

Flowchart & UML (Unified Modelling Language)

Nama kelompok : Elki Aditia Anugrah

M. Rifandy Syahermi

Jurusan : Pengembangan Perangkat Lunak

Pengertian Flowchart

Flowchart merupakan ilustrasi visual yang menggambarkan alur kerja atau proses dan solusi dari suatu studi tentang sebuah masalah. Setiap alur digambarkan dalam sebuah diagram yang saling terhubung. Nama lain flowchart adalah diagram alir atau bagan alir yang bertujuan untuk menggambarkan suatu tahapan penyelesaian masalah secara sederhana dan ringkas menggunakan simbol-simbol tertentu. Masing-masing simbol tersebut memiliki makna yang berbeda untuk menjelaskan setiap alur atau langkah yang dilakukan. Flowchart bisa dibikin melalui beberapa software, seperti Microsoft Office Word, Microsoft Office Power Point, atau beberapa situs khusus pembuat flowchart, seperti Lucidchart atau Creatly.

Fungsi Flowchart

Secara umum, fungsi flowchart atau diagram alir adalah untuk memberikan sebuah gambaran alur pengerjaan atau proses. Proses digambarkan melalui bagan-bagan atau simbol agar informasi yang disajikan lebih mudah dipahami.

Mengapa Butuh Flowchart

- Menentukan Proses Kerja

Misalnya, jika Anda tidak yakin apakah proses tertentu sepadan dengan waktu dan usaha, Flowchart proses dapat mengungkapkan kemacetan atau duplikasi tugas. Anda dapat menambahkan simbol ke Flowchart proses Anda di mana setiap simbol mewakili orang atau kelompok yang bertanggung jawab atas tugas, keputusan, atau keluaran. Tim manajemen Anda kemudian dapat menentukan apakah ada tim yang kewalahan atau jika beberapa tim tidak diperlukan untuk proses tersebut.

- Mengevaluasi Waktu Tugas

Tujuan penting lain dari Flowchart adalah untuk mengidentifikasi berapa lama waktu yang dibutuhkan untuk menyelesaikan setiap tugas dalam proses tertentu. Biasanya, grafik ini linier dan dibaca dari kiri ke kanan. Anda dapat menambahkan kolom ke bagan waktu untuk menunjukkan bagaimana Anda mengatur waktu setiap tugas sehingga Anda dapat mengevaluasi tugas dalam proses yang berkaitan dengan waktu.

Ini sangat berguna saat Anda menganalisis tugas dan proses berorientasi tenggat waktu karena bagan menunjukkan area di mana tugas memerlukan efisiensi lebih dan area di mana penyelesaian satu tugas bergantung pada penyelesaian tugas lain. Bagan alur yang menganalisis waktu sangat berguna ketika beberapa tim diperlukan dalam suatu proyek, dan komunikasi sangat penting untuk penyelesaian proyek itu.

- Memecahkan Masalah dan Memperbaiki Masalah

Sulit untuk memperbaiki bug dan masalah dalam alur kerja, jika Anda tidak tahu apa yang salah dan pada tahap proses mana yang salah. Itu sebabnya pemecahan masalah adalah tujuan penting dari Flowchart. Untuk merancang Flowchart pemecahan masalah, Anda biasanya mulai dengan mengidentifikasi masalah dan menulis masalah itu dalam kotak di bagian atas Flowchart. Anda kemudian akan menggambar panah ke kotak yang memiliki kemungkinan alasan berbeda untuk masalah itu, dan kemudian kotak untuk solusi yang mungkin, berdasarkan masalah yang Anda identifikasi.

Misalnya, jika Anda telah mengidentifikasi masalah sebagai ketidakpuasan dengan proses layanan pelanggan Anda, Anda kemudian akan menemukan alasan yang berbeda untuk ketidakpuasan itu, yang dapat mencakup waktu tunggu panggilan telepon yang lama, ketidakpuasan dengan waktu pengiriman, ketidakpuasan dengan pengetahuan perwakilan layanan pelanggan Anda atau persepsi bahwa perwakilan Anda kasar dan tidak membantu.

Anda akan mengembangkan solusi yang memungkinkan yang secara khusus membahas alasan pelanggan Anda tidak senang. Ini hanyalah salah satu contoh cara membuat Flowchart pemecahan masalah. Beberapa pemilik bisnis suka menggunakan model benar atau salah di mana masalah ditangani berdasarkan pernyataan benar atau salah sederhana yang ditulis dalam kotak di bagan.

Perkembangan & Sejarah Flowchart

Model atau metode pertama untuk mendokumentasikan sebuah aliran proses atau bagan alir, pertama kali diperkenalkan oleh Frank Gilberth anggota American Society of Mechanical Engineers (ASME) pada tahun 1921. Awal di tahun 1930-an adalah babak baru mengenai flowchart, dimana insinyur industri bernama Allan H. Mogensen melatih orang-orang bisnis dalam menggunakan alat-alat teknik industri dalam kegiatan serta pertemuan kerja di Lake Placid , New York. Pada tahun 1944, dilakukan kembali percobaan oleh Mogensen dan Seni Sepinager, dimana pada saat itu mereka mengembangkan sebuah program namun metodenya dirubah.

Pada tahun yang sama Ben S Graham, Direktur Teknik Formcraft di Standard Register Corporation, mengadaptasi sebuah program alir diagram proses untuk menampilkan dokumen serta hubungannya. Kemudian pada tahun 1947 hal yang baru dilakukan oleh ASME, yang mana mengadopsi beberapa simbol yang berasal dari Gilberth sebagai standar ASME dalam operasi dan Grafik Diagram alur.

Douglas Hartree menjelaskan bahwa Herman Goldstine dan John von Neumann mengembangkan diagram alur (awalnya diagram) untuk merencanakan program computer. Akun modernnya didukung oleh Insinyur IBM dan Goldstine. Pemrograman diagram alur asli Goldstine dan von Neumann dapat dilihat di laporan tidak terbit mereka. “Perencanaan dan Masalah Koding untuk instrument Komputasi Elektronik, Bagian 2, Volume 1” (tahun 1947) dimana diproduksi oleh karya von Neumann. Selain mendeskripsikan logika alur kontrol, Flowchart juga memungkinkan programmer untuk mengeluarkan bahasa program mesin pada memori komputer sebelum mengembangkan bahasa kumpulan dan assambeler.

Penggunaan Flowchart menjadi populer dalam mendeskripsikan algoritma komputer dan masih digunakan pada tujuan ini. Teknik Modern seperti UML, aktifitas diagram dan Diagram Drakon dapat dianggap sebagai ekstensi Flowchart tersebut. Pada tahun 1970 popularitas flowchart sebagai metode kita menurun ketika interaktif sambungan komputer dan generasi ketika bahasa pemrograman menjadi alat umum dari perdagangan, karena algoritma dapat dinyatakan jauh lebih ringkas dan readably sebagai kode sumber dalam suatu bahasa. Hingga saat ini flowchart senantiasa digunakan dalam menggambarkan sebuah proses dari aplikasi, terutama untuk memetakan sebuah algoritma.

Kegunaan Flowchart

Flowchart memiliki kegunaan antara lain :

- Menyederhanakan program yang kompleks dan rumit.
- Memberikan pemahaman kepada tim untuk menggunakan flowchart sebagai metode mengumpulkan data, mendeteksi masalah, mengembangkan perangkat, dan lain sebagainya.
- Flowchart berguna untuk merancang proses baru atau menambahkan fitur tambahan.
- Diagram alir atau flowchart dapat memudahkan komunikasi anggota satu dengan yang lainnya sehingga meminimalisir kesalahpahaman pada sebuah tim.

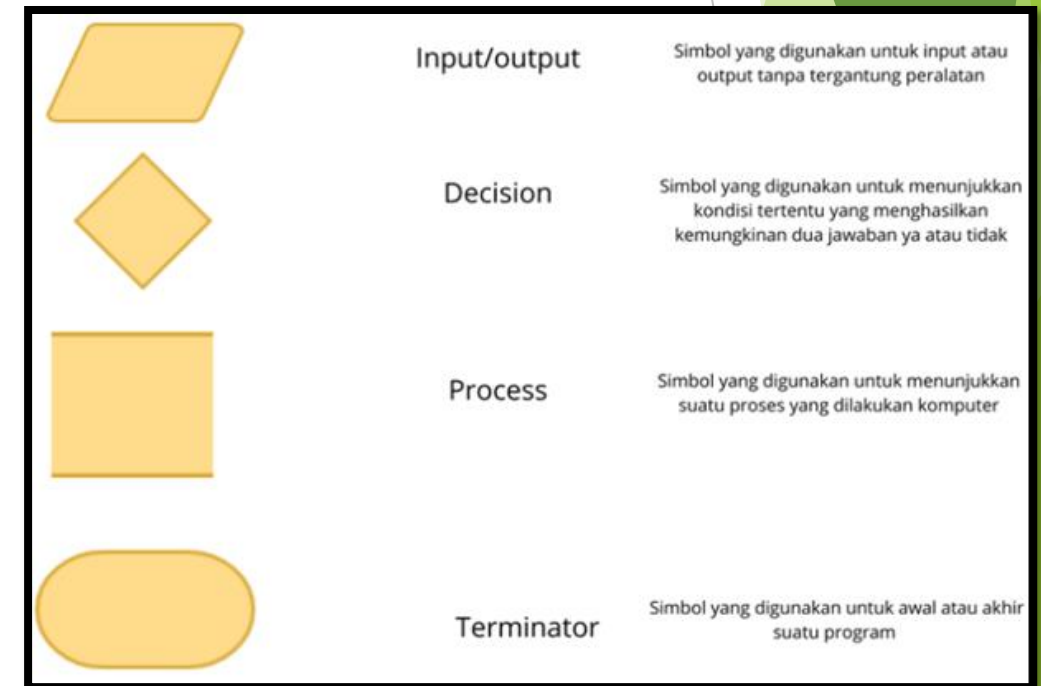
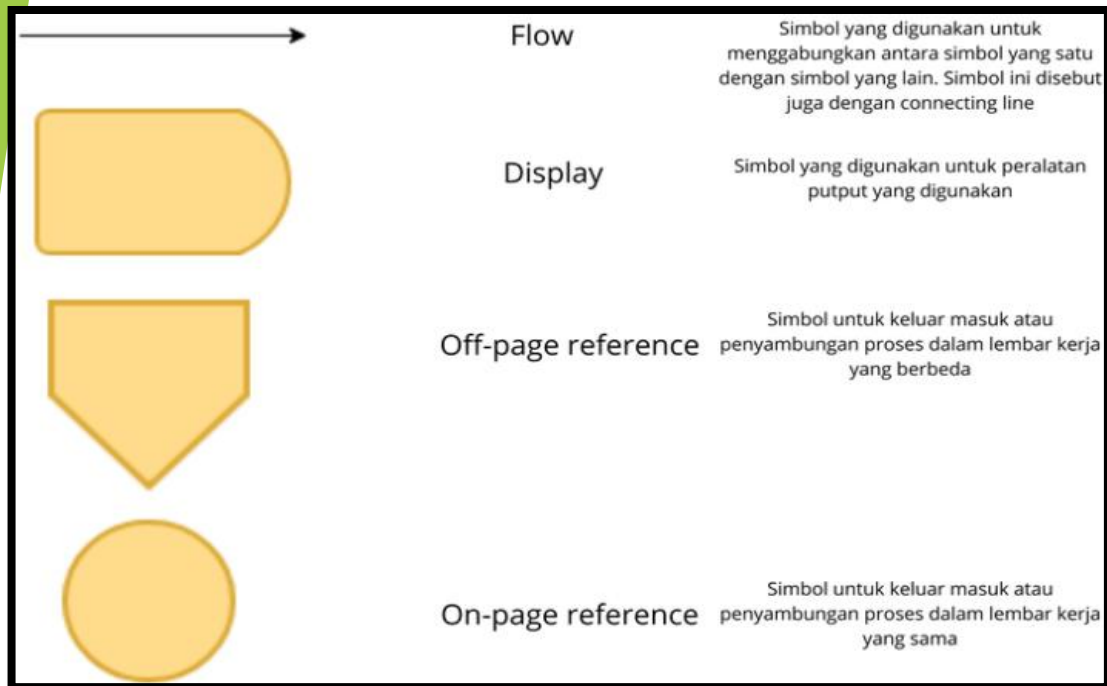
Jenis-Jenis Flowchart

1. Flowchart system. Merupakan diagram alir yang menggambarkan bagaimana tahapan kerja suatu data mengalir dalam sistem dan bagaimana keputusan diambil untuk mengendalikan sebuah peristiwa. Flowchart ini umumnya menggambarkan suatu sistem kerja secara keseluruhan.
2. Flowchart dokumen. Menggambarkan alur keseluruhan seluruh dokumen. Flowchart ini memberikan gambaran seluruh dokumen tanpa harus membacanya terlebih dahulu sehingga membantu pembaca lebih mudah memahaminya.
3. Flowchart skematik. Diagram alir yang menggambarkan suatu skema. Flowchart ini hampir mirip dengan flowchart sistem. Namun flowchart skematik memiliki gambaran lebih detail dan prosedural.
4. Flowchart proses. Merupakan flowchart yang menggambarkan proses suatu kegiatan. Umumnya bagan alir ini digunakan dalam proses produksi di bidang industri sebagai unsur dari analisis sistem produksi.
5. Flowchart program. Bagan alir satu ini menggambarkan alur pemrograman atau algoritma. Flowchart jenis ini biasanya digunakan sebagai patokan dalam membuat daftar program menggunakan bahasa komputer. Flowchart program ini juga biasa disebut dengan notasi algoritma

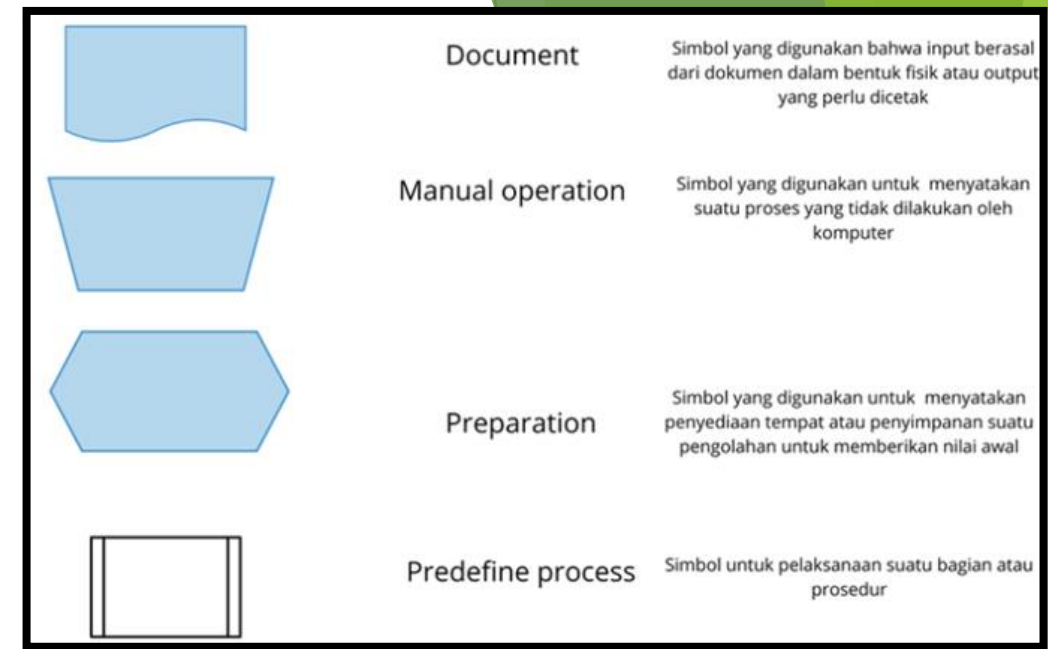
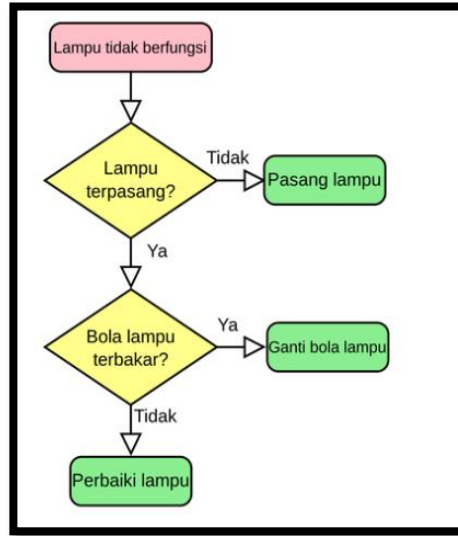
Simbol-Simbol Flowchart

- Flow, biasanya digambarkan dengan tanda panah. Flow disebut juga sebagai connecting line. Simbol ini digunakan untuk menghubungkan antara simbol yang satu dengan simbol yang lain.
- Display, simbol display berguna sebagai penjelasan peralatan output yang digunakan.
- Off-page reference, simbol ini berguna sebagai untuk menandai keluar masuk atau penyambungan proses dalam lembar kerja yang berbeda.
- On-page reference, simbol yang digambarkan dengan bentuk lingkaran ini berfungsi untuk penyambungan proses dalam lembar kerja yang sama
- Input/output, simbol input/output digambarkan dengan jajar genjang yang menggambarkan proses input atau output yang tidak tergantung pada alat.
- Decision, simbol yang berguna untuk menjelaskan kondisi tertentu yang menghasilkan kemungkinan jawaban ya atau tidak.
- Process, simbol process digunakan untuk menggambarkan suatu proses yang dilakukan di computer.
- Terminator, simbol yang menggambarkan sebuah awal atau akhir dari suatu program.

- Document, simbol document digunakan untuk menjelaskan bahwa masukan berasal dari dokumen yang berbentuk fisik atau keluaran yang perlu dicetak.
- Manual operation, simbol ini berfungsi untuk menggambarkan suatu proses yang tidak dilakukan oleh komputer.
- Preparation, simbol untuk menjelaskan suatu penyediaan tempat atau penyimpanan suatu pengolahan data.
- Predefine process, memiliki fungsi untuk melaksanakan suatu bagian atau prosedur tertentu. Untuk simbol-simbol flowchart.



Contoh Flowchart & Cara Membaca Flowchart



Flowchart yang terdiri dari simbol terminator, flow, dan decision di atas menunjukkan apakah lampu berfungsi atau tidak. Mulai dari digambarkan dengan bentuk terminator yaitu awal mula permasalahan "Lampu tidak berfungsi". Setelah itu terminator dihubungkan dengan simbol flow yang dihubungkan dengan simbol decision (simbol yang berbentuk belah ketupat). Simbol tersebut menunjukkan kondisi tertentu yang menghasilkan kemungkinan dua jawaban, pada diagram alir ini menunjukkan apakah lampu sudah terpasang atau bola lampu terbakar. Jika lampu tidak terpasang, maka langkah selanjutnya dihubungkan dengan terminator "pasang lampu" yang menunjukkan akhir dari suatu proses. Jika lampu sudah terpasang namun bola lampu terbakar maka dihubungkan dengan terminator "ganti bola lampu". Atau jika lampu tidak terbakar maka langkah selanjutnya yang dilakukan adalah "perbaiki lampu".

Pengertian UML(Unified Modelling Language)

UML (*Unified Modelling Language*) adalah suatu metode dalam pemodelan secara visual yang digunakan sebagai sarana perancangan sistem berorientasi objek. UML adalah suatu bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan suatu sistem informasi. UML dikembangkan sebagai suatu alat untuk analisis dan desain berorientasi objek oleh Grady Booch, Jim Rumbaugh, dan Ivar Jacobson. UML menyediakan notasi-notasi yang membantu memodelkan sistem dari berbagai perpekstif. UML tidak hanya digunakan dalam pemodelan perangkat lunak, namun hampir dalam semua bidang yang membutuhkan pemodelan. UML diharapkan mampu mempermudah pengembangan piranti lunak (RPL) serta memenuhi semua kebutuhan pengguna dengan efektif, lengkap, dan tepat. Perlu diketahui bahwa sistem yang baik itu berawal dari perancangan dan pemodelan yang matang. Salah satu yang bisa dipraktekkan, yaitu dengan menggunakan UML.

Sejarah UML(Unified Modelling Language)

UML dimulai secara resmi pada Oktober 1994, ketika Rumbaugh menggabungkan kekuatan dengan Booch. Mereka berdua lalu bekerja bersama di Relational Software Cooperation. Proyek ini memfokuskan pada penyatuan metode booch dan Rumbaugh(OMT). Pada bulan October 1995, UML merilis versi 0.8 dan pada waktu yang sama juga Jacobson bergabung dengan Relational. Cakupan dari UML pun semakin meluas. Kemudian dibangunlah persatuan untuk UML dengan beberapa organisasi yang akan menyumbangkan sumber dayanya untuk bekerja, mengembangkan,dan melengkapi UML. Melalui kepemimpinan Booch, Rumbaugh, dan Jacobson, sebuah metodologi yaitu UML 1 diresmikan pada tahun 1997. Metodologi ini memiliki banyak standar untuk tetap mempertahankan konsistensinya di semua diagram. Akhirnya pada tahun 2005, UML 2.0 dirilis dan sebagian besarnya didasarkan pada pendekatan object-oriented. Untuk tetap mengikuti perkembangan yang ada, versi UML 2.x diperbarui dan disempurnakan.

Mengapa Butuh UML(Unified Modelling Language)

Terdapat beragam cara untuk memvisualisasikan sistem Anda, namun Anda mungkin perlu untuk mengetahui mengapa Anda membutuhkan UML. Berikut adalah beberapa alasan Anda membutuhkan UML:

- Diagram UML memiliki peran penting dalam pemecahan masalah. Anda dapat menggambarkan masalah dunia nyata dan secara bertahap menggambarkan cara untuk menyelesaikannya.
- UML memberikan representasi visual dari masalah atau keseluruhan sistem. Dengan cara ini meskipun Anda bukan seorang pengembang, Anda akan dengan mudah memahami bagaimana sistem bekerja.
- Karena diagram mudah dipahami, Anda tidak perlu mempelajari serangkaian istilah sistem untuk mengetahui cara kerja atau proses sistem.
- Dengan menggunakan diagram UML, memungkinkan Anda untuk dapat berkolaborasi dan bekerja sama dengan seluruh tim.
- UML memiliki banyak penerapan aplikasi object-oriented, pengembangan website, pengembangan prototype, analisa bisnis, dan sebagainya.

Fungsi UML(Unified Modelling Language)

Adapun fungsi dari adanya UML yaitu sebagai berikut :

- Dapat memberikan bahasa pemodelan visual atau gambar kepada para pengguna dari berbagai macam pemrograman maupun proses umum rekayasa.
- Menyatukan informasi-informasi terbaik yang ada dalam pemodelan.
- Memberikan suatu gambaran model atau sebagai bahasa pemodelan visual yang ekspresif dalam pengembangan sistem.
- Tidak hanya menggambarkan model sistem software saja, namun dapat memodelkan sistem berorientasi objek.
- Mempermudah pengguna untuk membaca suatu sistem.
- Berguna sebagai blueprint, jelas ini nantinya menjelaskan informasi yang lebih detail dalam perancangan berupa coding suatu program.

Bagian-Bagian UML(Unified Modelling Language)

Bagian-bagian utama dari UML adalah view, diagram, model element, dan general mechanism.

1. View

View digunakan untuk melihat sistem yang dimodelkan dari beberapa aspek yang berbeda. Berikut merupakan jenis view pada UML:

- Use case View Mendeskripsikan fungsionalitas sistem yang seharusnya dilakukan sesuai yang diinginkan external actors. Actor yang berinteraksi dengan sistem dapat berupa user atau sistem lainnya. View ini digambarkan dalam use case diagrams dan kadang-kadang dengan activity diagrams. View ini digunakan terutama untuk pelanggan, perancang (designer), pengembang(developer), dan penguji sistem(tester).

- Logical View Mendeskripsikan bagaimana fungsionalitas dari sistem, struktur statis (class, object, dan relationship) dan kolaborasi dinamis yang terjadi ketika object mengirim pesan ke object lain dalam suatu fungsi tertentu. View ini digambarkan dalam class diagrams untuk struktur statis dan dalam state, sequence, collaboration, dan activity diagram untuk model dinamisnya. View ini digunakan untuk perancang (designer) dan pengembang (developer).
- Component View Mendeskripsikan implementasi dan ketergantungan modul. Komponen yang merupakan tipe lainnya dari code module diperlihatkan dengan struktur dan ketergantungannya juga alokasi sumber daya komponen dan informasi administrative lainnya. View ini digambarkan dalam component view dan digunakan untuk pengembang (developer).
- Concurrency View Membagi sistem ke dalam proses dan prosesor. View ini digambarkan dalam diagram dinamis (state, sequence, collaboration, dan activity diagrams) dan diagram implementasi (component dan deployment diagrams) serta digunakan untuk pengembang (developer), pengintegrasi (integrator), dan penguji (tester).
- Deployment View Mendeskripsikan fisik dari sistem seperti komputer dan perangkat (nodes) dan bagaimana hubungannya dengan lainnya. View ini digambarkan dalam deployment diagrams dan digunakan untuk pengembang (developer), pengintegrasi (integrator), dan penguji (tester).

2. Diagram

Diagram ini berbentuk grafik yang menunjukkan simbol elemen model yang disusun untuk mengilustrasikan bagaian atau aspek tertentu dari sistem. Sebuah diagram merupakan bagian dari suatu view tertentu. Adapun jenis diagram antara lain :

- Use case diagram, yaitu salah satu jenis diagram pada UML yang menggambarkan interaksi antara sistem dan aktor, use case diagram juga dapat men-deskripsikan tipe interaksi antara si pemakai sistem dengan sistemnya.
- Activity Diagram, atau diagram aktivitas yaitu salah satu jenis diagram pada UML yang dapat memodelkan proses-proses apa saja yang terjadi pada sistem.
- Sequence diagram, yaitu salah satu jenis diagram pada UML yang menjelaskan interaksi objek yang berdasarkan urutan waktu, sequence diagram juga dapat menggambarkan urutan atau tahapan yang harus dilakukan untuk dapat menghasilkan sesuatu seperti pada use case diagram.
- Class diagram, yaitu salah satu jenis diagram pada UML yang digunakan untuk menampilkan kelas-kelas maupun paket-paket yang ada pada suatu sistem yang nantinya akan digunakan. Jadi diagram ini dapat memberikan sebuah gambaran mengenai sistem maupun relasi-relasi yang terdapat pada sistem tersebut.

- Statemachine diagram, yaitu salah satu jenis diagram pada UML yang menggambarkan transisi maupun perubahan keadaan suatu objek pada sistem.
- Communication diagram, yaitu salah satu jenis diagram pada UML yang dapat menggambarkan tahapan terjadinya suatu aktivitas dan diagram ini juga menggambarkan interaksi antara objek yang ada pada sistem. Hampir sama seperti sequence diagram akan tetapi communication diagram lebih menekankan kepada peranan masing-masing objek pada sistem.
- Deployment diagram, yaitu salah satu diagram pada UML yang menunjukkan tata letak suatu sistem secara fisik, dapat juga dikatakan untuk menampilkan bagian-bagian software yang terdapat pada hardware dan digunakan untuk menerapkan suatu sistem dan hubungan antara komponen hardware. Jadi Deployment diagram intinya untuk menunjukkan letak software pada hardware yang digunakan sistem.
- Component diagram, yaitu salah satu jenis diagram pada UML yang menggambarkan software pada suatu sistem. Component diagram merupakan penerapan software dari satu ataupun lebih class, dan biasanya berupa file data atau .exe, source code, table, dokumen dsb.
- Object diagram, yaitu salah satu jenis diagram pada UML yang menggambarkan objek-objek pada suatu sistem dan hubungan antarnya.






- Composite structure diagram, yaitu salah satu jenis diagram pada UML yang menggambarkan struktur internal dari pengklasifikasian (class, component atau use case) dan termasuk titik-titik interaksi pengklasifikasian bagian lainnya dari suatu sistem. Ini hampir mirip seperti class diagram akan tetapi composite structure diagram menggambarkan bagian-bagian dari individu kelas saja bukan semua kelas.
- Interaction Overview Diagram, yaitu salah satu jenis diagram pada UML yang berguna untuk memvisualisasikan kerjasama dan hubungan antara activity diagram dengan sequence diagram.
- Package diagram, yaitu salah satu jenis diagram pada UML digunakan untuk mengelompokkan kelas dan juga menunjukkan bagaimana elemen model akan disusun serta menggambarkan ketergantungan antara paket-paket.
- Diagram Timing, salah satu jenis diagram pada UML yang disebut sebagai bentuk lain dari interaksi diagram, dimana fokus yang paling utamanya kepada waktu. Diagram timing berguna untuk menunjukan faktor-faktor yang membatasi waktu antara perubahan state terhadap objek yang berbeda.

Adapun macam-macam diagram UML dibagi menjadi 3, antara lain :

1. Structure diagram merupakan kumpulan diagram yang berfungsi untuk menjelaskan suatu struktur statis dari sistem yang dimodelkan.
2. Behaviour diagram merupakan kumpulan diagram yang digunakan untuk menjelaskan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.
3. Interaction diagram merupakan kumpulan diagram yang berfungsi untuk menjelaskan interaksi sistem dengan sistem lain maupun antar sistem pada sebuah sistem.

Notasi UML(Unified Modelling Language)

NO	GAMBAR	NAMA	KETERANGAN
1		<i>Actor</i>	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan <i>use case</i> .
2		<i>Dependency</i>	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (<i>independent</i>) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (<i>independent</i>).
3		<i>Generalization</i>	Hubungan dimana objek anak (<i>descendent</i>) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (<i>ancestor</i>).
4		<i>Include</i>	Menspesifikasikan bahwa <i>use case</i> sumber secara <i>eksplisit</i> .
5		<i>Extend</i>	Menspesifikasikan bahwa <i>use case</i> target memperluas perilaku dari <i>use case</i> sumber pada suatu titik yang diberikan.

6		<i>Association</i>	Apa yang menghubungkan antara objek satu dengan objek lainnya.
7		<i>Sistem</i>	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
8		<i>Use Case</i>	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
9		<i>Collaboration</i>	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (<i>sinergi</i>).
10		<i>Note</i>	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

Referensi

- <https://teropong.id/pengertian-uml-bagian-bagian-fungsi-dan-notasi-uml/>
- <https://idcloudhost.com/panduan/mengenal-uml-pengertian-dan-sejarahnya/>
- https://id.wikipedia.org/wiki/Unified_Modeling_Language
- <https://www.codepolitan.com/unified-modeling-language-uml/>
- <https://www.dicoding.com/blog/apa-itu-uml/>
- <https://tekno.kompas.com/read/2022/03/19/15300027/pengertian-flowchart-fungsi-jenis-simbol-dan-contoh-serta-cara-bacanya?page=all#:~:text=Secara%20umum%2C%20fungsi%20flowchart%20atau,yang%20disajikan%20lebih%20mudah%20dipahami>
- <https://sekawanstudio.com/blog/pengertian-flowchart/>

**SEKIAN
&
TERIMA KASIH**