**MAKALAH**

**ANALISIS DAN DESAIN PERANGKAT LUNAK**

**“UNIFIED MODELING LANGUAGE (UML)”**



**DISUSUN OLEH:**

**KELOMPOK 6**

* **ANNISA AULIYA RAMADHANI (E1E122003)**
* **DELA PUSPITA HELMI (E1E122007)**
* **GAMALIEL GUSRAYANTO (E1E122054)**
* **LM. YUDHY PRAYITNO (E1E122064)**
* **DIKHSAN DWIRANGGA TIBONG (E1E122092)**

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS HALUOLEO**

**KENDARI**

**2023**

# KATA PENGANTAR

Puji syukur kita panjatkan kehadirat Tuhan Yang Maha Esa, atas berkat rahmat dan hidayah-Nya, sehingga penyusun dapat menyelesaikan tugas makalah yang berjudul “Unified Modelling Language”.

Makalah ini disusun guna menyelesaikan tugas kelompok dari mata kuliah Analisis dan Desain Perangkat Lunak. Penyusun mengucapkan terima kasih kepada Ibu Statiswaty ST., MM.Si. selaku dosen mata kuliah Analisis dan Desain Perangkat Lunak. Berkat tugas yang diberikan ini, dapat menambah wawasan penyusun berkaitan dengan topik yang diberikan.

Dalam penyusunan makalah ini, penyusun menyadari masih jauh dari kata sempurna dan masih terdapat kekurangan serta kesalahan didalamnya. Oleh karena itu, penyusun mengharapkan segala saran dan kritik yang bersifat membangun dari semua pembaca guna menjadi acuan agar bisa lebih baik lagi dimasa yang akan datang.

Semoga makalah “Unified Modelling Language” dapat menambah wawasan bagi para pembaca dan bisa bermanfaat untuk perkembangan dan peningkatan ilmu pengetahuan.

Kendari, 11 Oktober 2023

Penyusun

# DAFTAR ISI

# BAB I

# PENDAHULUAN

## 1.1 Pengertian Unified Modeling Language (UML)

Unified Modelling Language (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan class dan operation dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasabahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C.

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan syntax/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML syntax mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (Object-Oriented Design), Jim Rumbaugh OMT (Object Modeling Technique), dan Ivar Jacobson OOSE (Object-Oriented Software Engineering).

## 1.2 Sejarah Unified Modeling Language (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. 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). Trio ini mengembangkan Ratinal Unified Process (RUP) Banyak perusahaan software merasakan bagaimana pentingnya UML dalam tujuan strategis mereka, sehingga beberapa perusahaan membentuk sebuah konsorsium yang terdiri dari perusahaan-perusahaan seperti:

1. Microsoft
2. Oracle
3. IBM
4. Hewlett-Packard
5. Intellicorp
6. I-Logix
7. DEC, Digital Equipment Corp
8. Texas instrument

Dari konsorsium tersebut pada bulan Januari 1997 lahirlah UML versi 1.0. Pada bulan September 1997 lahirlah UML versi 1.1, dengan 8 buah diagram, yaitu:

1. Use case diagram

2. Activity diagram

3. Sequence diagram

4. Collaboration diagram

5. Class diagram

6. Statechart diagram

7. Component diagram

8. Deployment diagram

Pada bulan November 1997 sebuah organisasi non profit standarisasi Object Management Group (OMG) mengakui UML sebagai sebuah bahasa pemodelan standar untuk aplikasi object oriented.

OMG didirikan pada bulan April 1989 oleh sebelas perusahaan software, dengan kantor pusat di Needham, MA, USA. (www.omg.org) Pada tahun 1999 lahirlah UML versi 1.3, menjadi 9 buah diagram, dengan penambahan:

* Business use case Diagram.

Pada May 2001 lahirlah UML versi 1.4, menjadi 10 buah diagram, dengan penambahan:

* Object Diagram

Pada tahun 2002 lahirlah UML versi 2.0, menjadi 13 buah diagram, dengan penambahan dan penggantian yaitu :

1. Use Case Diagram

2. Activity Diagram

3. Sequence Diagram

4. Communication Diagram (Collaboration diagram in versi 1.x)

5. Class Diagram

6. State Machine Diagram (Statechart diagram in versi 1.x)

7. Component Diagram

8. Deployment Diagram

9. Interaction Overview Diagram

10. Object Diagram

11. Package Diagram

12. Timing Diagram

## 1.3 Rumusan Masalah

1. Apa saja versi-versi Unified Modellling Language (UML)?
2. Apa saja jenis-jenis diagram Unified Modellling Language (UML)?

## Tujuan

1. Untuk mengetahui versi-versi Unified Modellling Language (UML).
2. Untuk mengetahui jenis-jenis diagram Unified Modellling Language (UML).

# BAB II

# PEMBAHASAN

## 2.1 Versi Unified Modeling Language (UML)

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: Diluncurkan 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. Ini mendefinisikan notasi yang lebih ekspresif dan memperkenalkan banyak konsep baru, termasuk Diagram Aktivitas dan Diagram Komunikasi.
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 hingga saat pengetahuan saya (hingga September 2021). 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 UML mencakup notasi dan elemen tambahan, serta peningkatan dalam spesifikasi bahasa, yang memungkinkan para pengembang perangkat lunak untuk melakukan pemodelan yang lebih efektif dan ekspresif. Penting untuk menggunakan versi yang sesuai dengan kebutuhan proyek Anda dan memahami perbedaan antara versi-versi UML tersebut untuk penggunaan yang lebih efisien.

## 2.2 Diagram Unified Modeling Language (UML)

1. Use Case Diagram

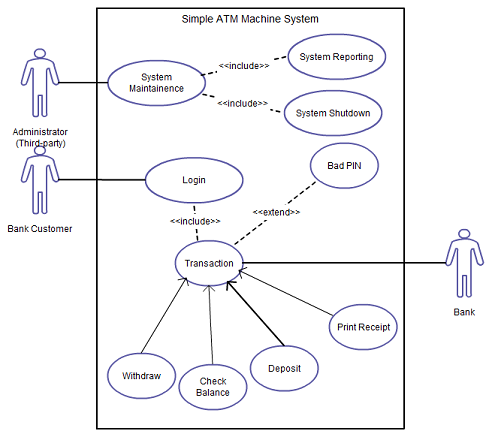
Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah use case merepresentasikan sebuah interaksi antara aktor dengan sistem. Use case merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-create sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.

Use case diagram dapat sangat membantu bila kita sedang menyusun requirement sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang test case untuk semua feature yang ada pada sistem.

Sebuah use case dapat meng-include fungsionalitas use case lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa use case yang di-include akan dipanggil setiap kali use case yang meng-include dieksekusi secara normal. Sebuah use case dapat di-include oleh lebih dari satu use case lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang common.

Sebuah use case juga dapat meng-extend use case lain dengan behaviour-nya sendiri. Sementara hubungan generalisasi antar use case menunjukkan bahwa use case yang satu merupakan spesialisasi dari yang lain.

Contoh Use Case Diagram :



2. Class Diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. Class menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

Class diagram menggambarkan struktur dan deskripsi class, package dan objek beserta hubungan satu sama lain seperti containment, pewarisan, asosiasi, dan lain-lain.

Class memiliki tiga area pokok:

1. Nama (dan stereotype)

2. Atribut

3. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut:

• Private, tidak dapat dipanggil dari luar class yang bersangkutan

• Protected, hanya dapat dipanggil oleh class yang bersangkutan dan anak-anak yang mewarisinya

• Public, dapat dipanggil oleh siapa saja

Class dapat merupakan implementasi dari sebuah interface, yaitu class abstrak yang hanya memiliki metoda. Interface tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah class. Dengan demikian interface mendukung resolusi metoda pada saat run-time.

Sesuai dengan perkembangan class model, class dapat dikelompokkan menjadi package. Kita juga dapat membuat diagram yang terdiri atas package.

Hubungan Antar Class:

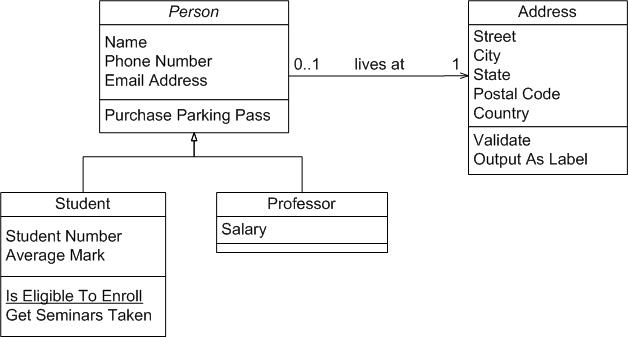
1. Asosiasi, yaitu hubungan statis antar class. Umumnya menggambarkan class yang memiliki atribut berupa class lain, atau class yang harus mengetahui eksistensi class lain. Panah navigability menunjukkan arah query antar class.

2. Agregasi, yaitu hubungan yang menyatakan bagian (“terdiri atas..”).

3. Pewarisan, yaitu hubungan hirarkis antar class. Class dapat diturunkan dari class lain dan mewarisi semua atribut dan metoda class asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari class yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.

4. Hubungan dinamis, yaitu rangkaian pesan (message) yang di-passing dari satu class kepada class lain. Hubungan dinamis dapat digambarkan dengan menggunakan sequence diagram yang akan dijelaskan kemudian.

Contoh Class Diagram :

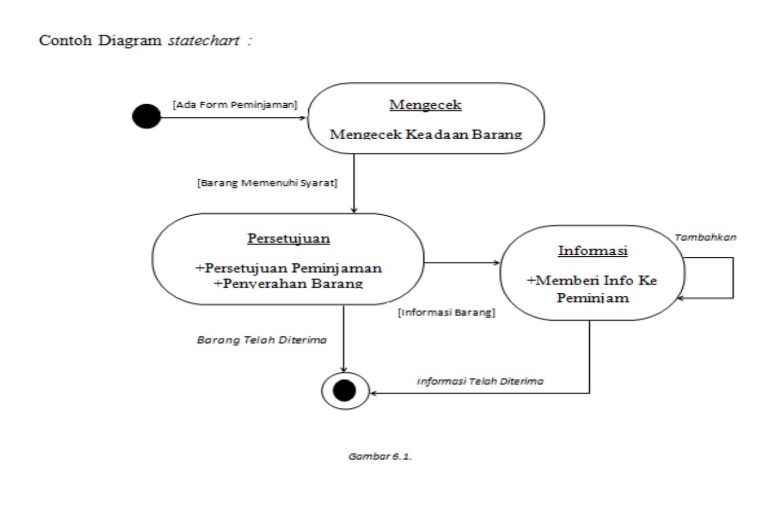


3. Statechart Diagram

Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu state ke state lainnya) suatu objek pada sistem sebagai akibat dari stimuli yang diterima. Pada umumnya statechart diagram menggambarkan class tertentu (satu class dapat memiliki lebih dari satu statechart diagram).

Dalam UML, state digambarkan berbentuk segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar state umumnya memiliki kondisi guard yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku. Action yang dilakukan sebagai akibat dari event tertentu dituliskan dengan diawali garis miring.

Titik awal dan akhir digambarkan berbentuk lingkaran berwarna penuh dan berwarna setengah.



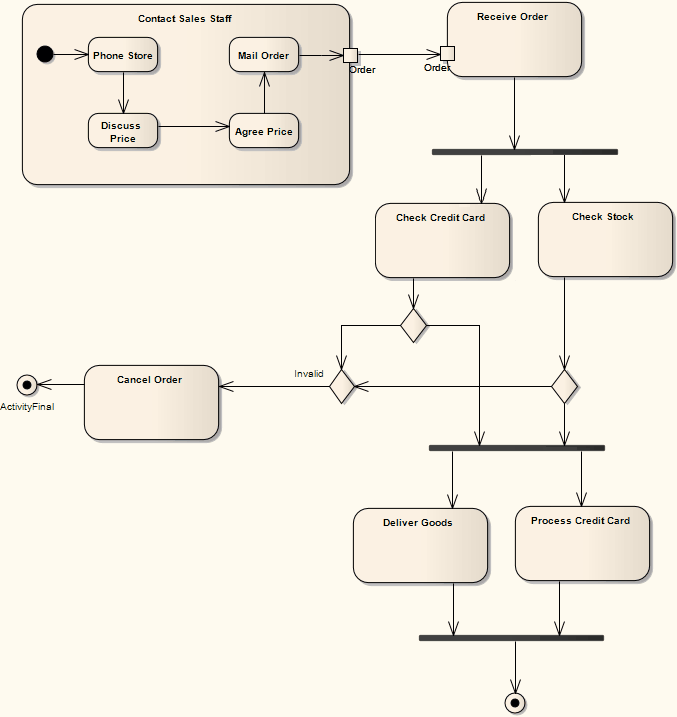
4. Activity Diagram

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, decision yang mungkin terjadi, dan bagaimana mereka berakhir. Activity diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

Activity diagram merupakan state diagram khusus, di mana sebagian besar state adalah action dan sebagian besar transisi di-trigger oleh selesainya state sebelumnya (internal processing). Oleh karena itu activity diagram tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu use case atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara use case menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.

Sama seperti state, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. Decision digunakan untuk menggambarkan behaviour pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (fork dan join) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal. Activity diagram dapat dibagi menjadi beberapa object swimlane untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.

Contoh Activity Diagram:

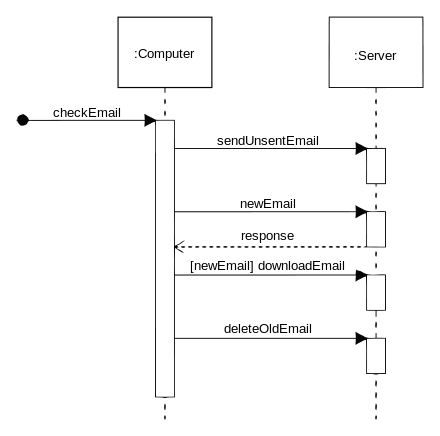
5. Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, display, dan sebagainya) berupa message yang digambarkan terhadap waktu. Sequence diagram terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah event untuk menghasilkan output tertentu. Diawali dari apa yang men-trigger aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki lifeline vertikal. Message digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, message akan dipetakan menjadi operasi/metoda dari class. Activation bar menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah message.

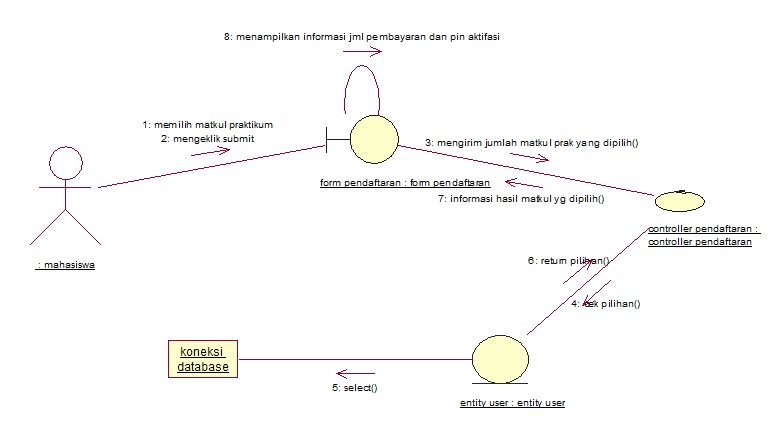
Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan icon khusus untuk objek boundary, controller dan persistent entity.

Contoh Sequence Diagram:

6. Collaboration Diagram

Collaboration diagram juga menggambarkan interaksi antar objek seperti sequence diagram, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian message. Setiap message memiliki sequence number, di mana message dari level tertinggi memiliki nomor 1. Messages dari level yang sama memiliki prefiks yang sama.

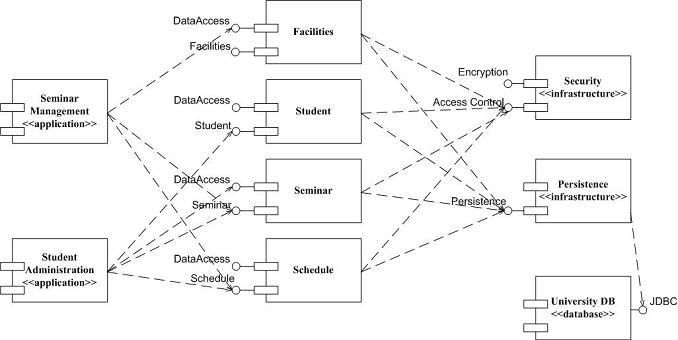
Contoh Collaboration Diagram:



7. Component Diagram

Component diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (dependency) di antaranya.

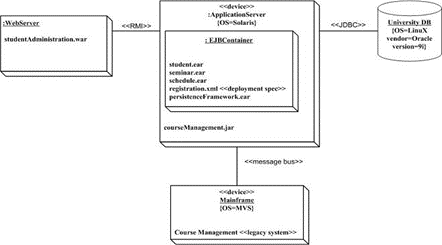
Komponen piranti lunak adalah modul berisi code, baik berisi source code maupun binary code, baik library maupun executable, baik yang muncul pada compile time, link time, maupun run time. Umumnya komponen terbentuk dari beberapa class dan/atau package, tapi dapat juga dari komponen-komponen yang lebih kecil. Komponen dapat juga berupa interface, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.

Contoh Component Diagram:

8. Deployment Diagram

Deployment/physical diagram menggambarkan detail bagaimana komponen di-deploy dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal

Sebuah node adalah server, workstation, atau piranti keras lain yang digunakan untuk men-deploy komponen dalam lingkungan sebenarnya. Hubungan antar node (misalnya TCP/IP) dan requirement dapat juga didefinisikan dalam diagram ini.



1. Interaction Overview Diagram

Interaction Overview diagram yaitu salah satu jenis diagram pada UML yang berguna untuk men-visualisasikan kerjasama dan hubungan antara activity diagram dengan sequence diagram.

A diagram of a room reservation

Description automatically generated

1. Object Diagram

Object diagram yaitu salah satu jenis diagram pada UML yang menggambarkan objek-objek pada suatu sistem dan hubungan antarnya.

A computer screen shot of a computer screen

Description automatically generated

1. Package Diagram

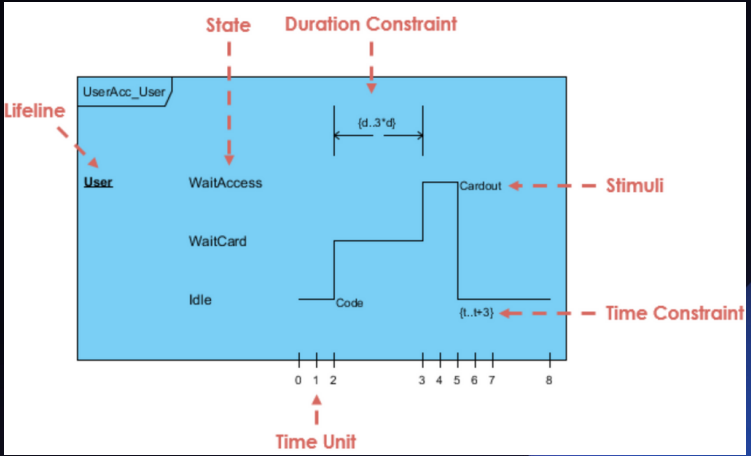
Package Diagram yaitu salah satu jenis diagram pada UML digunakan untuk mengelompokan kelas dan juga menunjukan bagaimana elemen model akan disusun serta mengambarkan ketergantungan antara paket-paket.

A diagram of a business

Description automatically generated

1. Timing Diagram

Timing diagram adalah bentuk lain dari interaction diagram, dimana fokus utamanya lebih ke waktu. Sebuah timing diagram merupakan berntuk khusus dari sequence diagram. Perbedaan antara timing diagram dan sequence diagram adalah sumbunya dibalik, sehingga waktu meningkat dari kiri ke kanan dan lifeline ditunjukkan dalam kompartemen terpisah yang disusun secara vertikal.



# BAB III

# PENUTUP

* 1. **Kesimpulan**

UML, atau Unified Modeling Language, adalah bahasa visual yang krusial dalam pengembangan perangkat lunak karena memberikan cara standar untuk merancang, memodelkan, dan mendokumentasikan sistem perangkat lunak. Dalam penggunaannya, UML memungkinkan para pengembang untuk menggambarkan struktur, fungsi, dan interaksi sistem dengan jelas dan terstruktur. Dengan demikian, UML tidak hanya memfasilitasi komunikasi yang efektif antara pengembang dan pemangku kepentingan, tetapi juga memungkinkan pengembangan perangkat lunak yang lebih terstruktur, terorganisir, dan efisien. Terdapat berbagai versi UML yang berkembang seiring waktu, namun inti dari bahasa ini tetap membantu dalam menghadapi kompleksitas proyek-proyek IT dengan berbagai jenis diagram yang masing-masing menggambarkan karakteristik unik dari sistem yang sedang dirancang.

* 1. **Saran**

Selaku penyusun makalah ini, menyarankan kepada para pembaca menerapkan UML dalam proyek nyata dan mengintegrasikannya dalam siklus pengembangan perangkat lunak. Pengalaman praktis dapat membantu untuk memahami bagaimana UML dapat memberikan nilai tambah dalam merancang dan mengembangkan perangkat lunak. Dan selalu up-to-date dengan versi terbaru dan fitur-fitur baru yang ditawarkan oleh UML. Karena, UML terus berkembang,

# BAB IV

# PERTANYAAN DAN JAWABAN

Pada sesi presentasi materi UML ini tidak ada pertanyaan maupun sanggahan yang dilontarkan oleh kelompok lain. Meskipun tidak ada pertanyaan ataupun sanggahan dari peserta lain pada sesi presentasi materi UML, bukan berarti bahwa materi yang disampaikan telah sempurna. Masih terdapat beberapa hal yang dapat dibahas dan dikembangkan lebih lanjut seperti kelengkapan diagram UML yang digunakan, kesesuaian penggunaan stereotype dan tagged values, keakuratan hubungan antar kelas, kelengkapan proses bisnis pada diagram aktivitas, serta standarisasi penamaan elemen-elemen diagram. Dengan melakukan review dan diskusi lebih lanjut, diharapkan kualitas diagram UML yang dihasilkan akan semakin baik dan sesuai dengan teori serta standar yang berlaku. Partisipasi aktif dari seluruh peserta dalam memberikan masukan dan pertanyaan sangat diperlukan agar materi yang dibahas dapat dipahami dengan komprehensif oleh semua pihak.

# DAFTAR PUSTAKA