan

1. UML 1.0: Versi pertama UML diperkenalkan pada tahun 1997. Ini adalah versi awal yang menyediakan notasi dasar untuk pemodelan perangkat lunak.
2. UML 1.1: Dilu ncurkan pada tahun 1998, versi ini mencakup perbaikan dan perbaikan atas UML 1.0.
3. UML 1.3: Tahun 1999 melihat UML 1.3 dengan perbaikan lebih lanjut dan penambahan elemen.
4. UML 1.4: Ini adalah versi penyesuaian UML yang dirilis pada tahun 2001 dengan beberapa perbaikan kecil.
5. UML 2.0: UML 2.0 adalah versi yang lebih signifikan yang diperkenalkan pada tahun 2005. Diagram b aru seperti objek, paket, waktu, interaksi ditambahkan ke UML. Fitur baru ditambahkan ke diagram aktivitas dan sekuens. Diagram kolaborasi diubah nama menjadi diagram komunikasi. Banyak fitur dan perubahan diperkenalkan dalam diagram yang ada.
6. UML 2.1: Diluncurkan pada tahun 2007, UML 2.1 adalah penyempurnaan atas UML 2.0 dengan peningkatan kecil.
7. UML 2.2: Versi ini dirilis pada tahun 2009 dan mencakup perbaikan serta perubahan dalam notasi dan spesifikasi.
8. UML 2.3: Diperkenalkan pada tahun 2010, UML 2.3 adalah versi yang mencakup perbaikan lebih lanjut dan klarifikasi terhadap spesifikasi bahasa.
9. UML 2.4: UML 2.4 dirilis pada tahun 2011 dengan perubahan minor terhadap versi sebelumnya.
10. UML 2.5: Ini adalah versi UML yang paling terbaru, UML 2.5 diperkenalkan pada tahun 2015 dan menyediakan notasi dan spesifikasi terkini untuk pemodelan perangkat lunak. UML 2.5 mencakup peningkatan kecil dan perbaikan pada versi sebelumnya.

Setiap versi membawa perubahan dan pembaruan, mulai dari pengenalan elemen baru hingga perbaikan bug dan peningkatan kinerja. UML berkembang untuk menjadi lebih efisien, sederhana, dan efektif dalam menghasilkan model-model sistem yang kompleks. Pemahaman tentang evolusi UML adalah penting untuk memahami landasan pemodelan yang digunakan dalam pengembangan perangkat lunak modern.

**4. Macam-macam Digram UML**

1. **Class Diagram**

Class Diagram adalah jenis diagram struktur dalam UML yang menggambarkan class, atribut, dan objek dalam suatu sistem, serta hubungan-hubungan antara mereka, seperti pewarisan, containment, dan asosiasi. Class Diagram digunakan untuk menggambarkan komponen-komponen yang ada dalam sistem, termasuk class yang akan diprogram, objek utama, dan interaksi di antara mereka. Class Diagram memiliki beberapa kegunaan penting.

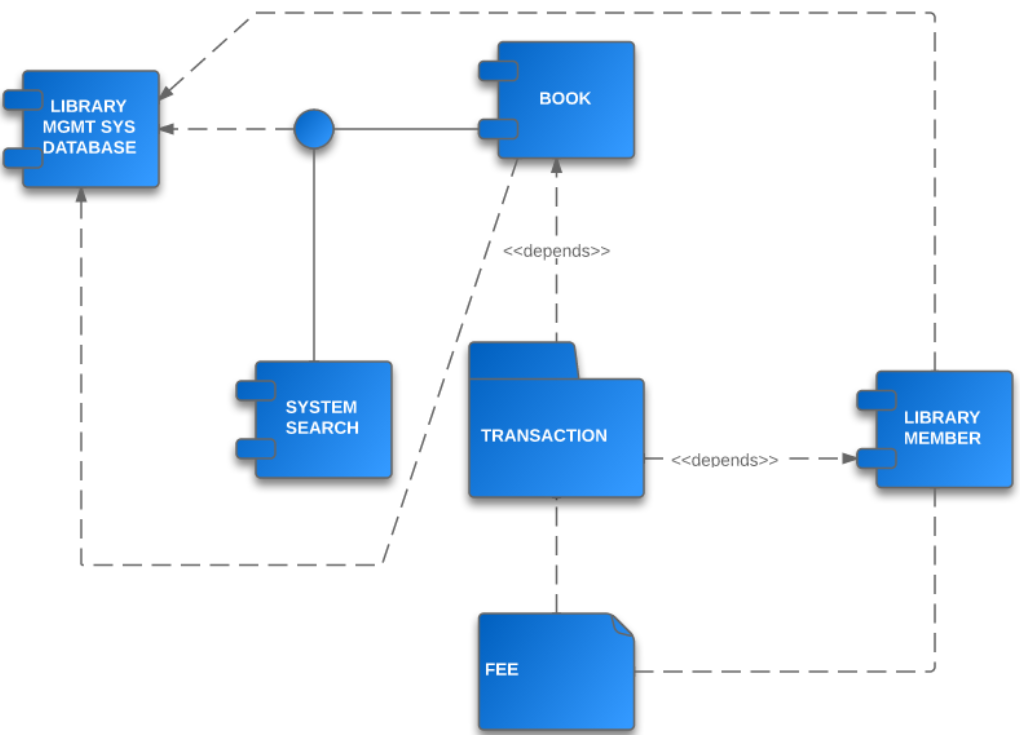
* + Pertama, itu menjelaskan model data untuk program informasi, baik yang sederhana maupun kompleks.
  + Ketiga, Class Diagram memungkinkan penyajian visual kebutuhan spesifik suatu informasi dan dapat berbagi informasi tersebut dengan seluruh bisnis.
  + Keempat, dengan Class Diagram, struktur program dapat dijabarkan dengan rinci dan jelas, memperhatikan kode spesifik yang diperlukan oleh program, sehingga memudahkan implementasi ke dalam struktur yang telah dijelaskan.
  + Terakhir, Class Diagram memberikan penjelasan implementasi-independen dari jenis program yang digunakan, yang dapat dipertukarkan di antara berbagai komponen sistem.

Dengan demikian, Class Diagram tidak hanya membantu menggambarkan struktur sistem, tetapi juga mendukung pemahaman dan implementasi program secara efektif.

1. **Component Diagram**

Component diagram adalah diagram UML yang menggambarkan struktur fisik dari sebuah sistem perangkat lunak. Diagram ini digunakan untuk mengilustrasikan bagaimana kode program dibagi menjadi beberapa komponen, dan mendeksripsikan hubungan antar komponen.

Component diagram biasanya digunakan saat bekerja dengan sistem yang kompleks yang memiliki banyak komponen. Komponen-komponen ini berkomunikasi satu sama lain menggunakan antarmuka. Antarmuka-antarmuka ini dihubungkan menggunakan penghubung.



1. Object Diagram

Object diagram adalah diagram UML yang menggambarkan tampilan lengkap atau sebagian dari struktur sistem yang dimodelkan pada waktu tertentu. Diagram ini menunjukkan objek-objek yang ada dalam sistem, serta atribut dan hubungan antar objek tersebut.

Object diagram bersifat statis, artinya diagram ini tidak menunjukkan perilaku objek-objek tersebut. Namun, diagram ini dapat digunakan untuk memahami struktur sistem secara keseluruhan, serta untuk mengidentifikasi objek-objek yang penting dalam sistem.

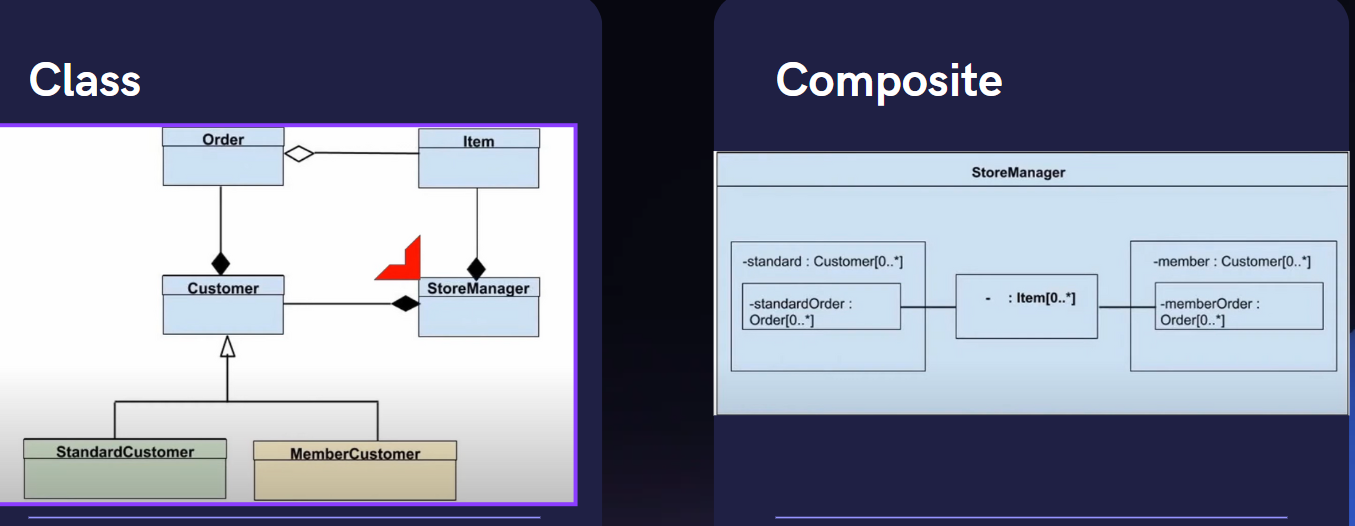
Objek diagram terdiri dari beberapa komponen, yaitu:

* Objek: Sebuah entitas yang mewakili sesuatu dalam dunia nyata atau dalam sistem. Objek dapat berupa orang, benda, tempat, atau konsep.
* Atribut: Karakteristik dari sebuah objek. Atribut dapat berupa data, properti, atau karakteristik lainnya.
* Hubungan: Hubungan antara dua atau lebih objek. Hubungan dapat berupa hubungan dependency, hubungan asosiasi, atau hubungan generalisasi.

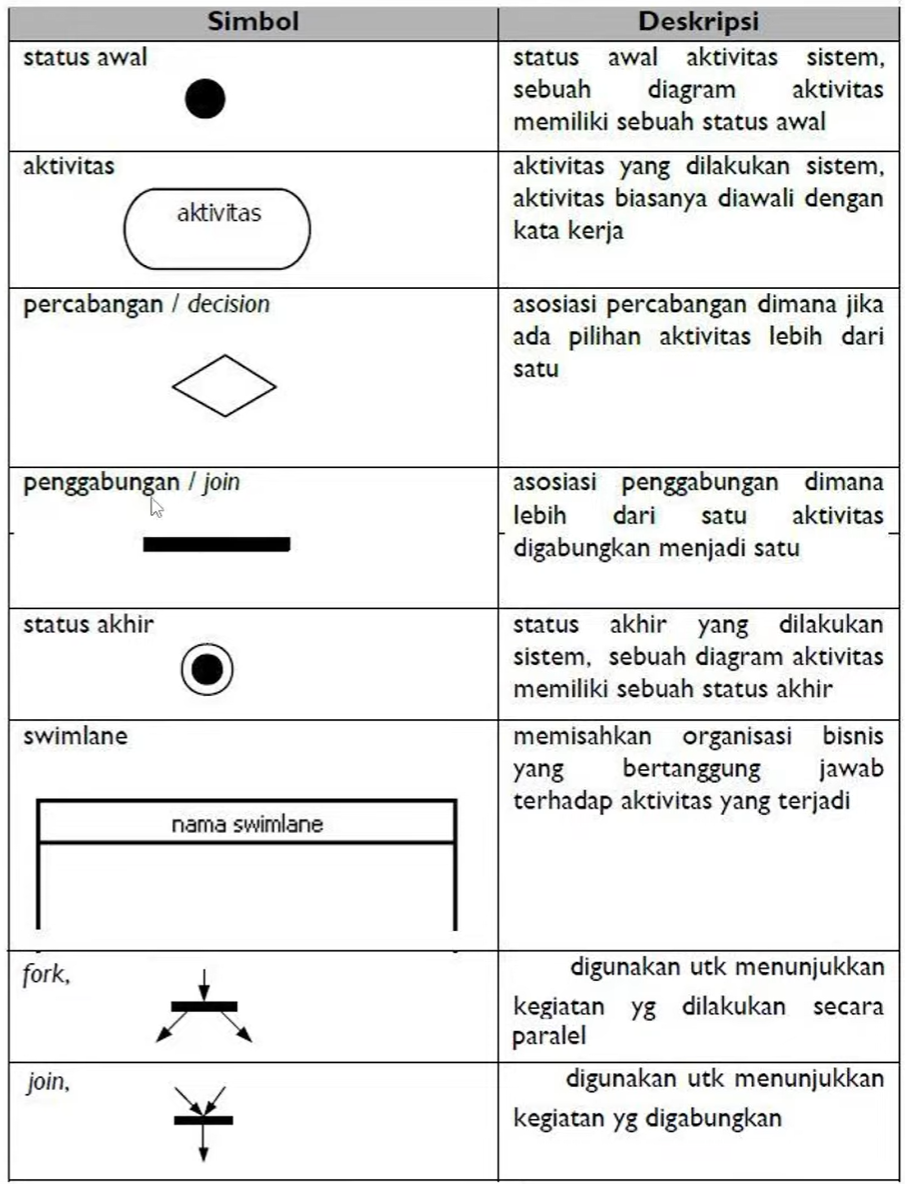
1. Composite Structure Diagram

Composite structure diagram adalah diagram UML yang menggambarkan struktur komposit dari sebuah sistem. Diagram ini menunjukkan bagaimana komponen-komponen sistem dihubungkan satu sama lain untuk membentuk struktur yang lebih besar. Tujuannya untuk memodelkan struktur internal dari (component, class, dan use case), termasuk hubungan pengklasifikasian ke bagian lain dari sebuah program.

Komponen-komponen dalam composite structure diagram dihubungkan satu sama lain dengan menggunakan hubungan asosiasi. Hubungan asosiasi ini dapat berupa hubungan agregasi atau hubungan komposisi.



**Activity Diagram**



Contoh diagram komunikasi

Pada contoh di bawah ini, diagram komunikasi menjelaskan proses untuk menambahkan acara ke kalender. Bahkan dalam contoh sederhana seperti ini, Anda akan melihat perintah dan permintaan yang tepat yang dibagikan di antara berbagai langkah dalam proses. Angka-angka pada setiap baris mewakili urutan dan opsi yang diaktifkan. Kita tahu bahwa beberapa tindakan terjadi secara bersamaan karena penggunaan huruf.